International Standard

ISO 21622-3

First edition
2024-03

Irrigation techniques — Remote monitoring and control for irrigation —

Part 3: Interoperability

Reference number
ISO 21622-3:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 23, *Tractors and machinery for agriculture and forestry,* Subcommittee SC 18, *Irrigation and drainage equipment and systems.*.

A list of all parts in the ISO 21622 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The purpose of this document is to allow interoperability between the different elements of an irrigation system that exist side by side in the same installation. The introduction of a standard in irrigation control systems will lead to improvements in irrigation programming practice and establish a clear separation between decision making and decision implementation.

This document establishes the specification and location of communication interfaces to be implemented in order to enable the data exchange on commercial products, as well as its integration in an interoperable architecture.

This document defines the mechanisms to exchange and integrate data from remote monitoring and control systems (RMCS) for irrigation, making it available to any management information system (MIS) with permissions.

A manufacturer can adopt this document in all the described levels or specifically in one of them, implementing only the speficications of application to its product.

— The application of this document to a subsystem (RMCS) involves:

a) the implementation of the architecture defined at Clause 4, including the subsystem and the event interfaces, exclusively with the actions supported by this subsystem. Clauses 5 and 6 are related to the kinds of irrigation entities that can be controlled and/or monitored. Its data will be available to any consumer with permissions that implements the defined interface.

b) the implementation of one of the proposed technologies for the subsystem and the event interfaces. The available implementations are described in the Annexes B or G for subsystem interface, and Annexes C or H for event interface. The developer can use any of these sets (Annexes B and C, or Annexes G and H) to implement this document. The implementation can be validated using the tests for subsystems included in the Annex D.

— The application of this document to a management information system (MIS) involves:

a) the implementation of the architecture defined at Clause 4, including the management interface, exclusively with the actions supported by this MIS. To access data from a subsystem the MIS requires permissions.

b) the implementation of one of the proposed technologies for the management interface. The available implementations are described in the Annexes A or F. The developer can use any of the two to implement the standard. The implementation can be validated using the tests for MIS. included in the Annex D.

— The application of this document to a coordination broker involves:

a) the implementation of the architecture defined at Clause 4 and all its interfaces.

b) the implementation of the interfaces performed using all the technologies used in the implementation annexes (Annexes A, B, C, F, G and H). The implementations can be validated using the tests for coordination brokers, included in the Annex D.

c) Annex E, defines two sets of functionalities for the development of a coordination broker.

Irrigation techniques — Remote monitoring and control for irrigation systems —

Part 3: Interoperability

1 Scope

This document establishes requirements for interoperability among systems developed for management and/or control of irrigation facilities. It can be applied under any technological platform and irrigation system, regardless of the water management scheme, with the exception of irrigation mobile machinery, including centre pivots and linear irrigation machines. Due to the dynamic nature of these machines, associated control systems and specific safety requirements, the applicability of exchangeable data for irrigation mechanical move systems was not fully determined at the time of this publication.

This document does not define hardware or software requirements for any of the systems to which it applies. It only concerns externally visible interfaces and places no restriction on the underlying implementations. It has been designed to avoid interference with proprietary solutions subjected to intellectual property. From the point of view of the data exchange, and to guarantee interoperability based on the previous premises, this document defines three communication interfaces (interface with management, interface with events and interface with subsystems) and the architecture to which these interfaces apply. Three levels of architecture has been defined to accommodate these interfaces:

— The Management Level, where any MIS conforming with this document is located. Out of all available data exchange methods, each MIS only implements those required to execute its functionalities.

— The Higher Control Level: coordination. At this level is performed the data integration among RMCS and the access control to it. A software element, called coordination broker, ensures this integration and allows the use of different technologies in its interfaces.

— The Lower Control Level: RMCS. These can also be referred to as irrigation subsystems. The RMCS perform the duties of the irrigation entity(s) under its control.

A coordination broker can be developed as a product by any of these manufacturers or by an independent developer using the standard specifications for this kind of software.

2 Normative reference

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 8601, *Data elements and interchange formats. Information interchange — Representation of dates and times*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
action
application of a *method* (3.7) to exchange data leading to accomplish typical irrigation duties

3.2
coordination broker
message broker pattern aplication responsible for the mapping of irrigation entities, for the collection and consolidation of their data, and for the management of the *procedural elements* (3.13) executed by them

Note 1 to entry: It shall conform with the management, *subsystems* (3.20) and events interfaces.

3.3
interface with events
functional connection that enables the exchange of information about events (according to IEC 62682) between the *coordination broker* (3.2) and the *subsystems* (3.20)

3.4
interface with management
functional connection that enables the exchange of information between MIS applications and the *coordination broker* (3.2)

3.5
interface with subsystems
functional connection that enables the exchange of information between the *coordination broker* (3.2) and the *subsystems* (3.20)

3.6
message broker pattern
architectural software pattern for message validation, transformation and routing

Note 1 to entry: A message broker is a software product designed as an intermediary to facilitate interactions between third-party applications. The message broker can also be called interface engine or integration broker.

3.7
method
mechanisms established for the exchange of data between interoperable systems

3.8
MIS application
computer program aimed at administrative and/or Operational decision-making in the irrigation entities

Note 1 to entry: A MIS is a tool acquired for an organization (like a User Community or an Irrigator) to execute one or more of the following specific functions (This list is descriptive and not restrictive.):

— administrative control;

— accounting control;

— maintenance;

— behavior modeling;

— operational management; and

— any other purpose aiming at improving decision-making.

3.9
optional conditioned parameter
parameter (3.12) that can be used once a conditioning parameter of an *action* (3.1) is set up

3.10
optional conditioning parameter
parameter (3.12) set up as a part of an action (3.1), that requires the activation of one or more additional required or *optional conditioned parameters* (3.9)

3.11
optional parameter
parameter (3.12) that is not required but can be activated for an *action* (3.1) without requiring the support of additional parameters

3.12
parameter
basic information contained by an *action* (3.1) in the interfaces

3.13
procedural element
specific application of a recipe in an irrigation entity

3.14
procedural model
representation of the reality used to describe the different parts [*procedural elements* (3.13)] of the process to be performed by the elements included in the physical model

3.15
property
required attribute to define an irrigation entity

Note 1 to entry: Note the difference between a parameter, which characterizes an *action* (3.1), and a property, which characterizes an entity

3.16
required parameter
parameter that shall be included when an *action* (3.1) is set up

3.17
required conditioned parameter
parameter required when an *action* (3.1) is set up including an optional or required conditioning parameter

3.18
physical model
representation of the reality used to describe the relations, dependences and hierarchy among the physical assets designed to perform a process

Note 1 to entry: The model is divided in seven levels, being the three upper levels focused on administrative purposes and the four lower levels in the process to be performed.

3.19
standard history
recorded set of values of a *property* (3.15) covering a daily time interval and recorded according to a predefined frequency

3.20
subsystem
term used by RMCS in terms of interoperability

Note 1 to entry: A remote monitoring and control system for irrigation (RMCS), is a set of hardware devices and software programs used to monitor and/or control – according to predefined parameters or user decisions – one or more irrigation entities. Typical components of a RMCS include:

— control software, linking data transmission/acquisition and process control;

— devices controlling or monitoring irrigation entities;

— other intermediate devices required for data integration and/or communication purposes.

4 Interoperability I: System architecture

4.1 Levels and components of an interoperable architecture

4.1.1 General

The levels and components are presented in Figure 1.

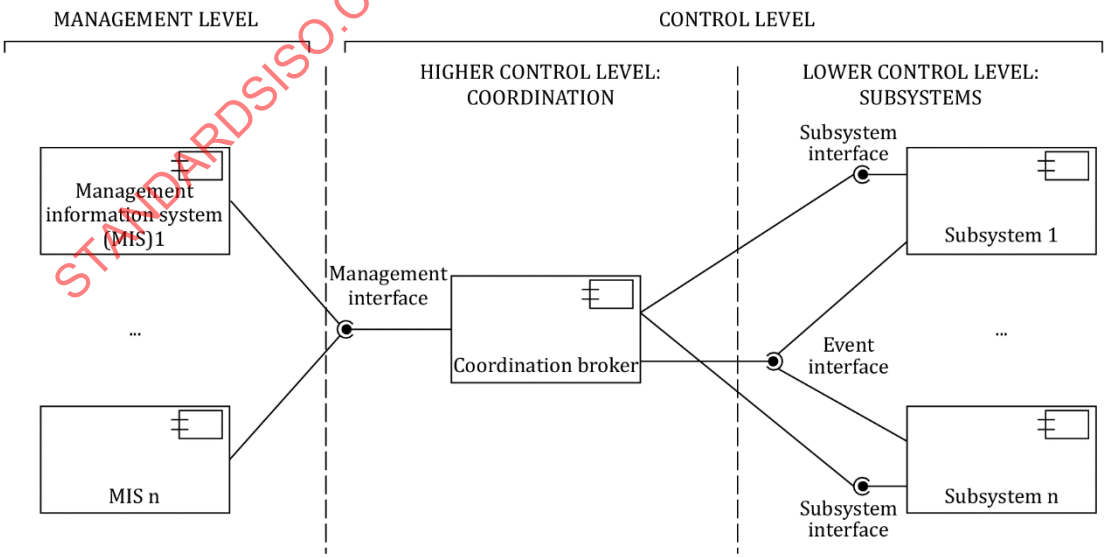


Figure 1 — Levels and components of an interoperable architecture

These architecture levels can be used by irrigators or user communities. Both uses the architecture defined in this document to integrate those management tools and subsystems involved in the management and control of his/her/its own property irrigation entities.

However, it is possible to enable other permissions to facilitate the exchange of data with third parties, always with the prior authorization of the data owner.

4.1.2 Control level

The control level comprises:

— the coordination broker; and

— one or more subsystems.

The subsystem interface is established between these two components.

4.1.2.1 Subsystem

The control of an irrigation entity, understood as the acquisition and continuous monitoring of process variables, execution of operations, changes of set points and emergency stops, among others, is always performed at the subsystem level.

4.1.2.2 Coordination broker

The coordination broker performs the following basic functions:

— mapping irrigation entities and their association to the subsystems controlling them;

— establishing an abstraction layer so that MIS applications do not require information on the subsystems responsible for each irrigation entity;

— collecting and consolidating properties, histories, events and procedural elements inherent to irrigation entities;

— coordinating subsystems, providing them with the information they may require, regardless of its origin;

— managing access permissions among the different components integrated in the interoperable architecture; and

— managing the access to irrigation entities and available methods for any component integrated in the interoperable architecture.

4.1.3 Management level

This level comprises all computer applications necessary for the efficient operation of the hydraulic infrastructure, as well as the processes required for the optimum use of water and energy. MIS applications retrieves the data they require from the control level.

The management interface is stablished between the management level and the upper control level.

4.2 Interface specifications

4.2.1 General

All data exchanged between the architecture levels defined shall be in accordance with the specifications included in this document.

The specifications in this document are not tied to any specific implementation, and only establish the foundations and minimum criteria required to guarantee an effective interoperability performance. The application of this document requires a specific communications protocol at the choice of the manufacturer or developer and adapted to its particular needs. The communication protocol, as well as the security criteria for all the defined interfaces, shall be in accordance with the proposed implementation annexes, Annexes A, B, C for SOAP Web Services or Annexes F, G or H for REST Web Services, depending on the communication protocol selected by the manufacturer.

4.2.2 Data access authorizations

All the interfaces defined in this document shall be secured using authorization systems appropriate for the technology or protocol to be implemented in order to identify the data consumer. The implementation annexes establish the security requirements to be applied in each case.

Any data consumer wishing to use a certain interface shall authenticate itself. The authentication methods to be used for interface access are those described in the implementation annexes.

From its user interface, and in addition to the authorization requirements established, any element of the interoperable architecture (MIS applications, coordination brokers and subsystems) should manage the access permissions for authorized consumers when using its interfaces, allowing:

— the creation of new permissions;

— the deletion of permissions; and

— the modification of this permissions.

This permissions for a data consumer can include access levels restrictions related to:

— the accessible irrigation entities; and

— the allowed methods from those defined for each interface.

Any action not meeting these conditions should be denied.

4.2.3 Irrigation entity identification

The access to any irrigation entity, in all levels of the interoperable architechture, shall be performed using its EntityID, in order to guarantee its univoque identification. This identifier shall be unique among the irrigation entities of an irrigation system.

When introducing or modifiying an irrigation entity, shall be checked that no duplicate irrigation entity identifiers (EntityIDs) exist.

4.2.4 Common methods to management and subsystem interfaces

4.2.4.1 General

Unless otherwise stated, all defined parameters are required and shall be separated by commas.

4.2.4.2 Writing a property of an entity (Write method)

Table 1 — Input parameters of the Write method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a character string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a character string that corresponds to an irrigation entity. This is a unique ID within the system.
PropertyName	PropertyName	Name of the property on which the action shall be executed.	Contains one of the possible names from the list of entity properties.
Value	string	Value of the property to be written.	Contains a character string with the fields corresponding to the value of the property to be written, separated by commas. Only used for non read-only properties.

Table 2 — Output parameters of the Write method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same parameter included in action input.
Response	string	Response obtained.	Contains two fields separated by commas: Value1, Value2 Where: Value1: closed list of integer numbers identifying the response. Possible values: 0, 1, 2, 3, 4 and 5. Value2: closed list with explanatory texts corresponding to each Value1: If Value1=0, “Action successfully executed”. Generated at recipient. If Value1=1, “Execution error”. Generated at recipient. If Value1=2, “Lexical error”. Generated at recipient. If Value1=3, “Not supported”. Generated at recipient. If Value1=4, “Communication error”. Generated by coordination when communication with reception subsystem is not obtained. If Value1=5, “Coordination error”. Generated by coordination when there is a failure in the

Output			
Name	**Type**	**Description**	**Specification**
			application. Value2 might be extended using a hyphen with as many explanatory texts as errors can be discriminated by a subsystem or a coordination broker.

Properties admitting Write method are specified for each entity type. The Response shall have Value1=3 when executed on a read-only property (see Table 26).

4.2.4.3 Reading a property of an entity (Read method)

For the reading of the properties, it is the decision of the subsystem if it returns data stored in the database of its control application or if it forces communication with its remote controller(s) or terminal(s).

Table 3 — Input parameters of the Read method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a character string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a character string that corresponds to a irrigation entity. It is a unique ID within the system.
PropertyName	Property Name	Name of the property on which the action shall be executed.	Contains one of the possible names from the list of properties of the entity.

Table 4 — Output parameters of the Read method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	Contains two fields separated by commas: Value1, Value2 Where: Value1: closed list of integer numbers identifying the response. Possible values: 0, 1, 2, 3, 4 and 5. Value2: closed list with explanatory texts corresponding to each Value1: If Value1=0, “Action successfully executed”. Generated at recipient. If Value1=1, “Execution error”. Generated at recipient. If Value1=2, “Lexical error”. Generated at recipient. If Value1=3, “Not supported”. Generated at

Output			
Name	**Type**	**Description**	**Specification**
			recipient. If Value1=4, “Communication error”. Generated by coordination when communication with reception subsystem is not obtained. If Value1=5, “Coordination error”. Generated by coordination when there is a failure in the application. If Value1=6, “Unavailable”. Generated at recipient for a request about a supported but not in use property. Value2 might be extended using a hyphen with as many explanatory texts as errors can be discriminated by a subsystem or a coordination broker.
Value	string	The value corresponding to the property requested.	Contains a string of fields separated by commas with the value corresponding to the property requested, respecting its format.
TimeStamp	string	Date/time when the read value was generated.	Date when the value was generated at source, using coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.

4.2.4.4 Reading the standard history of a property (ReadStandHist method)

The preparation of the standard history is performed at the control level. The values included in the standard history represents the ones registered during a natural day, establishing three different number of daily values samples: 24 (hourly sample), 48 (half hourly sample) or 96 (quarter hourly sample). The Response should have Value1=2 in requests for the current day.

Table 5 — Input parameters of the ReadStandHist method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a character string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a character string that corresponds to an irrigation entity. It is a unique ID within the system.
PropertyName	PropertyName	Name of the property on which the action shall be executed.	Contains one of the possible names from the list of properties of the entity.
Date	string	Date for which execution of the action is requested.	Date of the historic values in coordinated universal time (UTC) ISO 8601 format 3.17 MMDDhhmmss±hhmm.
NumHist	integer	Number of samples contained in the daily standard history requested.	The number of samples per day have one of the following values: 24, 48 or 96. The default value is 24.

Input			
Name	**Type**	**Description**	**Specification**
Statistics	string	Name of the statistical variable for which the standard history is generated.	Type of statistical variable for the calculation of the standard history. It contains one of the following listed values: — last — average — minimum — maximum The default value is last.

Table 6 — Output parameters of the ReadStandHist method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Reponse in Table 2.
HistStandard Values	string	The list of values of the standard history, according to the input parameters.	Contains a string of fields separated by commas with the value corresponding to the statistical value requested for the property, respecting its format. Each piece of data contains two fields: value and timestamp of the value at source.

4.2.4.5 Creating a procedural element (CreateRecipe method)

Table 7 — Input parameters of the CreateRecipe method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.
ProceduralID	string	ID of the procedural element that is being created via the action.	Contains a string that corresponds to procedural element. It is a unique ID within the system.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to a irrigation entity. It is a unique ID within the system.
EntityType	EntityType	Entity type from those listed.	Type of entity on which the user wishes to execute the action.
RecipeType	RecipeType	Type of procedural element to be created.	Type of procedural element to be created in the irrigation entity identified by EntityID. See operation recipe types in Table 47.
Attending to the selected RecipeType, specific input parameters are required. Optional parameters can also be introduced. The required and optional parameters of all recipe types have been defined in 7.3, 7.4 and 7.5.			

Table 8 — Output parameters of the CreateRecipe method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.

The Response shall have Value1=3 – Not supported when the CREATERECIPE method includes:

— one or more optional parameters not supported by the subsystem controlling the irrigation entity; or

— a recipe type not supported by the subsystem controlling the irrigation entity.

4.2.4.6 Stopping a procedural element (StopRecipe method)

Table 9 — Input parameters of the StopRecipe method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.
ProceduralID	string	ID of the procedural element that is being stopped by means of the action.	Contains a string that corresponds to the unique ID of the porcedural element to be stopped.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to an irrigation entity. It is a unique ID within the system.

Table 10 — Output parameters of the StopRecipe method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.

4.2.4.7 Reading procedural element report (ReadReport method)

Table 11 — Input parameters of the ReadReport method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to a irrigation entity. It is a unique ID within the system.
ProceduralID	string	ID of the procedural element for which the report is required.	Contains a string that corresponds to the procedural element. It is a unique ID within the system.
InstanceNumber	integer	Number of the instance of the procedural element for which the report is required.	Contains an integer that corresponds to the operation instance required, being: — 0, to obtain the original operation report. — 1 to n, to obtain the report of an instance of the original operation. If is not specified, the default value is "0".

Table 12 — Output parameters of the ReadReport method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.
Report	string	Set of data collected with the structure of a report.	Returns the resulting values for a procedural element at the moment of the request. The values shall be separated by commas: The structure of the report shall be in accordance with 6.2. If it is a partial report, the value EndCondition shall be null.
Status	Status	Status of the Procedural element.	Contains one of the values of Table 45.

4.2.4.8 Reading ID of procedural elements in the subsystem for an entity (ReadProceduralIDs method)

Table 13 — Input parameters of the ReadProceduralIDs method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to a irrigation entity. It is a unique ID within the system.
Date	string	Date selected to obtain the ProceduralIDs.	Date in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. The date is, by default, that of the current day.
Status	Status	List of possible statuses of a procedural element.	Filter of the status of the procedural elements for which the user wishes to know the ID. Only the values of Table 45. The default value is “Running”.

Table 14 — Output parameters of the ReadProceduralIDs method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.
ProceduralIDs	string	List of IDs of procedural elements.	Contains a string of fields separated by commas with the values corresponding to the proceduralIDs that meets with the input parameterization. Each proceduralID is composed by: value1=proceduralID, value2=number of instance that matchs with the request conditions. If the operation does not have repetitions, value2=0.

4.2.5 Specific methods of the subsystem interface

4.2.5.1 General

The next methods shall be executed from the coordination broker.

4.2.5.2 Subscription of the coordination broker to the events occurred in an irrigation entity (SubscribeEvent method)

Table 15 — Input parameters of the SubscribeEvent method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action, which is generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to a irrigation entity. It is a unique ID within the system.
Path	string	Path to access to the event server.	Contains the path to enqueue a irrigation entity events.

Table 16 — Output parameters of the SubscribeEvent method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.

4.2.5.3 Unsubscription of the coordination broker to the events occurred in an irrigation entity (UnsubscribeEvent method)

Table 17 — Input parameters of the UnsubscribeEvent method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action, which is generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to an irrigation entity. It is a unique ID within the system.

Table 18 — Output parameters of the UnsubscribeEvent method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table.

4.2.6 Specific actions of the management interface

4.2.6.1 General

The following methods are executed on the coordination broker.

4.2.6.2 Read IDs of all the irrigation entities (ReadEntityIDs method)

Table 19 — Input parameters of the ReadEntityIDs method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.

Table 20 — Output parameters of the ReadEntityIDs method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.
EntityIDs	string	List with all the IDs of the entities known by the coordination broker.	Returns the ID of all the entities, separated by commas, known by the coordination broker.

4.2.6.3 Read static data of an irrigation entity (ReadEntityData method)

Table 21 — Input parameters of the ReadEntityData method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to the irrigation entity. It is a unique ID within the system.

Table 22 — Output parameters of the ReadEntityData method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.
StaticData The output obtained contains the complete list of static data with its values, as defined in 6.2. Any value that does not apply to a certain EntityType is returned with a null value.			

The ReadEntityData method allows the MIS applications to build the structure of the hydraulic network.

4.2.6.4 Reading the ID of procedural elements in the coordination broker for an irrigation entity (GetProceduralIDs method)

Table 23 — Input parameters of the GetProceduralIDs method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action. Generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to an irrigation entity. It is a unique ID within the system.
Date	string	Date selected to obtain the ProceduralIDs.	Date in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. The date is, by default, that of the current day.
Status	Status	List of possible statuses of a procedural element.	Filter of the status of the procedural elements for which the user wishes to know the ID. Only the values of Table 45. The default value is "Running".

Table 24 — Output parameters of the GetProceduralIDs method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.
ProceduralIDs	string	List of IDs of procedural elements that have been filtered by the action.	Contains a string of fields separated by commas with the values corresponding to the proceduralIDs that meets with the input parameterization. Each proceduralID is composed by: value1=proceduralID, value2=number of instance that matchs with the request conditions.

4.2.6.5 Reading the events occurred in an irrigation entity (ReadEvent method)

Table 25 — Input parameters of the ReadEvent method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action, which is generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to a irrigation entity. It is a unique ID within the system.
ProceduralID	string	ID of the procedural element where the user wishes to execute the action.	Contains a string that corresponds to a procedural element. It is a unique ID within the system. The default value is null. If not entered, all events associated with the EntityID are returned.

Input			
Name	**Type**	**Description**	**Specification**
InitialDate	string	Initial date of the requested history.	Date in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
FinalDate	string	Final date of the requested history.	Date in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.

Table 26 — Output parameters of the ReadEvent method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	See Response in Table 2.
EventValues	EventValues	List of events which have taken place in the entity.	Contains a list of one or more events separated by commas, each one with 8 fields also separated by commas. The possible values are listed at Table 27.

All events described in this document, both for entities and for procedural elements, have the definition in the Table 27.

Table 27 — EventValue

Name	Type	Description	Specification
EventID	string	ID of the occurred event.	Contains a string that corresponds to an event ID. It is a unique ID within the subsystem that generates it.
EventName	EventName	Name for a listed event for an entity/procedural element	Contains the event description.
TimeStamp	string	Date/time when the event was generated.	Date when the event was generated at source, using coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
Relevance	integer	Relevance of the event. The relevance allows to establish priority routes for its communication if needed.	Contains an integer with the following possible values: — Relevance=0 implies low relevance. — Relevance=1 implies high relevance. This parameter can be configured at each subsystem, responding to user needs.
ProceduralID	string	ID of the procedural element in which the event has occurred.	Contains a string that corresponds to a procedural element. It is a unique ID within the system.
InstanceNumber	integer	Number of the ProceduralID instance to which the event belongs.	Contains an integer that corresponds to the instance of the ProceduralID where the event occurs.

4.2.7 Specific methods of event interface

4.2.7.1 New event occurred in an irrigation entity (NewEvent method)

Table 28 — Input parameters of the NewEvent method

Input			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Contains a string that identifies the action, which is generated and known by the element executing the method.
EntityID	string	ID of the entity where the user wishes to execute the action.	Contains a string that corresponds to an irrigation entity. It is a unique ID within the system.
EventValue	EventValue	Event which have taken place in the entity.	Contains a list of one or more events separated by commas, each one with 6 fields also separated by commas. The possible values are listed at Table 27

Table 29 — Output parameters of the NewEvent method

Output			
Name	**Type**	**Description**	**Specification**
ActionID	string	ID of the action.	Same element included in action input.
Response	string	Response obtained.	Contains two fields separated by commas: Value1, Value2 Where: Value1: closed list of integer numbers identifying the response. Possible values: 0, 1, 2 and 5. Value2: closed list with explanatory texts corresponding to each Value1: If Value1=0, “Action successfully executed”. Generated at recipient. If Value1=1, “Execution error”. Generated at recipient. If Value1=2, “Lexical error”. Generated at recipient. If Value1=5, “Coordination error”. Generated by coordination when there is a failure in the application. Value2 might be extended using a hyphen with as many explanatory texts as errors can be discriminated by a subsystem or a coordination broker.

4.2.8 Application of the methods on properties

The following restrictions are established on the execution of methods on the set of properties related to the irrigation entities and described in 5.4.

Table 30 — Application of the methods on properties

Properties	Write	Read	ReadStandHist
ActivityStatus	✓	✓	✓
SystemStatus		✓	✓
Mode		✓	✓
CumulativeVolumeOut	✓	✓	✓
InternalFlowOut		✓	✓
PressureOut		✓	✓
OpeningDegree		✓	✓
Temperature		✓	✓
RelativeHumidity		✓	✓
SolarRadiation		✓	✓
WindParameters		✓	✓
Rainfall		✓	✓
CumulatioveVolumeIn	✓	✓	✓
InternalFlowIn		✓	✓
PressureIn		✓	✓
InstantPower		✓	✓
EnergyConsumption		✓	✓
EntityPerformance		✓	✓
PowerOnTime		✓	✓
SoilWaterPotential		✓	✓
SoilWaterContent		✓	✓
SoilTemperature		✓	✓
SoilConductivity		✓	✓
WaterAcidity		✓	✓
WaterConductivity		✓	✓
DifferentialPressure		✓	✓

The rest of methods are not applicable to properties, but to irrigation entities or to procedural elements, so that:

— ReadEvent, CreateRecipe, StopRecipe, ReadProceduralIDs and ReadEntityData methods only apply to irrigation entities through EntityID;

— ReadReport method only applies to procedural elements through ProceduralID;

— ReadEntityIDs method only applies to the coordination broker.

5 Interoperability II: exchange of data from irrigation entities

5.1 General outline

The first part of exchangeable data is associated to the irrigation entities included in the physical model of this document. Table 31 summarizes the entities included in this document. These entities are characterized by a set of properties, static data and events. The static data are related to issues as identification, location, communication, design and historical data to be introduced by the user for management purposes. Among others, the data exchange allows to:

— obtain standardized data from an irrigation entity;

— execute certain actions over these entities;

— monitor their statuses;

— obtain historical data; and

— generate events related.

Table 31 — Irrigation entities of the physical model

Entity name	Acronym	Level	Entity name	Acronym	Level
Sector	IRA	Process cell	Simple irrigation hydrant	HYS/SIH	Equipment module
Pumping station	PMS	Unit	Virtual irrigation hydrant	VIH	Equipment module
Reservoir	RSV	Unit	Irrigation hydrant group	IHG	Equipment module
Irrigation network	INT	Unit	Manual irrigation hydrant	MIH	Equipment module
Solid-set irrigation system	SSS	Unit	Irrigation block	SSB	Equipment module
Fertigation station	FTS	Unit	Dosing module	DSM	Equipment module
Water filtration station	WFS	Unit	Fertilizer tank	FRT	Equipment module
Pumping line	PML	Equipment module	Dilute tank	DLT	Equipment module
Network control point	NCP	Equipment module	Filtering line	FTL	Equipment module
Network branch	NBU	Equipment module			

5.2 Static data of the irrigation entities

5.2.1 General

All irrigation entities have the static data, given in Table 32, for their complete and univocal identification.

Table 32 — Static data of the irrigation entities

Field	Type	Description	Applies to
Identification			
EntityID	string	Unique ID of the entity. It is a unique ID within the system.	All
EntityLevel	EntityLevel	Level of the physical model to which the entity belongs (Table 34 – EntityLevel).	All
EntityType	EntityType	Entity type from those listed in the physical model (Table 35 – EntityType).	All
ControlLevel	ControlLevel	Identification of the control level that is responsible for the irrigation entity between the ones defined in the architecture (Table 36 – ControlLevel).	All
Location			
SpatialData	string	UTM coordinates where the entity is located with the EPSG code associated. The coordinates uses the detailed specification available on 6.2.2.	All
PrecursorEntities	string	Entities that occupy the same level in the physical model and that are the origin of the water received by the entity. See specifications at 6.2.3.	All
SuccessorEntities	string	Entities that occupy the same level in the physical model and that are the destination of the water supplied by the entity. See specifications at 6.2.3.	All
ChildEntities	string	All the lower level entities that comprise the entity according to the physical model.	All
ParentEntity	string	Higher level entity that contains the entity, according to the physical model.	All
Communication			
Path	string	Path to access to the subsystem.	All
CommProtocol	CommProtocol	Communication protocol used by the subsystem controlling the entity (Table 37).	All
Design			
InternalFlowOut	boolean	Informs about whether the extended property exists and is supported y the irrigation entity.	HYS, VIH, MIH, NBU, INT, SSS and NCP
PressureOut	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, NBU, INT, SSS and NCP
DesignPressure	integer	Design pressure of the entity, expressed in Pa.	HYS, VIH, MIH, IHG, NBU, INT, PMS,

Field	Type	Description	Applies to
			PML,SSB, FTS, WFS and FTL.
DesignFlowRate	decimal	Design flow rate of the entity, expressed in m^3/s.	HYS, VIH, MIH, IHG, NBU, INT, PMS, PML,SSB, FTS, WFS and FTL.
Caliber	integer	Caliber of the entity, expressed in m.	HYS, VIH, MIH, NCP
Temperature	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
RelativeHumidity	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SolarRadiation	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
WindParameters	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
Rainfall	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
CumulativeVolumeOut	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	RSV and NCP
CumulativeVolumeIn	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	IHG, NBU, INT, PMS and NCP
InternalFlowIn	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	IHG, NBU, INT, PMS and NCP
PressureIn	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	IHG, NBU, INT, PMS and NCP
SoilWaterContent	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SoilWaterPotential	boolean	Informs about whether the extended property exists and is supported by the	HYS, VIH, MIH, IHG,

Field	Type	Description	Applies to
		irrigation entity.	PMS, RSV, SSS and NCP
SoilTemperature	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SoilConductivity	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
Behavior	Behavior	Contains one of the values established by defining the expected behavior of the entity in Table 37.	IHG
DesignSimultaneity	integer	Design simultaneity for the entities contained by the entity. This parameter is expressed as a percentage, from 0 to 100, without decimals, referring to the maximum number of outlets that can be simultaneously active. In case the value of the admissible simultaneous outputs does not correspond to an integer number, it should be rounded down.	NBU, INT and SSS
DesignPower	integer	Design power of the entity, expressed in W.	PMS, PML, FTS and WFS.
DesignCapacity	integer	Design volume capacity of the entity, expressed in m^3.	RSV
DesignAutonomy	integer	Design autonomy of the water supply without inputs, expressed in days.	RSV
ActivityStatus	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	NCP, INT and NBU
SystemStatus	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	NCP, INT and NBU
Mode	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	INT and NBU
OpeningDegree	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	NCP
InstantPower	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	PMS, PML
EnergyConsumption	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	PMS, PML
EntityPerformance	boolean	Informs about whether the extended property exists and is supported by the	PMS, PML

Field	Type	Description	Applies to
		irrigation entity.	
PowerOnTime	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	PML
Capacity	boolean	Informs about whether the extended property exists and is supported by the irrigation entity.	PMS
Area	integer	Irrigation surface covered, expressed in m^2.	SSS and SSB
Historical			
MeterReplacementEvent	string	Contains three fields separated by comas for each reported replacement, where — value1: Installation date in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. — value2: Old meter reading at the time of replacement, m^3. — value3: New meter reading at the time of replacement, m^3.	HYS, VIH, MIH, IHG and NCP
PowerOnTimeCounterStart	string	Start date of the PowerOnTime property counting. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.	PMS and PML
EnergyMeterCounterStart	string	Start date of the energy meter device counting. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.	PMS
InjectorNumber	integer	Number of injectors existing in the entity, expressed in units.	FTS
DesignCapacity	integer	Design volume capacity of the entity, expressed in m^3.	RSV, FRT and DLT
DesignAutonomy	integer	Design autonomy of the water supply without inputs, expressed in days.	RSV, FRT and DLT
FertigationMethod	FertigationMethod	Fertigation method used at the entity (Table 39).	FTS
FiltrationDegree	integer	Design filtration degree of the entity, expressed in µm.	WFS and FTL
MaxWorkingPressure	real	Maximum pressure for adecuate entity functioning, expresed in Pa.	WFS
MinWorkingPressure	real	Minimum pressure for adecuate entity functioning, expresed in Pa.	WFS

These data shall be accessible from MIS applications through the ReadEntityData method. The next tables provide the list of values of all enumerated parameters defined.

Table 33 — EntityLevel

List of values
Empty
ProcessCell
Unit
EquipmentModule

Table 34 — EntityType

List of values				
IRA	PMS	INT	RSV	SSS
PML	HYS	SSB	NCP	IHG
VIH	MIH	NBU	FTS	DSM
FRT	DLT	WFS	FTL	Empty

It is foreseen that additional irrigation entities (EntityType) can be included in this document as soon as their functional characterization is completely designed and tested.

Table 35 — ControlLevel

List of values
HigherControlLevel
LowerControlLevel

Table 36 — CommProtocol

List of values
Empty
SOAP12
REST

Table 36 can be expanded with new communication protocols in future versions.

Table 37 — Behavior

List of values
Empty
Behavior1
Behavior2

Two possible behaviors, related to IHG irrigation entities, are established:

— Behavior1. The IHG applies restrictions, not allowing simultaneous operations in the VIHs contained by it.

— Behavior2. The IHG applies restrictions, not allowing non simultaneous operations in the VIHs contained by it.

Table 38 — FertigationMethod

List of values
InLineInjection
DilutionTank

5.2.2 SpatialData specification

The spatial representation of any irrigation entity follows the Well-Known Text (WKT) format of the Open Geospatial Consortium (OGC). The representation system uses UTM coordinates, with European Petroleum Survey Group (EPSG). The SpatialData definition contains two fields, separated by commas:

— Field 1: EPSG code represented by five figures to define the entity's Coordinate Reference System.

— Field 2: entity coordinates, attending to WKT format definition for "POINT Z", "LINESTRING Z" and "POLYGON Z" geometries. The entity coordinates have: seven figures and three decimal for X coordinate, nine figures and three decimal for Y coordinate and four figures and three decimal for Z coordinate. All decimals shall be separated by a point.

The level of detail with which an irrigation entity is located is defined by the user when entering its coordinates, bearing in mind that the basic intention of the spatial data is to provide guidance on shape and location. The geometries proposed in Table 39, can be modified by the user as needed.

Table 39 — Geometries for irrigation entities

Entity name	Acronym	Geometry
Sector	IRA	Polygon
Pumping station	PMS	Point
Reservoir	RSV	Polygon
Irrigation network	INT	Polygon
Solid-set irrigation system	SSS	Polygon
Fertigation station	FTS	Point
Water filtration station	WFS	Point
Pumping line	PML	Point
Network control point	NCP	Point
Network branch	NBU	Polygon
Simple irrigation hydrant	HYS/SIH	Point
Virtual irrigation hydrant	VIH	Point
Irrigation hydrant group	IHG	Point
Manual irrigation hydrant	MIH	Point
Irrigation block	SSB	Polygon
Dosing module	DSM	Point
Fertilizer tank	FRT	Point

Entity name	Acronym	Geometry
Dilute tank	DLT	Point
Filtering line	FTL	Point

5.2.3 Topological data specification

5.2.3.1 General

The topological data of the irrigation entities is obtained from its PrecursorEntities and SuccessorEntities static data. Due to the equipment module level recursivity, the following entity types does not have topological information:

— Network branch (NBU).

5.2.3.2 Unit level

Table 40 — Topological data at the unit level

Field	Type	Description
Location		
PrecursorEntities	string	Contains the EntityIDs of all entities that occupy the same level in the physical model and that are the origin of the water received by the entity.
SuccessorEntities	string	Contains the EntityIDs of all entities that occupy the same level in the physical model and that are the destination of the water supplied by the entity.

Certain units can be the water source, so the requirement for a precursor does not apply to PMS and RSV irrigation entities.

Other units can be the water final destination, so the requirement for a successor does not apply to INT and SSS irrigation entities.

5.2.3.3 Equipment module level

Table 41 — Topological data at the equipment module level

Field	Type	Description
Location		
PrecursorEntities	string	Contains four fields for each precursor entity. The fields are separated by comas where: — value1: EntityID of the precursor entity. — value2: Connection length, expressed in metres, between the entity and its precursor entity. — value3: Connection diameter caliber, expressed in metres, between the entity and its precursor entity. — value4: Connection material, expressed in technical acronyms (PRFV, PVC, PEAD, steel, iron, concrete or any other material).

Field	Type	Description
Location		
SuccessorEntities	string	Contains four fields for each successor entity. The fields are separated by comas: — value1, value2, value3, value 4 Where: — value1: EntityID of the successor entity. — value2: Connection length, expressed in metres, between the entity and its successor entity. — value3: Connection diameter caliber, expressed in metres, between the entity and its successor entity. — value4: Connection material, expressed in technical acronyms (PRFV, PVC, PEAD, steel, iron, concrete or any other material).

Certain equipment modules can be the water source, so the requirement for a precursor not applies to NCP and PML irrigation entities.

Other equipment modules shall be the water final destination, so the requirements for a successor do not apply to HYS, MIH, VIH and SSB irrigation entities.

The construction of the hydraulic topology based on the precursor-successor relationships works properly to indicate the water direction in dendritic topology but has limitations for its use in mesh topology, in which the water direction can change. In this case, the water direction shall be selected in order to define the topology to be used.

5.3 Structure of an irrigation entity identifier (EntityID)

The ID of an irrigation entity shall be unique at this level and should be composed by the IDs of the upper level irrigation entities to which it belongs. The IDs shall respect the structure given in Table 42.

Table 42 — Structure of the EntityID

Name	Specification
Process cell ID	String formed by the value of the Table 34 corresponding to the entity at the process cell level of the physical model, followed by a three-figure number sequence.
Unit ID	String taken by the process cell ID containing the unit, followed by the code of the Table 34 corresponding to the entity, followed by a three-figure number sequence.
Equipment module ID	String taken by the process cell ID and the ID unit containing the equipment module, followed by the code of the Table 34 corresponding to the element at the equipment module level and by a three-figure number sequence. This ID admits recurrence with other equipment module IDs.

5.4 Properties

The properties contains, at least, the following information:

— value; and

— date on which that value was read (timestamp in coordinated universal time (UTC) ISO 8601 basic format YYYYMMDDhhmmss±hhmm).

Since the unit of measurement for the expression of all these properties is explicitly specified in this document, only the values are exchanged.

Table 43 can be extended with additional properties, according to the needs of the entities to be included in the physical model. Additional properties can be included in this document when their functional characterization are completed.

Table 43 — Properties specification and application

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
ActivityStatus	Status presented by an irrigation entity at the current moment.	N/A	Contains two fields separated by commas: value1, value2 Where: value1: indicates whether the entity is available to execute its functions. It may take one of the following listed values: — Active (introduced by user) — Inactive (introduced by user) value2: indicates the substatus of the entity. It may take one of the following listed values: — OutOfOrder (introduced by subsystem or user) — Unknown (introduced by subsystem) — Ready (introduced by subsystem or user) The following scenarios shall be taken into account: 1. If the entity receives the request to modify the ActivityStatus property to Inactive value when a procedural element is being executed, it stops, going to Stopped status. The corresponding event shall be generated and also the report of the executed irrigation. Once finished, the ActivityStatus property is set to Inactive. 2. If the entity receives a request for the execution of a new procedural element when it is in Inactive status, the response shall be in all cases a 1 - Execution error. 3. If a procedural element has been sent before inactivating the irrigation entity, its execution should be viable as long as its start condition is not reached and it is verified that the entity is still in the Inactive status. Once the start condition is reached (or the time end condition in a subsystem that supports partial executions), the procedural element shall pass to Stopped status. 4. If the procedural element previosly received contains repetitions or is based on calendar, its management shall be the same as described in point 3, but for each one of its instances. That is, several instances could pass to Stopped directly and later, others could be executed because the status of the irrigation entity has changed to Active before its start condition has been reached.	HYS, VIH, IHG, PMS, RSV, SSS, MIH, FTS and FTL.	INT, PML, NBU, SSB, NCP and FTL.
SystemStatus	Status presented	N/A	Contains seven fields, separated by commas, that identify the current state of the subsystem device	HYS, VIH, IHG, PMS,	INT, PML, NBU and

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
	by the control system controlling or monitoring the irrigation entity at the current moment.		controlling the irrigation entity: value1, value2, value3, value4, value5, value6, value7. Where: value1: number, with a decimal point, corresponding to the accumulator voltage expressed in volts (V). value2: percentage from 0 to 100, with a decimal point, referring to the remaining charge of the accumulator. This value is determined by the subsystem and is expressed as a percentage (%). If the subsystem does not perform this calculation, this field takes the value "Not supported". value3: number, with a decimal point, which corresponds to the register of the signal strength (Radio Signal Strength), expressed in decibels per milliwatt (dBm). If this data is not available, this field takes the value "Not supported". value4: associated to value3, listed value that corresponds to the expression of the signal strength, in terms interpretable by the user. The following range of possible values shall be used, being the relation with the dBm performed by subsystem: — Very good: excellent signal strength. — Good: good signal strength. — Fair: adequate signal strength. — Poor: poor signal strength. — Very poor: very poor signal strength. value5: percentage from 0 to 100, with a decimal point, corresponding to the percentage (%) of successful communications in relation to the total of registered communications. If this data is not available, this field takes the value "Not supported". Fields value3 and value4 can be complementary or alternative to value5. That is, every subsystem shall support values 3 and 4 or, alternatively, value 5. value6: UTC date recorded by the device clock, using ISO 8601 format YYYYMMDDhhmmss±hhmm. If no clock is available, this field takes the value "None". value7: UTC date, using ISO 8601 format YYYYMMDDDhhmmss±hhmm on which value6 has been recorded or, if it does not exist (value6 is "None"), the current date in the subsystem. If the subsystem is not working properly due power or communication events, values 1, 2, 3, 4, 5, 6 and 7 takes by default the value "Unknown".	RSV, SSS, MIH, FTS and FTL.	NCP.
Mode	Current functioning mode of the irrigation	N/A	Contains one field: value1 Where:	HYS, VIH, IHG, PMS, RSV, SSS, MIH, FTS	INT, NBU and NCP.

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
	entity.		value1: indicates the current functioning mode of the entity considered. It may take one of the following listed values: — OnDemand — Programmed — Monitoring (when no other type of procedural element are in running status) The value 1 are related to the type of procedural element in execution at the entity.	and FTL.	
CumulativeVolumeOut	Current value of volume registered by the irrigation entity in its outputs.	m^3	Contains one field: value1 Where: Value1: number with a decimal point containing the cumulative volume delivered since the date when the totalizer device was installed (MeterReplacementEvent static datum).	HYS, IRA, VIH, IHG, PMS, INT, SSS, NBU, MIH and FTS.	RSV, NCP and DSM.
InternalFlowOut	Current value of flow registered by the irrigation entity in its outputs.	m^3/s	Contains one field: value1 Where: value1: number, with a decimal point, that contains the flow delivered by the entity.	PMS and FTS.	HYS, IRA, VIH, IHG, INT, RSV, SSS, NBU, NCP, MIH and DSM.
PressureOut	Current value of pressure registered by the irrigation entity in its outputs.	Pa	Contains one field: value1 Where: value1: number, with a decimal point, that contains the output pressure of the specified entity.	PMS, FTS and WFS.	HYS, VIH, IHG, INT, SSS, NBU, NCP, MIH, DSM and FTL.
OpeningDegree	Property containing information referring to the position of a shut-off device.	%	Contains one field: value1 Where: value1: percentage, from 0 to 100, with a decimal point, referring to the position (from completely open -100 and completely closed – 0) of the shut-off element of the irrigation entity.	HYS, VIH, IHG and SSS.	INT, NBU and MIH.
Temperature	Air temperature registered at the entity.	°C	Contains one field: value1 Where: value1: air temperature.		HYS, IHG, PMS, RSV, SSS, NCP and MIH.
RelativeHumidity	Air relative humidity registered at the entity.	%	Contains one field: value1 Where: value1: relative humidity of the air.		HYS, IHG, PMS, RSV, SSS, NCP and MIH.

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
SolarRadiation	Solar radiation registered at the entity.	W/m^2	Contains one field: value1 Where: value1: solar radiation.		HYS, IHG, PMS, RSV, SSS, NCP and MIH.
WindParameters	Wind speed and direction registered at the entity.	m/s, deg	Contains two fields separated by commas: value1, value2 Where: value1: wind speed. Value2: wind direction.		HYS, IHG, PMS, RSV, SSS, NCP and MIH.
Rainfall	Rainfall registered at the entity.	$m^3/m^2/s$	Contains one field: value1 Where: value1: Cumulative volume of rain water.		HYS, IHG, PMS, RSV, SSS, NCP and MIH.
CumulativeVolumeIn	Current value of volume registered by the irrigation entity in its inputs.	m^3	Contains one field: value1 Where: value1: number with a decimal point containing the cumulative volume received since the date when the totalizer device was installed.	IRA, IHG, INT and FTS.	PMS, RSV, NBU, SSS and NCP.
InternalFlowIn	Current value of flow registered by the irrigation entity in its inputs.	m^3/s	Contains one field: value1 Where: value1: number, with a decimal point, that contains the flowreceived by the entity.	FTS	IRA, IHG, PMS, INT, RSV, NBU and NCP.
PressureIn	Current value of pressure registered by the irrigation entity in its inputs.	Pa	Contains one field: value1 Where: value1: number, with a decimal point, that contains the output pressure of the entity.	INT, FTS and WFS.	IHG, PMS, NBU, NCP, DSM and FTL.
InstantPower	Current amount of power absorbed by the entity during the execution of its functionalities.	W	Contains one field: value1 Where: value1: number, with a decimal point, that contains the absorbed instant electric power by the entity.	PMS	PML, FTS and WFS.
EnergyConsum	Entity	Ws,	Contains two fields:	PMS	FTS and

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
ption	energy consumption.	VArs	value1, value2 Where: value1: number, with a decimal point, corresponding to the active energy consumed by the entity since the date when the counting at the energy meter device was started (EnergyMeterCounterStart static datum). Value2: number, with a decimal point, corresponding to the reactive energy consumed by the entity since the date when the counting at the energy meter device was started (EnergyMeterCounterStart static datum).		WFS.
EntityPerformance	Entity hydraulic performance .	%	Contains one field: value1 Where: value1: percentage, from 0 to 100, with a decimal point corresponding to the entity performance during the execution of its functionalities. This property is obtained via the properties given in Formula (1): $E_{\mathrm{P}}(\%) = \left[\frac{9804{,}14 x F_{\mathrm{out}} x (P_{\mathrm{out}}/9804{,}14)}{I_{\mathrm{power}}}\right]$ (1) E_{p} = Entity Perform F_{out} = Internal flow out P_{out} = Pressure out I_{power} = Instant power	PMS	
PowerOnTime	Entity accumulated activity time.	s, N/A, s	Contains three fields: value1, value2, value3 Where: value1: number, with a decimal point, corresponding to the accumulated time since the date when the PowerOnTimeCounter was started. Value2: integer corresponding to the number of times that the entity has been executing its functionalities. Value3: number, with a decimal point, corresponding to the accumulated time on which the entity has been in ActivityStatus Unknown and/or OutofOrder.	PMS, FTS and WFS	PML, DSM and FTL.
Capacity	Volume contained by the entity.	m^3, %	Contains two fields: value1, value2 Where: value1: number with a decimal point, containing the current volume contained by the entity. Expressed in cubic metres. Value2: percentage, from 0 to 100 with a decimal point, corresponding to the quotient between the value1 and the DesignCapacity static datum.	RSV	NCP, PMS, FRT and DLT.

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
SoilWaterContent	Water contained by the soil linked to the entity in a reference area.	%, string	Contains from two to n fields, grouped in pairs. Each pair corresponds to a reference area existing in the entity: (value1, value2),(value n-1,value n) Where: value1: percentage, from 0 to 100 with a decimal point, corresponding to the water volume contained in the soil porosity. Value2: string with the geographical location of the value1 with its internal fields separed by commas. The content of value 2 uses the spatial data information defined on 6.2.1 for point geometry entities.		HYS, IHG, NCP, MIH and SSB.
SoilWaterPotential	Energy state of water in the soil linked to the entity in a reference area.	Pa, string	Contains from two to n fields, grouped in pairs. Each pair corresponds to a reference area existing in the entity: (value1, value2),(value n-1,value n) Where: value1: number with a decimal point, containing the current water retaining potential of the soil. Value2: string with the geographical location of the value1 with its internal fields separed by commas. The content of value 2 uses the spatial data information defined on 6.2.1 for point geometry entities.		HYS, IHG, NCP, MIH and SSB.
SoilTemperature	Temperature in the soil linked to the entity in a reference area.	°C, string	Contains from two to n fields, grouped in pairs. Each pair corresponds to a reference area existing in the entity: (value1, value2),(value n-1,value n) Where: value1: number with a decimal point, containing the current temperature of the soil. Value2: string with the geographical location of the value1 with its internal fields separed by commas. The content of value 2 uses the spatial data information defined on 6.2.1 for point geometry entities.		HYS, IHG, NCP, MIH and SSB.
SoilConductivity	Conductivity in the soil linked to the entity in a reference area.	S/m, string	Contains from two to n fields, grouped in pairs. Each pair corresponds to a reference area existing in the entity: (value1, value2),(value n-1,value n) Where: value1: number with a decimal point, containing the current conductivity of the soil. Value2: string with the geographical location of the value1 with its internal fields separed by commas. The content of value 2 uses the spatial data information defined on 6.2.1 for point geometry entities.		HYS, IHG, NCP, MIH and SSB.

PropertyName	Description	Unit (S.I.)	Specification	Required for	Extended for
WaterAcidity	Acidity value of the water.	N/A	Contains one field: value1 Where: value1: number, with a decimal point, containing the pH of the water.	FTS	
WaterConductivity	Electrical conductivity of the water.	S/m	Where: value1: number, with a decimal point, containing the current conductivity of the water. Expressed in S/m.	FTS	
DifferentialPressure	Pressure diference between the entity inlet and output.	Pa	Contains one field: value1 Where: Value1: number with a decimal point containing the pressure difference between the entity inlet and output. Expressed in pascals.	WFS	FTL

The properties in use vary with the irrigation entity modeled. It is required to take into account the properties defined as required or extended for each irrigation entity in the table. A number of these properties are common between entities. Additional entities may require properties which have not yet been defined.

Each subsystem controlling an irrigation entity shall support its required properties defined in Table 43. Additionally, for extended properties, and taking into account that can be implemented or not, the subsystem responds to one of the following situations.

— The property is supported and in use. The subsystem shall return its value as defined in Table 43.

— The property is supported by the subsystem but non-existent in the irrigation entity. The subsystem shall generate the message “Unavailable” when the value of such property is requested.

— The property is not supported by the subsystem. The subsystem shall generate the message “Not supported” when the value of such property is requested.

5.5 Entity events

The specification of events for all irrigation entities is presented in Table 44, where are only included the events related to the entity. Table 79 includes the events related to procedural elements.

In those events where EventName-Field2 is defined as Active/Inactive, the value Active should be understood as corresponding to the activation (or starting) of the event and the value Inactive as corresponding to the deactivation (or ending) of the event. Thus, to generate the event with Inactive value in field 2, it is required that previously has had Active value, always occurring the inactivation after the activation. Note that the listed events can be extended with new events and with the development of additional entities.

Table 44 — EventName

EventName-Field1	EventName-Field2	EventName-Field3	Description	Applies to
ActivityStatusChanged	Previous value	Current value	The ActivityStatus property changes its value.	All entities
SystemStatusUnknown			The SystemStatus property changes its value to unknown.	PMS, RSV, SSS, FTS, WFS, NCP, HYS, VIH, IHG and MIH
ModeChanged	Previous value	Current value	The Mode property changes its value.	All entities
InternalFlowOutIsNot0OpeningDegreeIs0	Active/Inactive		Activation/deactivation of flow other than zero with entity closed event.	HYS and VIH
GeneralFailure	Active/Inactive		Activation/deactivation of a general failure in an unit entity. The entity is not available when the event is active.	PMS, RSV, FTS and WFS
Failure	Active/Inactive		Failure in on status in an equipment module entity. The entity is not available when the event is active.	PML, NCP, NBU, DSM and FTL.
CumulativeVolumeOutSync	Previous value	Current value	The CumulativeVolumeOut property has been synchronized.	All entities
MinimumCapacity	Active/Inactive	Current value	Activation/deactivation of minimum level reached on a source entity. Related to Capacity property.	RSV, FRT and DLT
MaximumCapacity	Active/Inactive	Current value	Activation/deactivation of maximum level reached on a destination entity. Related to Capacity property.	RSV, FRT and DLT
CumulativeVolumeIn≠CumulativeVolumeOut			The CumulativeVolumeOut and CumulativeVolumeIn properties, has not the same value (±5 %).	INT, NBU and IHG
ContinuosCleaning	Active/Inactive		Activation/deactivation of a failure on in filter flushing.	WFS and FTL
MaxWorkingPressure	Active/Inactive		Activation/deactivation of maximum working pressure detected.	WFS
MinWorkingPressure	Active/Inactive		Activation/deactivation of minimum working pressure detected.	WFS
AutonomyWarning	Active/Inactive	Remaining autonomy (voltage)	Activation/deactivation of an event: the energy autonomy of the irrigation entity is near to end.	All entities
AccessWarning	Active/Inactive		Activation/deactivation of a warning of physical access to the irrigation entity.	All entities
SubsystemEvent	Available to define	Available to define	Open event for proprietary events of a subsystem.	All entities
AccumulativePropertySync	Previous value	Current value	The value of a accumulative property has been synchronized.	All entities

6 Interoperability III: Exchange of data from procedural elements performed in irrigation entities

6.1 General outline

The second part of exchangeable data is associated to the processes performed by the irrigation entities included in the physical model of this document. These processes are identified in the procedural model in this document. Among others, the data exchange allows to:

— generate procedural elements for an irrigation entity;

— execute actions over these procedural elements;

— monitor their statuses;

— generate events related; and

— obtain partial and final reports.

All the elements described in the procedural model contains the necessary information to characterize their status when requested. In certain status a procedural element may not accept the execution of actions over it.

6.2 Statuses and actions

This document defines five basic statuses for the procedural elements.

Table 45 — Procedural elements statuses

Status	Description
Idle	Initial status. Awaiting a start condition.
Running	The start condition has been reached; the programmed tasks are being performed.
Stopped	The procedural element has been detained by means of a request for stopping.
Completed	An end condition has been reached and it is finished.
Unknown	The status is not known at the moment of the request.

The actions available to be applied to the procedural elements are included in Table 46.

Table 46 — Actions available in each status

	Idle	Running	Stopped	Completed
StopRecipe	YES	YES	NO	NO

6.3 Types of Operation recipes

6.3.1 General

According to the types of irrigation entities, the Operation recipe types are defined in Table 47.

Table 47 — Operation recipes

Operation recipe type (RecipeType)	Description	Applicable to	Property Mode value
IrrigationRecipe1	Supply of water (or irrigation) by time or volume.	HYS, IHG/VIH and SSS	Programmed
IrrigationRecipe2	Supply of water (or irrigation) without end condition.	HYS	OnDemand
IrrigationRecipe3	Supply of water (or irrigation) based on a calendar and executions by time or volume.	HYS, IHG/VIH and SSS	Programmed
IrrigationRecipe4	Supply of water (or irrigation) based on thresholds.	HYS and SSS	Programmed
PumpingRecipe1	Pumping of irrigation water according to a flow rate set point.	PMS	Programmed
PumpingRecipe2	Pumping of irrigation water according to flow rate and level set points, typically related to RSV.	PMS	Programmed
PumpingRecipe3	Pumping of irrigation water according to a pressure set point, typically required by a INT.	PMS	OnDemand
PumpingRecipe4	Pumping of irrigation wate according to flow rate and pressure set points, typically required by a INT.	PMS	OnDemand
PumpingRecipe5	Pumping of irrigation water attending to time habilitation.	PMS	Programmed
NetworkBranchRecipe1	Procedure of water delivering to one or more recipients, guaranteeing the presure and flow requirements.	NBU	N/A
ControlPointRecipe1	Programming of a control point for safety and maintenance purposes.	NCP	N/A
ControlPointRecipe2	Programming of a control point to provide specific flow rate or pressure downstream the irrigation entity.	NCP	Programmed
ControlPointRecipe3	Programming of a control point to monitorize one or more properties of the irrigation entity.	NCP	Monitoring
FertigationRecipe1	Addition of fertilizers to the irrigation water by injection of a proportional quantity of fertilizer to a volume of irrigation water.	FTS	Programmed
FertigationRecipe2	Addition of fertilizers to the irrigation water by injection of a proportional quantity of fertilizer throughout the Operation.	FTS	Programmed
FertigationRecipe3	Addition of fertilizers to the irrigation water by injection of a bulk of fertilizer based on a time criteria during the Operation.	FTS	Programmed
FertigationRecipe4	Addition of fertilizers to the irrigation water by injection of a bulk of fertilizer based on a volume criteria during the Operation.	FTS	Programmed
FertigationRecipe5	Addition of a premixed fertilizer to the irrigation water.	FTS	Programmed

Operation recipe type (RecipeType)	Description	Applicable to	Property Mode value
FiltrationRecipe1	Physical treatment of irrigation water to reduce the amount of solid particles present.	WSF	Programmed

Note that the Operation recipes described in Table 47 can be extended to response to the needs of the irrigation entities to be included in this document.

6.3.2 Parameterization of header

All the parameters described in the header of an Operation recipe are required.

Table 48 — Generic parameter of the header of an Operation recipe

Header		
Name of parameter	**Type of parameter**	**Description**
ProceduralID	string	Unique ID of a procedural element.
EntityID	string	Unique ID of the entity for which the Operation is intended.
RecipeType	RecipeTtype	Recipe type, as described in Table 47.

6.3.3 Formula parameterization (common parameters) for irrigation Operations

Table 54 presents the formula parameters that are common to all irrigation Operations.

Table 49 — Generic parameters for irrigation Operations

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0.
		NewRecipe	queue	Another queuing Operation reaches its start condition (StartDate).
Optional parameters				
	OpeningDegree		integer	Opening degree of the entity, expressed as a percentage. Dead zone ±5. Default value 100.
Optional conditioning parameters				
	MonCumulativeVolumeOut		boolean	Activate/deactivate the monitoring of the output volume. Default value 0.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
	MonInternalFlowOut		boolean	Activate/deactivate the monitoring of the output flow. Default value 0.
	MonPressureOut		boolean	Activate/deactivate the monitoring of the output pressure. Default value 0.
Required parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the ThresholdHH is exceeded, during the configured time, an event is generated.
Optional parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut				
	CmdHH		boolean	Its activation forces the Operation to stop its execution when ThresholdHH is reached. Parameter conditioned by ThresholdHH.Default value 0.
Required parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the ThresholdHH is exceeded, during the configured time, an event is generated. Required if ThresholdLL=0.
	ThresholdLL		decimal	Very low threshold value of the property for which monitoring is activated. When the ThresholdLL is exceeded, during the configured time, an event is generated. Required if ThresholdHH=0.
Optional parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut				
	TimeHH		integer	Time in seconds where the threshold HH shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 seconds.
	CmdHH		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdHH. Default value 0.
	TimeLL		integer	Time in seconds where the threshold LL shall be exceeded. Parameter conditioned by ThresholdLL. Default value 30 s.
	CmdLL		boolean	Its activation forces the Operation to stop when ThresholdLL is reached during the configured time. Parameter conditioned by ThresholdLL. Default value 0.

6.3.4 Additional parameters of an IrrigationRecipe1

The following specific parameters shall be added to the generic parameters to design an Operation recipe of type IrrigationRecipe1. When the recipe is running, the Mode property of the irrigation entity changes to Programmed.

Table 50 — Specific parameters of IrrigationRecipe1

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. Expressed in seconds. No default value.
Required parameters for SSS and IHG				
	TimeSequence		string	String composed of one or more groups of two fields separated by commas. Each group corresponds to the parameterization of a SSB or a VIH. The string can contain as many groups as this kind of entities has the entity. Each group has: Value1: EntityID of the SSB or VIH where the value 2 is going to be executed. Value2: quantity of time to irrigate, expressed in seconds.
Optional parameters				
		MaxVolume	decimal	Maximum volume during the Operation. With a decimal point and expressed in m^3. No default value.
Optional parameters for SSS and IHG				
	VolumeSequence		string	String composed of one or more groups of two fields separated by commas. Each group corresponds to the parameterization of a SSb or a VIH. The string can contain as many groups as this kind of entities has the entity. Each group has: Value1: EntityID of the SSB or VIH where the value 2 is going to be executed. Value2: quantity of water to irrigate, expressed in m^3.
Optional conditioning parameters				
Repetition			boolean	Activate the Operation repetition. Default value 0 (deactivate).
Required parameters conditioned by activation of Repetition				
	RepetitionInterval		integer	Time interval between repetitions, expressed in seconds and added to the StartDate parameter of the last execution.
	RepetitionNumber		integer	Number of times that the RepetitionInterval parameter shall be repeated. Without limitation.

If the repetition of the operation is set up, any of the resulting repetitions shall be considered an instance of it. The subsystem in charge shall assign an internal identification for each instance of the operation.

6.3.5 Additional parameters of a IrrigationRecipe3

The following specific parameters shall be added to the generic parameters to design an Operation recipe of type IrrigationRecipe3. When the recipe is running, the Mode property of the irrigation entity shall be changed to Programmed.

Table 51 — Specific parameters of IrrigationRecipe3

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. Expressed in seconds. No default value.
	WeekCalendar		string	Contains 7 fields separated by commas, representing a week (first field is for Monday and the last for Sunday according to ISO 8601). Each field can take one of the next values: — Value=1, when the irrigation recipe should be executed in that day. — Value=0, irrigation recipe should not be executed in that day. Without default values.
		FinalDate	string	Date/time consigned for the end of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
Optional parameters				
		MaxVolume	integer	Maximum volume during the Operation. With a decimal point and expressed in m^3. No default value.

Any of the resulting repetitions result of the operation setup shall be considered an instance of it. The subsystem in charge assigns an internal identification for each instance of the operation.

6.3.6 Additional parameters of a IrrigationRecipe4

The design of an Operation type IrrigationRecipe4 requires specific parameters in addition to the abovementioned generic parameters. An IrrigationRecipe4 shall contain, at least, one of the optional parameters listed in Table 52. When the recipe is running, the Mode property of the irrigation entity shall be changed to Programmed.

Table 52 — Specific parameters of IrrigationRecipe4

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. Expressed in seconds. No default value.
Optional parameters				
		MaxRainfall	string	Contains 2 fields separated by commas: — Field 1. Maximum rainfall, with a decimal point, that stops the Operation during its execution. Expressed in $m^3/m^2/s$. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
		MaxWindSpeed	string	Contains 2 fields separated by commas: — Field 1. Maximum wind speed, with a decimal point, obtained from the WindParameters property that stops the Operation during its execution. Expressed in m/s. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MinSoilWaterContent			string	Contains 2 fields separated by commas: — Field 1. Minimum value of the WaterSoilContent property, with a decimal point, that starts the Operation. Expressed in %. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
		MaxSoilWaterContent	string	Contains 2 fields separated by commas: — Field 1. Maximum value of the WaterSoilContent, with a decimal point, that stops the Operation during its execution. Expressed in %. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MinSoilWaterPotential			string	Contains 2 fields separated by commas: — Field 1. Minimum value of the WaterSoilPotential property, with a decimal point, that starts the Operation. Expressed in Pa. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
		MaxSoilWaterPotential	string	Contains 2 fields separated by commas: — Field 1. Maximum value of the WaterSoilPotential, with a decimal point, that stops the Operation during its execution. Expressed in Pa. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
SolarRadiationTriggerOn			string	Contains 2 fields separated by commas: — Field 1. Value of the SolarRadiation property, with a decimal point, that starts the Operation. Expressed in W/m^2. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
		SolarRadiationTriggerOff	string	Contains 2 fields separated by commas: — Field 1. Value of the SolarRadiation property, with a decimal point, that stops the Operation during its execution. Expressed in W/m^2. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.

6.3.7 Formula parameterization for NetworkBranchRecipe1

This Operation recipe cannot be set up by the user to avoid its omission.

The subsystem responsible should establish as Initial condition the first start date (StartDate) among the Operations for its contained entities. The same criterion shall be used to define the Operation final condition, using the last final date among the Operations to execute by its contained entities.

An additional check is required for a NetworkBranchRecipe1. The subsystem analyses the maximum simultaneity during all the NBU Operation and blocks the execution when overcomes the one fixed at 5.2.

Table 53 — Generic parameters of the recipe formula for NetworkBranchRecipe1

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	The initial condition of the first Operation of the contained entities is reached. Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0. The subsystem stops all the contained Operations in Idle or Running status.
		MaxDuration	integer	The end time condition of the last Operation of the contained entities is reached.
		NewRecipe	queue	Another queuing Operation reaches its start condition (StartDate).
Optional conditioning parameters				
	MonCumulativeVolumeOut		boolean	Activate/deactivate the monitoring of the output volume. Default value 0.
	MonInternalFlowOut		boolean	Activate/deactivate the monitoring of the output flow. Default value 0.
	MonPressureOut		boolean	Activate/deactivate the monitoring of the output pressure. Default value 0.
	MonCumulativeVolumeIn		boolean	Activate/deactivate the monitoring of the input volume. Default value 0.
	MonInternalFlowIn		boolean	Activate/deactivate the monitoring of the input flow. Default value 0.
	MonPressureIn		boolean	Activate/deactivate the monitoring of the input pressure. Default value 0.
Required parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut and CumulativeVolumeIn.				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the ThresholdHH is exceeded, during the configured time, an event is generated.
Optional parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut and CumulativeVolumeIn.				
	CmdHH		boolean	Its activation forces the Operation to stop its execution when ThresholdHH is reached. Default value 0.
Required parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut, InternalFlowIn and PressureIn				

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the ThresholdHH is exceeded, during the configured time, an event is generated. Required if ThresholdLL=0.
	ThresholdLL		decimal	Very low threshold value of the property for which monitoring is activated. When the ThresholdLL is exceeded, during the configured time, an event is generated. Required if ThresholdHH=0.
Optional parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut, InternalFlowIn and PressureIn.				
	TimeHH		integer	Time in seconds where the ThresholdHH shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
	CmdHH		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdHH. Default value 0.
	TimeLL		integer	Time in seconds where the ThresholdLL shall be exceeded. Parameter conditioned by ThresholdLL. Default value 30 s.
	CmdLL		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdLL. Default value 0.

6.3.8 Formula parameterization (common parameters) for pumping Operations

Table 54 — Generic parameters for pumping Operations

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0.
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. Expressed in seconds. No default value.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
		NewRecipe	queue	Another queuing Operation reaches its start condition (StartDate).
Optional parameters				
		MaxVolume	integer	Maximum volume during the Operation. With a decimal point and expressed in m^3. No default value.
Optional conditioning parameters				
Repetition			boolean	Activate/deactivate the recipe repetition. Default value 0.
	WeekCalendar		string	Contains 7 fields, representing a week (first field is for Monday and the last for Sunday according to ISO 8601). Each field can take one of the next values: — Value=1, when the irrigation recipe should be executed in that day. — Value=0, irrigation recipe should not be executed in that day. Without default values.
	MonCumulativeVolumeOut		boolean	Activate/deactivate the monitoring of the output volume. Default value 0.
	MonInternalFlowOut		boolean	Activate/deactivate the monitoring of the output flow. Default value 0.
	MonPressureOut		boolean	Activate/deactivate the monitoring of the output pressure. Default value 0.
Required parameters conditioned by activation of Repetition				
	RepetitionInterval		integer	Time interval between repetitions, expressed in seconds and added to the StartDate of the last execution.
	RepetitionNumber		integer	Number of times that the RepetitionInterval parameter shall be repeated. Without limitation.
Required parameters conditioned by activation of WeekCalendar				
		FinalDate	string	Date/time consigned for the end of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
Required parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the ThresholdHH is exceeded, during the configured time, an event is generated.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Optional parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut				
	CmdHH		boolean	Its activation forces the Operation to stop its execution when ThresholdHH is reached. Default value 0.
Required parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the ThresholdHH is exceeded, during the configured time, an event is generated. Required if ThresholdLL=0.
	ThresholdLL		decimal	Very low threshold value of the property for which monitoring is activated. When the ThresholdLL is exceeded, during the configured time, an event is generated. Required if ThresholdHH=0.
Optional parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut				
	TimeHH		integer	Time in seconds where the ThresholdHH shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
	CmdHH		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdHH. Default value 0.
	TimeLL		integer	Time in seconds where the ThresholdLL shall be exceeded. Parameter conditioned by ThresholdLL. Default value 30 s.
	CmdLL		boolean	Its activation forces the Operation to when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdLL. Default value 0.

6.3.9 Additional parameters of a PumpingRecipe1

Table 55 — Specific parameters for PumpingRecipe1

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	InternalFlowOutSetPoint		decimal	Flow rate setpoint, with a decimal point, to maintain during the Operation execution. With a decimal point and expresed in m^3/s.

6.3.10 Additional parameters of a PumpingRecipe2

Table 56 — Specific parameters for PumpingRecipe2

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	SourceLevel_InternalFlowOutSetPoint		string	LevelFlowRate setpoints to accomplish during the Operation execution. The parameter contains one or more Level-FlowRate relations where: — Value1: Is the minimum level required at source for a certain Value2, expressed in metres. — Value2: Is the flow rate setpoint to follow for corresponding the Value1, expressed in m^3/s. The Value1 shall be measured to define which Value2 is used as setpoint. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.
	DestinationLevel_InternalFlowOutSetPoint		string	Level-FlowRate setpoints to accomplish during the Operation execution. The parameter contains one or more Level-FlowRate relations where: — Value1: Is the minimum level required at destination for a certain Value2, expressed in metres. — Value2: Is the flow rate setpoint to follow for the corresponding Value1, expressed in m^3/s. The Value1 shall be measured to define which Value2 is used as setpoint. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.

This Operation type not allows the simultaneous use of SourceLevel-FlowRateSetPoint and DestinationLevel-FlowRateSetPoint.

6.3.11 Additional parameters of a PumpingRecipe3

Table 57 — Specific parameters for PumpingRecipe3

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				

	PressureOutSetPoint		decimal	Pressure setpoint, with a decimal point, to maintain during the Operation execution. With a decimal point and expresed in Pa.

6.3.12 Additional parameters of a PumpingRecipe4

Table 58 — Specific parameters for PumpingRecipe4

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	InternalFlowOutSetPoint_ PressureOutSetPoint		string	FlowRate-Pressure setpoints to accomplish during the Operation execution. The parameter contains one or more FlowRate-Pressure relations where: — Value1: Is the flow rate required for a certain Value2, expressed in m^3/s. — Value2: Is the pressure setpoint to follow for the corresponding Value1, expressed in Pa. The value 1 shall be measured to define which Value2 is used as setpoint. Each pair of values shall be separated by commas. There is no limitation to the number of pairs included in the parameter.

6.3.13 Additional parameters of a PumpingRecipe5

The PumpingRecipe5 uses only the common parameters for pumping Operations.

6.3.14 Formula parameterization (common parameters) for control point Operations

Table 59 — Generic parameters for control point Operations

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. Expressed in seconds. No default value.
		NewRecipe	queue	Another queuing Operation reaches its start condition (StartDate).
Optional conditioning parameters				
	MonCumulativeVolumeOut		boolean	Activate/deactivate the monitoring of the output volume. Default value 0.
	MonInternalFlowOut		boolean	Activate/deactivate the monitoring of the output flow. Default value 0.
	MonPressureOut		boolean	Activate/deactivate the monitoring of the output pressure. Default value 0.
Required parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the threshold HH is exceeded, during the configured time, an event is generated.
Optional parameters conditioned by activation of the monitoring of a cumulative property: CumulativeVolumeOut				
	CmdHH		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Default value 0.
Required parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut				
	ThresholdHH		decimal	Very high threshold value of the property for which monitoring is activated. When the threshold HH is exceeded, during the configured time, an event is generated. Required if ThresholdLL=0.
	ThresholdLL		decimal	Very low threshold value of the property for which monitoring is activated. When the threshold LL is exceeded during the configured time, an event is generated. Required if ThresholdHH=0.
Optional parameters conditioned by activation of the monitoring of a property: InternalFlowOut, PressureOut				
	TimeHH		integer	Time in seconds where the threshold HH shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
	CmdHH		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdHH. Default value 0.
	TimeLL		integer	Time in seconds where the threshold LL shall be exceeded. Parameter conditioned by ThresholdLL. Default value 30 s.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
	CmdLL		boolean	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdLL. Default value 0.

6.3.15 Formula parameterization for ControlPointRecipe1

The ControlPoint1 uses only the common parameters for control point Operations.

6.3.16 Formula parameterization for ControlPointRecipe2

Table 60 — Specific parameters for ControlPointRecipe2

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Optional parameters				
	InternalFlowOutSetPoint		decimal	Flow setpoint, with a decimal point, to maintain during the Operation execution. With a decimal point and expresed in m^3/s.
	PressureOutSetPoint		decimal	Pressure setpoint, with a decimal point, to maintain during the Operation execution. With a decimal point and expresed in Pa.

6.3.17 Formula parameterization for ControlPointRecipe3

Table 61 — Specific parameters for ControlPointRecipe3

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Optional parameters				
	MaxLevel		decimal	Level setpoint which determines that the entity is at full capacity for the closing of its water inputs. With a decimal point and expresed in m.
	MinLevel		decimal	Level setpoint which determines that the entity is at empty capacity for the closing of its water outputs. With a decimal point and expresed in m.

6.3.18 Formula parameterization (common parameters) for fertigation Operations

Table 62 — Generic parameters for fertigation Operations

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYYYYYMMDDhhmmssZ. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0.
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. Expressed in seconds. No default value.
		NewRecipe	queue	Another queuing recipe reaches its start condition (StartDate).
Optional parameters				
		MaxVolume	decimal	Maximum volume during the Operation. With a decimal point and expressed in m^3. No default value.

6.3.19 Additional parameters of a FertigationRecipe1

Table 63 — Parameters of the recipe formula for FertigationRecipe1

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	QuantityRatioSetPoint		string	String composed of one or more groups of four fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. The fields 3 and 4 of each group are mutually exclusive: may both be empty but not filled in simultaneously. Each group has: Value1: DSM number, expressed as an integer. Value2: quantity of fertilizer, with a decimal point and expressed in m^3, to be proportionally dosified per volume unit of irrigation water and directly related to the total volume to be fertilized (MaxVolume). Value3: pH setpoint for dosifying adjust. Value4: electrical conductivity for dosifying adjust, with a decimal point and expressed in S/m.

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
				The value2 can be adapted attending to value3 or value4 setpoint.
Optional parameters				
	pHSetPoint		real	Acidity value, with a decimal point, to maintain during the Operation execution.
	ECSetPoint		real	Electrical conductivity value, with a decimal point, to maintain during the Operation execution, in S/m.

6.3.20 Additional parameters of a FertigationRecipe2

Table 64 — Parameters of the recipe formula for FertigationRecipe2

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	TimeRatioSetPoint		string	String composed of one or more groups of four fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. The fields 3 and 4 of each group are mutually exclusive: may both be empty but not filled in simultaneously. Each group has: Value1: DSM number, expressed as an integer. Value2: quantity of time, expressed in seconds, during which the fertilizer is going to be proportionally dosified until the Operation reaches its end by time (MaxDuration). Value3: pH setpoint for dosifying adjust, with a decimal point. Value4: electrical conductivity for dosifying adjust, with a decimal point and expressed in S/m. The value2 can be adapted attending to value3 or value4 setpoint.
Optional parameters				
	pHSetPoint		real	Acidity value, with a decimal point, to maintain during the Operation execution.
	ECSetPoint		real	Electrical conductivity value, with a decimal point, to maintain during the Operation execution, in S/m.

6.3.21 Additional parameters of a FertigationRecipe3

Table 65 — Parameters of the recipe formula for FertigationRecipe3

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	FertigationTime		real	String composed of one or more groups of two fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. Each group has: Value1: DSM number, expressed as an integer. Value2: Time to apply the fertilizer bulk by the DSM, expressed in s.

6.3.22 Additional parameters of a FertigationRecipe4

Table 66 — Parameters of the recipe formula for FertigationRecipe4

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
	FertigationVolume		real	String composed of one or more groups of two fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. Each group has: Value1: DSM number, expressed as an integer. Value2: Volume of fertilizer to apply as fertigation bulk by the DSM, with decimal point and expressed in m^3.

6.3.23 Additional parameters of a FertigationRecipe5

The FertigationRecipe5 uses only the common parameters for fertigation Operations.

6.3.24 Formula parameterization for FiltrationRecipe1

Table 67 — Parameters of the recipe formula for FiltrationRecipe1

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmssZ. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0.
		MaxDuration	integer	Maximum time of duration of the Operation, since the start condition has been reached. With decimal point and expressed in seconds. No default value.
	MaxTime		integer	Maximum time between filter flushing, in s.
		NewRecipe	queue	Another queuing recipe reaches its start condition (StartDate).
Optional parameters				
	DifferencialPressure		real	Pressure diference, measured between filter inlet and outlet, that activates the filter flushing. Expressed in Pa. No default value.

6.4 Types of Unit Procedure recipes

6.4.1 General

According to the types of irrigation entities described, the Unit Procedure recipe types are defined as indicated in Table 68.

Table 68 — Unit Procedure recipes

Unit Procedure recipe type	Description	Applicable to	Entity Mode when executing
IrrigationNetworkUnitProcedure1	Procedure of water delivering to one or more recipients, guaranteeing the presure and flow requirements.	INT	N/A
SolidSetUnitProcedure1	Procedure for water delivering to one or more irrigation blocks, guaranteeing the presure and flow requirements.	SSS	Programmed
PumpingUnitProcedure1	Procedure for pumping irrigation water to provide flow with the right pressure conditions.	PMS	Programmed
ReservoirUnitProcedure1	Procedure for monitoring the storagement of irrigation water.	RSV	N/A

Unit Procedure recipe type	Description	Applicable to	Entity Mode when executing
FertigationUnitProcedure1	Procedure for execute a set of fertigations.	FTS	Programmed
FiltrationUnitProcedure1	Procedure for irrigation water filtration.	WFS	Programmed

Note that the unit procedure recipes described in Table 68 can be extended following the complete description of new irrigation entities to be included in this document.

6.4.2 Parameterization of header

All the parameters described in Table 69 are required.

Table 69 — Generic parameter of the header of a Unit Procedure recipe

Header		
Name of parameter	**Type of parameter**	**Description**
ProceduralID	string	Unique ID of a procedural element.
EntityID	string	Unique ID of the entity for which the operation is intended.
RecipeType	RecipeType	Recipe type, as described in Table 68.

6.4.3 Formula parameterization for Unit Procedures

This procedural elements recipe cannot be set up by the user to avoid its omission and shall be used in all unit procedures.

The responsible for procedural elements generation, should establish as initial condition the first start date (StartDate) among the Operations that it contains. The same criterion shall be used to define its end condition, using the last final date among the Operations to execute by its contained entities.

The subsystem analyses the maximum simultaneity during all the Unit Procedure execution, monitoring all Operations to be executed in the irrigation entities composing the recipient entity (INT or SSS). The subsystem blocks the creation of the Unit Procedure, or the new Operations adding, when during its execution the simultaneity overcomes the one fixed at 6.1 in order to guarantee the right pressure and flow requirements.

Table 70 — Parameters of the recipe formula for Unit Procedures

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	The initial condition of the first Operation of the contained entities is reached. Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.

		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0. The subsystem stops all the contained Operations in Idle or Running status.
		MaxDuration	integer	The final condition of the last Operation of the contained entities is reached.
		NewRecipe	queue	Another queuing Unit Procedure reaches its start condition (StartDate).

6.5 Types of Procedure recipes

The Procedure recipe types are defined in the Table 71.

Table 71 — Procedure recipes

Procedure recipe type	Description	Applicable to	Entity Mode when executing
IrrigationSectorProcedure1	General control of the main process to be performed (to irrigate), based on the activities developed in the irrigation entities (and its dependences) composing the irrigation sector.	IRA	N/A

Note that the Procedure recipes described in Table 71 can be extended following the complete description of new irrigation entities to be included in this document.

6.5.1 Parameterization of header

All the parameters described in the header of a Procedure recipe are required.

Table 72 — Generic parameter of the header of a Unit Procedure recipe

Header		
Name of parameter	**Type of parameter**	**Description**
ProceduralID	string	Unique ID of a procedural element.
EntityID	string	Unique ID of the entity for which the operation is intended.
RecipeType	RecipeType	Recipe type, as described in Table 71.

6.5.2 Formula parameterization for Procedures

This procedural elements recipe cannot be set up by the user to avoid its omission and shall be used in all procedures.

The responsible for procedural elements generation, should establish as Initial condition the first start date (StartDate) among the Unit Procedures that it contains. The same criterion shall be used to define its end condition, using the last final date among the Unit Procedures to execute by its contained entities.

The subsystem analyses the dependencies among the irrigation entities to assure that all required ones are involved in the Procedure, in order to guarantee the right operational conditions for all of them, along its execution.

Table 73 — Parameters of the recipe formula for unit procedures

Formula (RecipeType)				
Initial condition	**Running parameters**	**Final condition**	**Type**	**Description**
Required parameters				
StartDate			string	The initial condition of the first Operation of the contained entities is reached. Date/time consigned for the start of the Operation. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
		StopRecipe	boolean	StopRecipe action is sent to the subsystem. Default value 0. The subsystem stops all the contained Operations in Idle or Running status.
		MaxDuration	integer	The final condition of the last Operation of the contained entities is reached.
		NewRecipe	queue	Another queuing Procedure reaches its start condition (StartDate).

6.6 Report

The report of a procedure, unit procedure or operation should be generated at the control level that runs it. The reports of the procedures and unit procedures are generated by data aggregation from the reports of the operations contained. The basic unit of information is, therefore, the operation report.

If the report is from a repetition (instance) of an operation, all the information included is from it, including the status.

Table 74 — Report of a Procedure, Unit Procedure or Operation

Report of a procedural element		
Field	**Type**	**Description**
ProceduralID	string	Unique ID of the procedural element to be reported.
InstanceNumber	integer	Number of the instance to which the report belongs. For operations without repetitions, this field is 0. The value 0 is also used to identify the report of the original operation. For any other instances this field can use values from 1 to n.
RecipeType	string	Type of recipe associated to the ProceduralID.
InitialDate	string	Date/time when the initial condition was reached. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
FinalDate	string	Date/time when the final condition was reached. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
EndCondition	string	End condition reached during the execution of the procedural element.
Duration	integer	Expression, in seconds, of the time between the initial and final condition.
Volume	integer	Volume registered between the initial and final condition, in the property CumulativeVolumeOut. Expressed in m^3.
OpeningDegree	integer	Average OpeningDegree registered in the irrigation entity during the execution of the procedural element. Expressed as a percentage.

Report of a procedural element		
Field	**Type**	**Description**
MaxFlowOut	decimal	Maximum output flow registered in the irrigation entity during the execution of the procedural element. Expressed in m^3/s.
MaxFlowOutTimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MaxFlowOut was captured.
MinFlowOut	decimal	Minimum output flow registered in the irrigation entity during the execution of the procedural element. Requires timestamp. Expressed in m^3/s.
MinFlowOutTimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MinFlowOut was captured.
AverageFlowOut	decimal	Average input flow registered in the irrigation entity during the execution of the procedural element. Expressed in m^3/s.
MaxPressureOut	decimal	Maximum pressure reachedin the irrigation entity during the execution of the procedural element. Expressed in Pa.
MaxPressureOutTimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MaxPressureIn was captured.
MinPressureOut	decimal	Minimum pressure reached in the irrigation entity during the execution of the procedural element. Expressed in Pa.
MinPressureOutTimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MinPressureIn was captured.
EventIDs	string	List of events triggered during the execution of the procedural element.

All the information included in the report, and referring to a non-existent or not supported extended property, shall have null value.

6.6.1 Additional report parameters of Pumping recipes

Table 75 — Additional parameters in reports for PMS Operations

Report of a procedural element		
Field	**Type**	**Description**
EnergyConsumption	string	Acumulated value, with a decimal point, of the EnergyConsumption property at the end of the procedural element execution, being: — Value1: active energy consumed. Expressed in Ws. — Value2: reactive energy consumed. Expressed in Vars.
MaxInstantPower	decimal	Maximum value of the InstantPower property registered in the irrigation entity during the execution of the procedural element. Expressed in W.
MaxInstantPowerTimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MaxInstantPower was captured.
AvgEntityPerformance	integer	Average value of the EntityPerformance property registered in the irrigation entity during the execution of the procedural element. Expressed in percentage (%).

Report of a procedural element		
Field	**Type**	**Description**
MaxEntityPerformance	integer	Maximum value of the EntityPerformance property registered in the irrigation entity during the execution of the procedural element. Expressed in percentage (%).
MaxEntityPerformanceTimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MaxEntityPerformance was captured.
MinEntityPerformance	integer	Minimum value of the EntityPerformance property registered in the irrigation entity during the execution of the procedural element. Expressed in percentage (%).
MinEntityPerformance TimeStamp	string	Date/time in coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm when the value MinEntityPerformance was captured.
EndSourceCapacity	decimal	If the water is supplied by a RSV, the value of its Capacity property at the end of the procedural element execution.
StartSourceCapacity	decimal	If the water is supplied by a RSV, the value of its Capacity property at the start of the procedural element execution.
EndDestinationCapacity	decimal	If the water is supplied to a RSV, the value of its Capacity property at the end of the procedural element execution.
StartDestinationCapacity	decimal	If the water is supplied to a RSV, the value of its Capacity property at the start of the procedural element execution.

6.6.2 Additional report parameters of Reservoir recipes

Table 76 — Additional parameters in reports for RSV Operations

Report of a procedural element		
Field	**Type**	**Description**
InitialCapacity	string	Accumulated water volume, with decimal point, registered in the irrigation entity when the procedural element execution stars, being: — Value1: accumulated volume expressed in m^3. — Value2: accumulated volume expressed in %.
FinalCapacity	string	Accumulated water volume, with decimal point, registered in the irrigation entity when the procedural element execution ends, being: — Value1: accumulated volume expressed in m^3. — Value2: accumulated volume expressed in %.

6.6.3 Additional report parameters of Fertigation recipes

Table 77 — Additional parameters in reports for FTS Operations

Report of a procedural element		
Field	**Type**	**Description**
AveragepH	real	Average acidity registered in the entity during the execution of the procedural element.
AverageEC	real	Average electrical conductivity registered in the entity during the execution of the procedural element.

Report of a procedural element		
Field	**Type**	**Description**
FertilizerComsumption	string	Contains, separated by commas, as many fertilizer consumptionw as DSM has the entity.

6.6.4 Additional report parameters of Filtration recipes

Table 78 — Additional parameters in reports for WFS Operations

Report of a procedural element		
Field	**Type**	**Description**
FlushingsByTime	integer	Acumulated value of flushings activated by the time condition.
FlushingsByPressure	integer	Acumulated value of flushings activated by the differential pressure condition.

6.7 Procedural element events

The events corresponding to the procedural elements are given in Table 79. Events in this table are to be combined with those listed in Table 44. The listed events can be extended with new procedural elements when described.

Table 79 — EventName

EventName (Field1)	EventName (Field2)	EventName (Field3)	Description	Applies to
StartedOperation			A procedural element has changed to the status Running.	All entities
StoppedOperation			A running or idle procedural element reaches the Stopped status (following the StopRecipe action).	All entities
CompletedOperation			A running procedural element reaches the Completed status.	All entities
OperationErrorStarting	Active/Inactive		A procedural element was not correctly executed. Althoug the start condition was reached, the Operation did not start.	All entities
OperationErrorEnding	Active/Inactive		A procedural element was not executed correctly. When the end condition was reached, the Operation had not finished.	All entities
InternalFlowOutIs0OpeningDegreeIsNot0	Active/Inactive		Alarm of zero flow with irrigation entity open. To generate the event with Inactive value in field 2, is required that previously the field 2 of the event has been in Active.	HYS and VIH
ThresholdHHTriggeredOn	PropertyName	Current value	The ThresholdHH of a monitoring property has been reached.	All entities

EventName (Field1)	EventName (Field2)	EventName (Field3)	Description	Applies to
ThresholdHHTriggeredOff	PropertyName	Current value	The ThresholdHH of a monitoring property has been deactivated.	All entities
ThresholdLLTriggeredOn	PropertyName	Current value	The ThresholdLL of a monitoring property has been reached.	All entities
ThresholdLLTriggeredOff	PropertyName	Current value	The ThresholdLL of a monitoring property has been deactivated.	All entities
OperationErrorSetPoint			A procedural element was not correctly executed. The subsystem can not reach or maintain the setpoint introduced.	PMS and FTS
OperationOnRecipient			A procedural element was received by the destination device.	All entities

Annex A
(normative)

Management interface with SOAP 1.2

A.1 Overview

This annex outlines the information needed to implement the management interface through Web Services and SOAP technology. This implementation permits MIS applications to retrieve data from the upper control level: coordination. Information can be generated at this control level or be requested to the lower control level (subsystems). MIS applications only need to implement the methods required to perform their management tasks.

This annex concerns MIS and coordination brokers adapted to this document. The interface between these architecture elements is addressed by a Web Services:

— Upper control level: the coordination broker publishes a Web Service as a server; and

— Any application located at management level acts as a Web Services client.

This annex defines a set of methods available for any MIS application by its publication by the upper control level, where the coordination brokers are located.

A.2 Requirements

A.2.1 General

The names of the classes and the methods presented in this annex shall by used for the implementation of the management interface when using SOAP1.2. Likewise, shall be used the following protocols defined by the Web Services Protocol Stack:

— HTTP 1.1 (Hypertext Transfer Protocol) RFC 2616;

— TLS Within HTTP/1.1 RFC 2817;

— TLS 1.2 RFC 5246;

— SOAP 1.2. (Simple Object Access Protocol). The specifications of this protocol are available at http://www.w3.org/TR/soap/;

— WSDL 2.0 (Web Services Description Language). Language of web service's definition, developed by the Web Services Description Working Group. The specifications of this language are available at http://www.w3.org/2002/ws/desc/; and

— XML (Extensible Markup Language). The language of definition of the data contained on the SOAP is at http://www.w3.org/XML/

A.2.2 Data protection

Data protection regulations in the jurisdiction of use can apply.

A.2.3 Security

The SOAP interface described in this document uses SecurityTokenReference security method, as specified in WS Security 1.1. This token works similarly to the Oauth2 framework. Therefore, a request shall first be made to the authorization server. This authorization server provides a token to access the services described in this document.

The servers shall have a digital certificate for ensure the security through the communication channel defined, using HTTPS over TLS 1.2.

The security criteria defined in this annex shall be understood as minimal, being possible to establish extra criteria to access data.

A.2.4 Header

The header of all requests and responses shall be encapsulated in an HTTP 1.1 transaction; the HTTP 1.1 header shall include the following attribute as specified by the SOAP 1.2 standard:

— Content-Type: application / soap + xml; Charset = utf-8; action = "http: //www.water.domain/ <op>"

Where <op> is the operation to invoke with the SOAP 1.2 protocol. E.g. to invoke the Read method, the following HTTP header shall be included in the HTTP 1.1 transaction:

— Content-Type: application / soap + xml; Charset = utf-8; action = http: //www.water.domain/Read

A.2.5 Server requirements

The server shall provide the URL (uniform resource locator) to access to the Web Services described in this annex. Additionally, any application located at the upper control level shall provide the URI (uniform resource identifier) of the server for each Web Service. The structure of the Web Services is the following URI:

— Http: // <domain_name>: <port> / ManagementServices

It is possible to retrieve the WSDL describing each web service using the following URI:

— Http: // <domain_name>: <port> / ManagementServices?Wsdl

The structure of the authorization server is the following URI:

— Http: // <domain_name>:<port>/Security

It is possible to retrieve the WSDL describing each web service using the following URI:

— Http: // < domain_name >: <port>/Security?Wsdl

A.2.6 Web Services Description Language (WSDL) contract

The WSDL is a description of the web service published by the server that establishes a contract with all its possible clients. It specifies the interface through which a client can gain access to the service and details about how it shall be used.

The server and the clients shall fully respect the WSDL described in this annex (even the sequence of the parameters) to ensure communication through the interface. The WSDL is composed by the classes defined in A.3 and A.4.

A.2.7 Classes and enumerations

The published Web Services can be case sensitive for enumerated values. The UpperCamelCase convention shall be used in all classes and enumerations.

A.3 Implementation classes for web services

A.3.1 Implementation criteria

This subclause describes the methods and classes required to implement the management interface between the MIS applications (management level) and the coordination broker (upper control level), defined at the document architecture Clause 4. The following presents the structure and attributes of the defined classes.

A.3.2 ManagementServices class

This class contains all defined methods.

Table A.1 — ManagementServices class

METHODS			
Name	**Parameters**	**Response**	**Description**
Write	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. PropertyName: PropertyName. Contains one of the values of A.3.25. Value: StringValue. Value of the property to be established.	WriteResponse	This method allows the writing of a property of an irrigation entity. This method in only valid for properties that can be modified,.
Read	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. PropertyName: PropertyName. Contains one of the values of A.3.25.	ReadResponse	This method allows the reading of a property of an irrigation entity.

METHODS			
Name	**Parameters**	**Response**	**Description**
ReadStandardHist	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. PropertyName: PropertyName. Contains one of the values of A.3.25. Date: DateTimeValue. Date from which the data are requested, in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format. NumHist: IntegerValue. Number of samples contained in the standard history of the requested Date. This parameter can use three values: 24, 48 o 96. Statistics: Statistics.	ReadStandardHistResponse	This method allows the reading of the standard history of a property of an irrigation entity. Selection of the parameter Statistics that defines the result obtained by the method.
ReadEvent	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. ProceduralID: StringValue. ID of the element of the procedural model where the user wishes to execute the action. Default value: null. InitialDate: DateTimeValue. Initial date from which the events are requested, in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format. FinalDate: DateTimeValue. Final date until which the events are requested, in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format.	ReadEventResponse	This method allows obtaining the events occurred in an irrigation entity between an initial and a final date defined by the user. The default value for ProceduralID is null, so if no value is entered, it returns all events that occurred in the irrigation entity during the requested time period. When a ProceduralID is introduced, this method returns only the events linked to the ProceduralID and ocurred during the requested time period.

METHODS			
Name	**Parameters**	**Response**	**Description**
CreateRecipe	ActionID: StringValue. ID of the action. Operation: Operation	CreateRecipeResponse	This method allows creating a procedural element (procedure, unit procedure or operation). All the procedural element parameters are contained in the Operation class.
StopRecipe	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. ProceduralID: StringValue. ID of the procedural element where the action shall be executed.	StopRecipeResponse	This method allows stopping the procedural element identified by its ProceduralID. It also requires identifying the irrigation entity.
ReadReport	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. ProceduralID: StringValue. ID of the procedural element where the action shall be executed. InstanceNumber: IntegerValue. Number of the procedural element instance for which the report is required. Default value: 0.	ReadReportResponse	This method allows obtaining the report of a procedural element using its ProceduralID. It also requires identifying the irrigation entity.
ReadProceduralIDs	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. Date: DateTimeValue. Date from which the data are requested in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format. Default value: current day. Status: Status. Default value: Running.	ReadProceduralIDsResponse	This method allows obtaining the list of ProceduralIDs loaded at the subsystem level. If Date parameter is empty, the default value is the current day. If Status parameter is empty, the default value is “Running”.

METHODS			
Name	**Parameters**	**Response**	**Description**
GetProceduralIDs	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. Date: DateTimeValue. Date from which the data are requested in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format. Status: Status.	GetProceduralIDsResponse	This method obtains the list of procedural elements loaded in the Coordination broker. If Date parameter is empty, the default value is the current day. If Status parameter is empty, the default value is "Running".
ReadEntityIDs	ActionID: StringValue. ID of the action.	ReadEntityIDsResponse	This method allows reading the EntityID of all the irrigation entities loaded in the coordination broker.
ReadEntityData	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed.	ReadEntityDataResponse	This method allows reading the static data of an irrigation entity.

A.3.3 Response class

Table A.2 — Response class

Parameter	Type	Description
ResponseCode	IntegerValue	Returns an IntegerValue, according to the list below: 0, 1, 2, 3, 4, 5, 6
ResponseMessage	StringValue	Returns a message explaining the ResponseCode. 0 - Action successfully executed 1 - Execution error 2 - Lexical error 3 - Not supported 4 - Communication error 5 - Coordination error 6 - Unavailable The minimum message shall be one of the listed above. The message can be extended (using a hyphen) with as many explanatory texts as errors can be discriminated by a subsystem or a coordination broker.

The value 6 – Unavailable is only applicable to Read and ReadStandardHist methods.

A.3.4 WriteResponse class

Table A.3 — WriteResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.

A.3.5 ReadResponse class

Table A.4 —ReadResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
Value	StringValue	Current value of the property requested. The value respects the format defined for the property.

A.3.6 ReadStandardHistResponse class

Table A.5 — ReadStandardHistResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3
HistStandardValues	StringValue	Contains a string of fields separated by commas with the value corresponding to the Statistic and NumHist parameters requested. Each piece of data contains two fields: the value and its timestamp of the value at source, in UTC ISO8601 format YYYYMMDDhhmmss±hhmm.

A.3.7 ReadEventResponse class

Table A.6 — ReadEventResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
EventValues	ArrayOfEventValues	Contains an array with the events that comply with the conditions of the request separated by commas. Each event is composed by eight fields also separated by commas.

A.3.8 ReadReportResponse class

Table A.7— ReadReportResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
Report	StringValue	Contains the report of the procedural element according to the data existing at the time of the request. The structure of the report shall conform to what is defined for each type of procedural element. Regardless of the state of the report (complete or partial), the structure shall be maintained by returning with null value all the fields that have no value at the time of the request.
Status	Status	Contains one of the values of A.3.19.

A.3.9 CreateRecipeResponse class

Table A.8 — CreateRecipeResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.

A.3.10 StopRecipeResponse class

Table A.9 — StopRecipeResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.

A.3.11 Operation class

Table A.10 — Operation class

Parameter	Type	Description
ProceduralID	StringValue	Contains the ID of the procedural element model that is being to be created.
EntityID	StringValue	Contains the ID of the entity where the procedural element is going to be executed.
EntityType	EntityType	Contains one of the values of A.3.28.
RecipeType	RecipeType	Contains one of the values of A.3.20.
OperationParameters	ArrayOfOperationParameter	Array of parameters of the procedural element.

A.3.12 OperationParameter class

Table A.11 — OperationParameter class

Parameter	Type	Description
RecipeParameter	RecipeParameterType	Contains one of the values of A.3.21.
MonitoringParameterName	MonitoringParameterType	Contains one of the values of A.3.22.
RepetitionParameterName	RepetitionParameterType	Contains one of the values of A.3.23.
WeekCalendarParameterName	WeekCalendarParameterType	Contains one of the values of A.3.24.
ParameterType	ParameterType	Contains one of the values of A.3.18.
ParameterValue	StringValue	Value for the parameter, according to ParameterType format.

A.3.13 ReadProceduralIDsResponse class

Table A.12 — ReadProceduralIDsResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
ProceduralIDs	StringValue	Contains the ProceduralIDs that meet with the parameterization of the request, separated by commas.

A.3.14 ReadEntityIDsResponse class

Table A.13 — Clase ReadEntityIDsResponse

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
EntityIDs	StringValue	Contains the IDs of all the irrigation entities known by the coordination broker, separated by commas.

A.3.15 GetProceduralIDsResponse class

Table A.14 — GetProceduralIDsResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
ProceduralIDs	StringValue	Returns the IDs separated by commas that meet the parameterization marked on the request.

A.3.16 ReadEntityDataResponse class

Table A.15 — ReadEntityDataResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of A.3.3.
EntityID	StringValue	Unique ID of the entity. It is a unique ID within the system.
EntityLevel	EntityLevel	Contains one of the values of A.3.27.
EntityType	EntityType	Contains one of the values of A.3.28.
ControlLevel	ControlLevel	Identification of the control level that is responsible for the hydraulic entity between the ones defined in the architecture. Contains one of the values of A.3.29.
SpatialData	StringValue	UTM coordinates where the entity is located with the EPSG code associated. Values are separated by commas.
PrecursorEntity	StringValue	Entity that occupies the same level in the physical model and that is the origin of the water received by the entity.
SuccessorEntities	StringValue	Entities that occupy the same level in the physical model and that are the destination of the water supplied by the entity.
ChildEntities	StringValue	All the lower level entities that comprise the entity according to the physical model.
ParentEntity	StringValue	Higher level entity that contains the entity, according to the physical model.
InternalFlowOut	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
PressureOut	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
DesignPressure	IntegerValue	Design pressure of the entity, expressed in Pa.
DesignFlowRate	StringValue	Design flow rate of the entity, expressed in m^3/s.
Caliber	IntegerValue	Caliber of the entity, expressed in m.
Temperature	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
RelativeHumidity	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
SolarRadiation	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
WindParameters	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
Rainfall	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
CumulativeVolumeOut	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.

Parameter	Type	Description
CumulativeVolumeIn	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
InternalFlowIn	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
SoilWaterContent	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
SoilWaterPotential	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
SoilTemperature	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
SoilConductivity	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
PressureIn	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
Behavior	Behavior	The expected behavior of the entity. Contains one of the values of A.3.29.
DesignSimultaneity	IntegerValue	Design simultaneity for the entities contained by the entity. This parameter is expressed as a percentage, from 0 to 100, without decimals, referring to the maximum number of outlets that can be simultaneously active.
DesignPower	IntegerValue	Design power of the entity, expressed in W.
DesignCapacity	IntegerValue	Design volume capacity of the entity, expressed in m^3.
DesignAutonomy	IntegerValue	Design autonomy of the water supply without inputs, expressed in days.
ActivityStatus	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
SystemStatus	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
Mode	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
OpeningDegree	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
InstantPower	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
EnergyConsumption	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
EntityPerformance	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
PowerOnTime	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
Capacity	BooleanValue	Informs about whether the extended property exists and is supported in the hydraulic element.
Area	StringValue	Irrigation surface covered, expressed in m^2.

Parameter	Type	Description
MeterReplacementEvent	StringValue	Contains three fields separated by comas: value1, value2, value3 Where: — value1: Installation date of the master volume totalizer where the property CumulativeVolumeOut is obtained. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. — value2: Old meter reading at the time of replacement, m^3. — value3: New meter reading at the time of replacement, m^3.
PowerOnTimeCounterStart	DateTimeValue	Start date of the PowerOntime property counting. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
EnergyMeterCounterStart	DateTimeValue	Start date of the energy meter device counting. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
FiltrationDegree	IntegerValue	Design filtration grade for the entity, expressed in µm.
MaxWorkingPressure	StringValue	Maximum working pressure for the entity, expressed in Pa.
MinWorkingPressure	StringValue	Minimum working pressure for the entity, expressed in Pa.
InjectorNumber	IntegerValue	Number of injectors existing in the entity, expressed in units.
FertigationMethod	FertigationMethod	Contains one of the values of A.3.31.

A.3.17 EventValue class

Table A.16 — EventValue class

Parameter	Type	Description
EventID	StringValue	Contains a character string that corresponds to an event ID. It is a unique ID within the subsystem that generates it.
EventName	StringValue	Contains the event description, according to the EventName string specification (A.3.32).
TimeStamp	DateTimeValue	Date when the event was generated at source, using coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
Relevance	IntegerValue	Contains an integer with the following possible values: — Relevance=0 implies low relevance. — Relevance=1 implies high relevance. This parameter can be configured at each subsystem, responding to user needs.
ProceduralID	StringValue	Contains a character string that corresponds to an element of the procedural model. It is a unique ID within the system.
InstanceNumber	IntegerValue	Contains an integer that corresponds to the instance of the ProceduralID where the event occurs.

A.3.18 ParameterType enumeration

Table A.17 contains the data type that can be contained in a parameter.

Table A.17 — ParameterType enumeration

Value	Description
StringValue	Plain text string.
IntegerValue	Integer number.
DecimalValue	Real number, with dot as a decimal separator.
DateTimeValue	Date in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format.
BooleanValueValue	1 if true, 0 if false.

A.3.19 Status enumeration

Table A.18 contains the list of possible values of the parameter.

Table A.18 — Status enumeration

Value	Description
Idle	The procedural element execution is pending.
Running	The procedural element is being executed.
Completed	The procedural element is finished.
Stopped	The procedural element has been halted.
Unknown	The status of the procedural element is unknown because the status of the irrigation entity where it is executed is unknown.

A.3.20 RecipeType enumeration

Table A.19 contains a value from the enumeration, thatcan be extended with new values.

Table A.19 — RecipeType enumeration

RecipeType	Description	Applicable to
IrrigationRecipe1	Supply of water (or irrigation) by time or volume.	HYS,IHG/VIH and SSS
IrrigationRecipe2	Supply of water (or irrigation) without end condition.	HYS
IrrigationRecipe3	Supply of water (or irrigation) based on a calendar and executions by time or volume.	HYS, IHG/VIH and SSS
IrrigationRecipe4	Supply of water (or irrigation) based on thresholds.	HYS and SSS
PumpingRecipe1	Pumping of irrigation water according to a flow rate set point.	PMS
PumpingRecipe2	Pumping of irrigation water according to flow rate and level set points, typically related to RSV.	PMS
PumpingRecipe3	Pumping of irrigation water according to a pressure set point, typically required by a INT.	PMS
PumpingRecipe4	Pumping of irrigation wate according to flow rate and pressure set points, typically required by a INT.	PMS
PumpingRecipe5	Pumping of irrigation water attending to time habilitation.	PMS
NetworkBranchRecipe1	Procedure of water delivering to one or more recipients, guaranteeing the presure and flow requirements.	NBU

RecipeType	Description	Applicable to
ControlPointRecipe1	Programming of a control point for safety and maintenance purposes.	NCP
ControlPointRecipe2	Programming of a control point to provide specific flow rate or pressure to the irrigation water.	NCP
ControlPointRecipe3	Programming of a control point to monitorize the water input and/or water output.	NCP
FertigationRecipe1	Addition of different fertilizers to the irrigation water by injection of a proportional quantity of fertilizer to a volume of irrigation water.	FTS
FertigationRecipe2	Addition of different fertilizers to the irrigation water by injection of a proportional quantity of fertilizer throughout the Operation.	FTS
FertigationRecipe3	Addition of different fertilizers to the irrigation water by injection of a bulk of fertilizer based on a time criteria during the Operation.	FTS
FertigationRecipe4	Addition of different fertilizers to the irrigation water by injection of a bulk of fertilizer based on a volume criterion during the Operation.	FTS
FertigationRecipe5	Addition of a premixed fertilizer to the irrigation water.	FTS
FiltrationRecipe1	Physical treatment of irrigation water to reduce the amount of solid particles present.	WSF
IrrigationNetworkUnitProcedure1	Procedure of water delivering to one or more recipients, guaranteeing the presure and flow requirements.	INT
SolidSetUnitProcedure1	Procedure for water delivering to one or more irrigation blocks, guaranteeing the presure and flow requirements.	SSS
PumpingUnitProcedure1	Procedure for pumping irrigation water to provide flow with the right pressure conditions.	PMS
ReservoirUnitProcedure1	Procedure for monitoring the storagement of irrigation water.	RSV
FertigationUnitProcedure1	Procedure for irrigation water filtration.	FTS
FiltrationUnitProcedure1	Procedure for execute a set of fertigations.	WFS
IrrigationSectorProcedure1	General control of the main process to be performed (to irrigate), based on the activities developed in the irrigation entities (and its dependences) composing the irrigation sector.	IRA

A.3.21 RecipeParameterType enumeration

Table A.20 contains the list of possible parameter that can be included in a procedural element.

Table A.20 — RecipeParameterType enumeration

Parameter	Type	Description
StartDate	DateTimeValue	Date/time consigned for the start of the procedural element. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.

Parameter	Type	Description
MaxDuration	IntegerValue	Maximum time of duration of the procedural element. Expressed in seconds. No default value, its inclusion is required.
MaxVolume	DecimalValue	Maximum volume during the procedural element. Expressed in m^3. No default value.
OpeningDegree	IntegerValue	Opening degree of the entity, expressed as a percentage. Dead zone ±5. Default value 100.
MonCummulativeVolumeOut	BooleanValue	Activate/deactivate the monitoring of the output volume. Default value 0.
MonInternalFlowOut	BooleanValue	Activate/deactivate the monitoring of the output flow. Default value 0.
MonPressureOut	BooleanValue	Activate/deactivate the monitoring of the output pressure. Default value 0.
MonCumulativeVolumeIn	BooleanValue	Activate/deactivate the monitoring of the input volume. Default value 0.
MonInternalFlowIn	BooleanValue	Activate/deactivate the monitoring of the input flow. Default value 0.
MonPressureIn	BooleanValue	Activate/deactivate the monitoring of the input pressure. Default value 0.
Repetition	BooleanValue	Activate/deactivate the recipe repetition. Default value 0.
WeekCalendar	StringValue	Contains 7 fields, representing a week (first field is for Monday and the last for Sunday according to ISO 8601). Each field can take one of the next values: — Value=1, when the irrigatrionrecipe shall be executed in that day. — Value=0, irrigatrionrecipe shall not be executed in that day. Without default values.
InternalFlowOutSetPoint	DecimalValue	Flow value, with a decimal point, to maintain during the operation execution, in m^3/s.
SourceLevel_InternalFlowOutSetPoint	StringValue	Level-FlowRate values to accomplish during the procedural element execution. The parameter contains one or more Level-FlowRate relations where: — Value 1: Is the minimum level required at source to use the FlowRate setpoint, expressed in metres. — Value 2: Is the flow rate setpoint to follow, expressed in m^3/s. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.
DestinationLevel_InternalFlowOutSetPoint	StringValue	Level-FlowRate values to accomplish during the procedural element execution. The parameter contains one or more Level-FlowRate relations where: — Value 1: Is the minimum level required at destination to use the FlowRate setpoint, expressed in metres. — Value 2: Is the flow rate setpoint to follow, expressed in m^3/s. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.

Parameter	Type	Description
PressureOutSetPoint	DecimalValue	Pressure value, with a decimal point, to maintain during the procedural element execution. Expressed in Pa.
InternalFlowOutSetPoint_ PressureOutSetPoint	StringValue	FlowRate-Pressure values to accomplish during the procedural element execution. The parameter contains one or more FlowRate-Pressure relations where: — Value 1: Is the flow rate required for a certain value 2, expressed in m^3/s. — Value 2: Is the pressure setpoint to follow for the corresponding value 1, expressed in Pa. The value 1 shall be measured to define which value 2 is used as setpoint. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.
MaxRainfall	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum rainfall, with a decimal point, that stops the Operation during its execution. Expressed in $m^3/m^2/s$. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MaxWindSpeed	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum wind speed, with a decimal point, obtained from the WindParameters property that stops the Operation during its execution. Expressed in m/s. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MinSoilWaterContent	StringValue	Contains 2 fields separated by commas: — Field 1. Minimum value of the WaterSoilContent property, with a decimal point, that starts the Operation. Expressed in %. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MaxSoilWaterContent	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum value of the WaterSoilContent, with a decimal point, that stops the Operation during its execution. Expressed in %. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MinSoilWaterPotential	StringValue	Contains 2 fields separated by commas: — Field 1. Minimum value of the WaterSoilPotential property, with a decimal point, that starts the Operation. Expressed in Pa. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.

Parameter	Type	Description
MaxSoilWaterPotential	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum value of the WaterSoilPotential, with a decimal point, that stops the Operation during its execution. Expressed in Pa. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
SolarRadiationTriggerOn	StringValue	Contains 2 fields separated by commas: — Field 1. Value of the SolarRadiation property, with a decimal point, that starts the Operation. Expressed in W/m^2. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
SolarRadiationTriggerOff	StringValue	Contains 2 fields separated by commas: — Field 1. Value of the SolarRadiation property, with a decimal point, that stops the Operation during its execution. Expressed in W/m^2. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MaxLevel	DecimalValue	Level setpoint which determines that the entity is at full capacity for the closing of its water inputs. With a decimal point and expresed in m.
MinLevel	DecimalValue	Level setpoint which determines that the entity is at empty capacity for the closing of its water outputs. With a decimal point and expresed in m.
MaxTime	IntegerValue	Maximum time between filter flushing, in s.
DifferentialPressure	DecimalValue	Pressure difference, measured between filter inlet and outlet, that activates the filter flushing. Expressed in Pa. No default value.
QuantityRatioSetPoint	StringValue	String composed of one or more groups of four fields separated by commas. Each group corresponds to an injector parameterization. The can contain as many groups as DSM has the entity. The fields 3 and 4 of each group are mutually exclusive: may both be empty but not filled in simultaneously. Each group has: — Value1: DSM number, expressed as an IntegerValue. — Value2: quantity of fertilizer, expressed in m^3, to be proportionally dosified per volume unit of irrigation water and directly related to the total volume to be fertilized (MaxVolume). — Value3: pH setpoint for dosifying adjust, expressed as a real. — Value4: electrical conductivity for dosifying adjust, expressed as a real. The value2 can be adapted attending to value3 or value4 setpoint.

Parameter	Type	Description
TimeRatioSetPoint	StringValue	String composed of one or more groups of four fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. The fields 3 and 4 of each group are mutually exclusive: may both be empty but not filled in simultaneously. Each group has: — Value1: DSM number, expressed as an IntegerValue. — Value2: quantity of time, expressed in seconds, during which the fertilizer is going to be proportionally dosified until the Operation reaches its end by time (MaxDuration). — Value3: pH setpoint for dosifying adjust, expressed as a real. — Value4: electrical conductivity for dosifying adjust, expressed as a real. The value2 can be adapted attending to value3 or value4 setpoint.
pHSetPoint	DecimalValue	Acidity value, with a decimal point, to maintain during the Operation execution.
ECSetPoint	DecimalValue	Electrical conductivity value, with a decimal point, to maintain during the Operation execution, in S/m.
FertigationTime	StringValue	String composed of one or more groups of two fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. Each group has: — Value1: DSM number, expressed as an IntegerValue. — Value2: Time to apply the fertilizer bulk, expressed in s.
FertigationVolume	StringValue	String composed of one or more groups of two fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. Each group has: — Value1: DSM number, expressed as an IntegerValue. — Value2: Volume of fertilizer to apply as fertigation bulk, expressed in m^3.

A.3.22 MonitoringParameterType enumeration

Table A.21 contains a list of parameters to introduce when a monitoring is enabled.

Table A.21 — MonitoringParameterType enumeration

Parameter	Type	Description
ThresholdHH	DecimalValue	Very high threshold value of the variable for which monitoring is activated. When the ThresholdHH is exceeded, an event is generated. This is a required parameter when MonCummulativeVolumeOut and/or MonCummulativeVolumeIn are activated. This parameter can also be required when the ThresholdLL are not established and MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn are activated.
ThresholdLL	DecimalValue	Very low threshold value of the variable for which monitoring is activated. When the ThresholdLL is exceeded, an event is generated. This parameter is required when the ThresholdHH are not established and MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn are activated.
CmdHH	BooleanValueValue	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdHH. Default value 0. In case this parameter is used with MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn it shall be used with TimeHH.
CmdLL	BooleanValue	Its activation forces the Operation to stop when ThresholdLL is reached during the configured time. Parameter conditioned by ThresholdLL. Default value 0. In case this parameter is used with MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn it shall be used with TimeLL.
TimeHH	IntegerValue	Time in seconds where the ThresholdHH shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
TimeLL	IntegerValue	Time in seconds where the ThresholdLL shall be exceeded. Parameter conditioned by ThresholdLL. Default value 30 s.

A.3.23 RepetitionParameterType enumeration

Table A.22 contains a list of parameters to introduce when the Repetition parameter is enabled.

Table A.22 — RepetitionParameterType enumeration

Parameter	Type	Description
RepetitionInterval	IntegerValue	Time interval between repetitions, expressed in seconds and added to the StartDate parameter of the last execution.
RepetitionNumber	IntegerValue	Number of times that the RepetitionInterval parameter shall be repeated. Without limitation.

A.3.24 WeekCalendarParameterType enumeration

Table A.23 contains a list of parameters to introduce when the WeekCalendar parameter is enabled.

Table A.23 — WeekCalendarParameterType enumeration

Parameter	Type	Description
FinalDate	DateTimeValue	Date/time consigned for the end of the procedural element. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.

A.3.25 PropertyName enumeration

Table A.24 contains the list of properties defined in this document for the irrigation entities.

Table A.24 — PropertyName enumeration

Value	Description
ActivityStatus	Entity activity status
SystemStatus	Subsystem status
Mode	Entity functioning mode
CumulativeVolumeOut	Entity cumulative output volume
InternalFlowOut	Entity output flow
PressureOut	Entity output pressure
OpeningDegree	Entity opening degree
Temperature	Temperature registered at the entity
RelativeHumidity	Relative humidity registered at the entity
SolarRadiation	Solar radiation registered at the entity
WindParameters	Wind speed and direction registered at the entity
Rainfall	Rainfall registered at the entity
CumulativeVolumeIn	Entity cumulative input volume
InternalFlowIn	Entity input flow
PressureIn	Entity input pressure
InstantPower	Entity absorbed instant power during the execution of its functionalities
EnergyConsumption	Entity energy consumption
EntityPerformance	Entity hydraulic performance
PowerOnTime	Entity accumulated activity time
Capacity	Water contained by the entity
SoilWaterContent	Water contained by the soil in a reference area
SoilWaterPotential	Energy state of water in the soil in a reference area
SoilTemperature	Temperature in the soil in a reference area
SoilConductivity	Conductivity in the soil in a reference area
DifferentialPressure	Differential pressure registered between the input and output of the entity
WaterAcidity	Acidity of the irrigation water registered in the entity
WaterConductivity	Electrical conductivity of the irrigation water registered in the entity

A.3.26 Statistics enumeration

Table A.25 contains the type of statistical for the calculation of the standard history. The enumeration contains the following values.

Table A.25 — Statistics enumeration

Value	Description
Last	Returns the last value registered for each time slot of a standard history.
Average	Returns the average value registered for each time slot of a standard history.
Maximum	Returns the maximum value registered for each time slot of a standard history.
Minimum	Returns the minimum value registered for each time slot of a standard history.

A.3.27 EntityLevel enumeration

Table A.26 contains the levels of the physical model to which an irrigation entity can belong.

Table A.26 — EntityLevel enumeration

Value
Empty
Area
ProcessCell
Unit
EquipmentModule

A.3.28 EntityType enumeration

Table A.27 contains the listed entities included in the physical model.

Table A.27 — EntityType enumeration

List of values				
IRA	PMS	INT	RSV	SSS
PML	HYS	SSB	NCP	NBU
IHG	VIH	MIH	WFS	FTL
FTS	DSM	DLT	FRT	Empty

A.3.29 ControlLevel enumeration

Table A.28 contains the possible control levels of an irrigation entity.

Table A.28 — ControlLevel enumeration

List of values
Empty
HigherControlLevel
LowerControlLevel

A.3.30 Behavior enumeration

Table A.29 contains the possible behaviors of an IHG irrigation entity type as defined in the physical model.

Table A.29 — Behavior enumeration

List of values
Empty
Behavior1
Behavior2

A.3.31 FertigationMethod enumeration

Table A.30 contains the fertigation methods of a FTS entity.

Table A.30 — FertigationMethod enumeration

List of values
Empty
InLineInjection
DilutioneTank

A.3.32 EventName string

Table A.31 shall be used taking into account the values defining each event. Table A.31 can be extended with new values. A subsystem exchanges the events assigned to the entity that it controls.

Table A.31 — EventName string

EventName-Field1	EventName-Field2	EventName-Field3	Description
ActivityStatusChanged	Previous value	Current value	The ActivityStatus property changes its value.
SystemStatusUnknown	Previous value	Current value	The SystemStatus property changes its value to unknown.
ModeChanged	Previous value	Current value	The Mode property changes its value.
InternalFlowOutIsNot0OpeningDegreeIs0	Active/Inactive		Activation/deactivation of flow other than zero with entity closed event.
GeneralFailure	Active/Inactive		Activation/deactivation of a general failure in an unit entity. The entity is not available when the event is active.
Failure	Active/Inactive		Failure in on status in an equipment module entity. The entity is not available when the event is active.
CumulativeVolumeOutSync	Previous value	Current value	The CumulativeVolumeOut property has been synchronized.
MinimumCapacity	Active/Inactive	Current value	Activation/deactivation of minimum level reached on a source entity. Related to Capacity property.

EventName-Field1	EventName-Field2	EventName-Field3	Description
MaximumCapacity	Active/Inactive	Current value	Activation/deactivation of maximum level reached on a destination entity. Related to Capacity property.
CumulativeVolumeIn≠CumulativeVolumeOut			The CumulativeVolumeOut and CumulativeVolumeIn properties, has not the same value (±5%).
ContinuosCleaning	Active/Inactive		Activation/deactivation of a failure on in filter flushing.
MaxWorkingPressure	Active/Inactive		Activation/deactivation of maximum working pressure detected.
MinWorkingPressure	Active/Inactive		Activation/deactivation of minimum working pressure detected.
AutonomyWarning	Active/Inactive	Remaining autonomy (voltage)	Activation/deactivation of an event: the energy autonomy of the irrigation entity is near to end.
AccessWarning	Active/Inactive		Activation/deactivation of a warning of physical access to the irrigation entity.
SubsystemEvent	Available to define	Available to define	Open event for proprietary events of a subsystem.
AccumulativePropertySync	Previous value	Current value	The value of a accumulative property has been synchronized.
StartedOperation			A procedural element has changed to the status Running.
StoppedOperation			A running or idle procedural element reaches the Stopped status (following the StopRecipe action).
CompletedOperation			A running procedural element reaches the Completed status.
OperationErrorStarting	Active/Inactive		A procedural element was not correctly executed. Although the start condition was reached, the Operation did not start.
OperationErrorEnding	Active/Inactive		A procedural element was not executed correctly. When the end condition was reached, the Operation had not finished.
InternalFlowOutIs0OpeningDegreeIsNot0	Active/Inactive		Alarm of zero flow with irrigation entity open. To generate the event with Inactive value in field 2, is required that previously the field 2 of the event has been in Active.
ThresholdHHTriggeredOn	PropertyName	Current value	The ThresholdHH of a monitoring property has been reached.
ThresholdHHTriggeredOff	PropertyName	Current value	The ThresholdHH of a monitoring property has been deactivated.
ThresholdLLTriggeredOn	PropertyName	Current value	The ThresholdLL of a monitoring property has been reached.
ThresholdLLTriggeredOff	PropertyName	Current value	The ThresholdLL of a monitoring property has been deactivated.

EventName-Field1	EventName-Field2	EventName-Field3	Description
OperationErrorSetPoint			A procedural element was not correctly executed. The subsystem can not reach or maintain the setpoint introduced.
OperationOnRecipient			A procedural element was received by the destination device.

A.4 Implementation classes for authorization server

A.4.1 ManagementSecurity class

Table A.32 — ManagementSecurity class

METHODS			
Name	**Name**	**Name**	**Name**
GetToken	grant_type: StringValue. This parameter shall have the next fixed value: "Password". username: StringValue. User to be authenticated by the server. password: StringValue. Password related to the user to be authenticated. scope: StringValue. Optional parameter to define limitations for the requested token, for example for read only operations.	TokenResponse	This method allows the obtaining of the token to access the different services.
RefreshToken	grant_type: StringValue. This parameter shall have the next fixed value: "Refresh_token". refresh_token: StringValue. Value received in the TokenResponse request as Refresh_token value. scope: StringValue. Optional parameter to define limitations for the requested token, for example for read only operations.	TokenResponse	This method allows the token to be updated if it has expired.

A.4.2 TokenResponse class

Table A.33 — TokenResponse class

Parameter	Type	Description
access_token	StringValue	Token that shall be added to requests in order to access services.
token_type	StringValue	This parameter shall have the next fixed value: "Bearer", corresponding to the token type to be used.

Parameter	Type	Description
access_token	StringValue	Token that shall be added to requests in order to access services.
expires_in	StringValue	Validity time of the token provided, expressed in s.
refresh_token	StringValue	When the expiration time has been reached, if the token is used an error message shall be received. This value can be used to avoid this error and receive a new token.
scope	StringValue	The same value included in the request.

A.5 WSDL

```
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://www.water.domain" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://www.water.domain">
    <wsdl:types>
        <s:schema targetNamespace="http://www.water.domain" elementFormDefault="qualified">
            <s:simpleType name="RecipeType">
                <s:restriction base="s:string">
                    <s:enumeration value="IrrigationRecipe1"/>
                    <s:enumeration value="IrrigationRecipe2"/>
                    <s:enumeration value="IrrigationRecipe3"/>
                    <s:enumeration value="IrrigationRecipe4"/>
                    <s:enumeration value="PumpingRecipe1"/>
                    <s:enumeration value="PumpingRecipe2"/>
                    <s:enumeration value="PumpingRecipe3"/>
                    <s:enumeration value="PumpingRecipe4"/>
                    <s:enumeration value="PumpingRecipe5"/>
                    <s:enumeration value="NetworkBranchRecipe1"/>
                    <s:enumeration value="ControlPointRecipe1"/>
                    <s:enumeration value="ControlPointRecipe2"/>
                    <s:enumeration value="ControlPointRecipe3"/>
                    <s:enumeration value="FertigationRecipe1"/>
                    <s:enumeration value="FertigationRecipe2"/>
                    <s:enumeration value="FertigationRecipe3"/>
                    <s:enumeration value="FertigationRecipe4"/>
                    <s:enumeration value="FertigationRecipe5"/>
                    <s:enumeration value="FiltrationRecipe1"/>
                    <s:enumeration value="IrrigationNetworkUnitProcedure1"/>
                    <s:enumeration value="SolidSetUnitProcedure1"/>
                    <s:enumeration value="PumpingUnitProcedure1"/>
                    <s:enumeration value="ReservoirUnitProcedure1"/>
                    <s:enumeration value="FiltrationUnitProcedure1"/>
                    <s:enumeration value="FertigationUnitProcedure1"/>
                    <s:enumeration value="IrrigationSectorProcedure1"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="EntityType">
                <s:restriction base="s:string">
                    <s:enumeration value="IRA"/>
                    <s:enumeration value="PMS"/>
                    <s:enumeration value="RSV"/>
```

```
                    <s:enumeration value="INT"/>
                    <s:enumeration value="SSS"/>
                    <s:enumeration value="PML"/>
                    <s:enumeration value="HYS"/>
                    <s:enumeration value="SSB"/>
                    <s:enumeration value="NCP"/>
                    <s:enumeration value="VIH"/>
                    <s:enumeration value="IHG"/>
                    <s:enumeration value="NBU"/>
                    <s:enumeration value="MIH"/>
                    <s:enumeration value="WFS"/>
                    <s:enumeration value="FTL"/>
                    <s:enumeration value="FTS"/>
                    <s:enumeration value="DSM"/>
                    <s:enumeration value="DLT"/>
                    <s:enumeration value="FRT"/>
                    <s:enumeration value="Empty"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="ControlLevel">
                <s:restriction base="s:string">
                    <s:enumeration value="Empty"/>
                    <s:enumeration value="HigherControlLevel"/>
                    <s:enumeration value="LowerControlLevel"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="EntityLevel">
                <s:restriction base="s:string">
                    <s:enumeration value="Empty"/>
                    <s:enumeration value="Area"/>
                    <s:enumeration value="ProcessCell"/>
                    <s:enumeration value="Unit"/>
                    <s:enumeration value="EquipmentModule"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="MonitoringParameterType">
                <s:restriction base="s:string">
                    <s:enumeration value="ThresholdHH"/>
                    <s:enumeration value="CmdHH"/>
                    <s:enumeration value="TimeHH"/>
                    <s:enumeration value="ThresholdLL"/>
                    <s:enumeration value="TimeLL"/>
                    <s:enumeration value="CmdLL"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="RepetitionParameterType">
                <s:restriction base="s:string">
                    <s:enumeration value="RepetitionInterval"/>
                    <s:enumeration value="RepetitionNumber"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="WeekCalendarParameterType">
                <s:restriction base="s:string">
                    <s:enumeration value="FinalDate"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="RecipeParameterType">
```

```
                <s:restriction base="s:string">
                    <s:enumeration value="StartDate"/>
                    <s:enumeration value="MaxDuration"/>
                    <s:enumeration value="MaxVolume"/>
                    <s:enumeration value="OpeningDegree"/>
                    <s:enumeration value="MonCumulativeVolumeOut"/>
                    <s:enumeration value="MonInternalFlowOut"/>
                    <s:enumeration value="MonPressureOut"/>
                    <s:enumeration value="MonCumulativeVolumeIn"/>
                    <s:enumeration value="MonInternalFlowIn"/>
                    <s:enumeration value="MonPressureIn"/>
                    <s:enumeration value="Repetition"/>
                    <s:enumeration value="WeekCalendar"/>
                    <s:enumeration value="InternalFlowOutSetPoint"/>
                    <s:enumeration value="SourceLevel_InternalFlowOutSetPoint"/>
                    <s:enumeration value="DestinationLevel_InternalFlowOutSetPoint"/>
                    <s:enumeration value="InternalFlowOutSetPoint_PressureOutSetPoint"/>
                    <s:enumeration value="PressureOutSetPoint"/>
                    <s:enumeration value="MaxRainfall"/>
                    <s:enumeration value="MaxWindSpeed"/>
                    <s:enumeration value="MinSoilWaterContent"/>
                    <s:enumeration value="MaxSoilWaterContent"/>
                    <s:enumeration value="MinSoilWaterPotential"/>
                    <s:enumeration value="MaxSoilWaterPotential"/>
                    <s:enumeration value="SolarRadiationTriggerOn"/>
                    <s:enumeration value="SolarRadiationTriggerOff"/>
                    <s:enumeration value="MaxLevel"/>
                    <s:enumeration value="MinLevel"/>
                    <s:enumeration value="MaxTime"/>
                    <s:enumeration value="DifferentialPressure"/>
                    <s:enumeration value="QuantityRatioSetPoint"/>
                    <s:enumeration value="pHSetPoint"/>
                    <s:enumeration value="ECSetPoint"/>
                    <s:enumeration value="TimeRatioSetPoint"/>
                    <s:enumeration value="FertigationTime"/>
                    <s:enumeration value="FertigationVolume"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="ParameterType">
                <s:restriction base="s:string">
                    <s:enumeration value="StringValue"/>
                    <s:enumeration value="IntegerValue"/>
                    <s:enumeration value="DecimalValue"/>
                    <s:enumeration value="DateTimeValue"/>
                    <s:enumeration value="BooleanValue"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="Status">
                <s:restriction base="s:string">
                    <s:enumeration value="Idle"/>
                    <s:enumeration value="Running"/>
                    <s:enumeration value="Stopped"/>
                    <s:enumeration value="Completed"/>
                    <s:enumeration value="Unknown"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="Statistics">
```

```
                    <s:restriction base="s:string">
                        <s:enumeration value="Last"/>
                        <s:enumeration value="Average"/>
                        <s:enumeration value="Maximum"/>
                        <s:enumeration value="Minimum"/>
                    </s:restriction>
                </s:simpleType>
                <s:simpleType name="Behavior">
                    <s:restriction base="s:string">
                        <s:enumeration value="Empty"/>
                        <s:enumeration value="Behavior1"/>
                        <s:enumeration value="Behavior2"/>
                    </s:restriction>
                </s:simpleType>
                <s:simpleType name="FertigationMethod">
                    <s:restriction base="s:string">
                        <s:enumeration value="Empty"/>
                        <s:enumeration value="InLineInjection"/>
                        <s:enumeration value="DilutioneTank"/>
                    </s:restriction>
                </s:simpleType>
                <s:element name="CreateRecipe">
                    <s:complexType>
                        <s:sequence>
                            <s:element name="ActionID" type="s:string" minOccurs="1" maxOccurs="1"/>
                            <s:element name="Operation" type="tns:Operation" minOccurs="1" maxOccurs="1"/>
                        </s:sequence>
                    </s:complexType>
                </s:element>
                <s:complexType name="ArrayOfMonitoringParameter">
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="unbounded" name="MonitoringParameter" nillable="true"
type="tns:MonitoringParameter"/>
                    </s:sequence>
                </s:complexType>
                <s:complexType name="MonitoringParameter">
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="MonitoringParameterName"
type="tns:MonitoringParameterType"/>
                        <s:element minOccurs="1" maxOccurs="1" name="MonitoringParameterValue" type="s:string"/>
                        <s:element minOccurs="1" maxOccurs="1" name="MonitoringParameterType" type="tns:ParameterType"/>
                    </s:sequence>
                </s:complexType>
                <s:complexType name="ArrayOfRepetitionParameter">
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="2" name="RepetitionParameter" nillable="true"
type="tns:RepetitionParameter"/>
                    </s:sequence>
                </s:complexType>
                <s:complexType name="RepetitionParameter">
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="RepetitionParameterName"
type="tns:RepetitionParameterType"/>
                        <s:element minOccurs="1" maxOccurs="1" name="RepetitionParameterValue" type="s:string"/>
                        <s:element minOccurs="1" maxOccurs="1" name="RepetitionParameterType"  type="tns:ParameterType"/>
                    </s:sequence>
                </s:complexType>
```

```
<s:complexType name="WeekCalendarParameter">
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="WeekCalendarParameterName" type="tns:WeekCalendarParameterType"/>
<s:element minOccurs="1" maxOccurs="1" name="WeekCalendarParameterValue"  type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="WeekCalendarParameterType"   type="tns:ParameterType"/>
</s:sequence>
</s:complexType>
<s:complexType name="Operation">
<s:sequence>
<s:element name="ProceduralID" type="s:string" minOccurs="1" maxOccurs="1"/>
<s:element name="EntityID" type="s:string" minOccurs="1" maxOccurs="1"/>
<s:element name="EntityType" type="s:string" minOccurs="1" maxOccurs="1"/>
<s:element name="RecipeType" type="tns:RecipeType" minOccurs="1" maxOccurs="1"/>
<s:element name="OperationParameters" type="tns:ArrayOfOperationParameter" minOccurs="1" maxOccurs="1"/>
</s:sequence>
</s:complexType>
<s:complexType name="ArrayOfOperationParameter">
<s:sequence>
<s:element name="OperationParameter" type="tns:OperationParameter" nillable="true" minOccurs="1" maxOccurs="unbounded"/>
</s:sequence>
</s:complexType>
<s:complexType name="OperationParameter">
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="RecipeParameter" type="tns:RecipeParameterType"/>
<s:element minOccurs="1" maxOccurs="1" name="ParameterValue" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="ParameterType" type="tns:ParameterType"/>

<s:element minOccurs="0" maxOccurs="1" name="MonitoringParameters" type="tns:ArrayOfMonitoringParameter"/>
<s:element minOccurs="0" maxOccurs="1" name="RepetitionParameters" type="tns:ArrayOfRepetitionParameter"/>
<s:element minOccurs="0" maxOccurs="1" name="WeekCalendarParameter" type="tns:WeekCalendarParameter"/>
</s:sequence>
</s:complexType>
<s:complexType name="ArrayOfEventValues">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="EventValue" nillable="true" type="tns:EventValue"/>
</s:sequence>
</s:complexType>
<s:complexType name="EventValue">
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="EventID" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="EventName" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="TimeStamp" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="Relevance" type="s:integer"/>
<s:element minOccurs="1" maxOccurs="1" name="ProceduralID" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="InstanceNumber" type="s:integer"/>
</s:sequence>
</s:complexType>
<s:complexType name="Response">
<s:sequence>
<s:element name="ResponseCode" type="s:int" minOccurs="1" maxOccurs="1"/>
```

```
                    <s:element name="ResponseMessage" type="s:string" minOccurs="0" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="CreateRecipeResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="CreateRecipeResponse" type="tns:CreateRecipeResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="CreateRecipeResult">
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="ReadProceduralIDs">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                        <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                        <s:element name="Date" type="s:string" maxOccurs="1"/>
                        <s:element name="Status" type="tns:Status" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="ReadProceduralIDsResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ReadProceduralIDsResponse" type="tns:ReadProceduralIDsResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="ReadProceduralIDsResult">
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                    <s:element name="ProceduralIDs" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="GetProceduralIDs">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                        <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                        <s:element name="Date" type="s:string" maxOccurs="1"/>
                        <s:element name="Status" type="tns:Status" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="GetProceduralIDsResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="GetProceduralIDsResponse" type="tns:GetProceduralIDsResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
```

```
        <s:complexType name="GetProceduralIDsResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="ProceduralIDs" type="s:string" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="Write">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="PropertyName" type="s:string" maxOccurs="1"/>
                    <s:element name="Value" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="WriteResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="WriteResponse" type="tns:WriteResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="WriteResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="Read">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="PropertyName" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadResponse" type="tns:ReadResult" minOccurs="0" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="Value" type="s:string" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="ReadStandardHist">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
```

```
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="PropertyName" type="s:string" maxOccurs="1"/>
                    <s:element name="Date" type="s:string" maxOccurs="1"/>
                    <s:element name="NumHist" type="s:int" maxOccurs="1"/>
                    <s:element name="Statistics" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadStandardHistResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadStandardHistResponse" type="tns:ReadStandardHistoryResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadStandardHistoryResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="HistStandardValues" type="s:string" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="ReadEvent">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="ProceduralID" type="s:string" maxOccurs="1"/>
                    <s:element name="InitialDate" type="s:string" maxOccurs="1"/>
                    <s:element name="FinalDate" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadEventResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadEventResponse" type="tns:ReadEventResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadEventResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="EventValues" type="tns:ArrayOfEventValues" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="StopRecipe">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="ProceduralID" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="StopRecipeResponse">
```

```
            <s:complexType>
                <s:sequence>
                    <s:element name="StopRecipeResponse" type="tns:StopResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="StopResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="ReadReport">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="ProceduralID" type="s:string" maxOccurs="1"/>
                    <s:element name="InstanceNumber" type="s:int" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadReportResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadReportResponse" type="tns:ReadReportResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadReportResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="Report" type="s:string" maxOccurs="1"/>
                <s:element name="Status" type="tns:Status" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="ReadEntityIDs">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadEntitiesIDsResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string"/>
                <s:element name="Response" type="tns:Response"/>
                <s:element name="EntityIDs" type="s:string"/>
            </s:sequence>
        </s:complexType>
        <s:element name="ReadEntityIDsResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadEntityIDsResponse" type="tns:ReadEntitiesIDsResult"/>
                </s:sequence>
            </s:complexType>
```

```
        </s:element>
        <s:element name="ReadEntityData">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string"/>
                    <s:element name="EntityID" type="s:string"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadEntityDataResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string"/>
                <s:element name="Response" type="tns:Response"/>
                <s:element name="EntityLevel" type="tns:EntityLevel"/>
                <s:element name="EntityType" type="tns:EntityType"/>
                <s:element name="ControlLevel" type="tns:ControlLevel"/>
                <s:element name="SpatialData" type="s:string"/>
                <s:element name="PrecursorEntities" type="s:string"/>
                <s:element name="SuccessorEntities" type="s:string"/>
                <s:element name="ChildEntities" type="s:string"/>
                <s:element name="ParentEntity" type="s:string"/>
                <s:element name="InternalFlowOut" type="s:boolean"/>
                <s:element name="PressureOut" type="s:boolean"/>
                <s:element name="DesignPressure" type="s: integer"/>
                <s:element name="DesignFlowRate" type="s:string"/>
                <s:element name="Caliber" type="s: integer"/>
                <s:element name="Temperature" type="s:boolean"/>
                <s:element name="RelativeHumidity" type="s:boolean"/>
                <s:element name="SolarRadiation" type="s:boolean"/>
                <s:element name="WindParameters" type="s:boolean"/>
                <s:element name="Rainfall" type="s:boolean"/>
                <s:element name="SoilWaterContent" type="s:boolean"/>
                <s:element name="SoilWaterPotential" type="s:boolean"/>
                <s:element name="SoilConductivity" type="s:boolean"/>
                <s:element name="SoilTemperature" type="s:boolean"/>
                <s:element name="CumulativeVolumeOut" type="s:boolean"/>
                <s:element name="CumulativeVolumeIn" type="s:boolean"/>
                <s:element name="InternalFlowIn" type="s:boolean"/>
                <s:element name="PressureIn" type="s:boolean"/>
                <s:element name="Behavior" type="tns:Behavior"/>
                <s:element name="DesignSimultaneity" type="s:integer"/>
                <s:element name="DesignCapacity" type="s: integer"/>
                <s:element name="DesignPower" type="s: integer"/>
                <s:element name="DesignAutonomy" type="s: integer"/>
                <s:element name="ActivityStatus" type="s:boolean"/>
                <s:element name="SystemStatus" type="s:boolean"/>
                <s:element name="Mode" type="s:boolean"/>
                <s:element name="OpeningDegree" type="s:boolean"/>
                <s:element name="InstantPower" type="s:boolean"/>
                <s:element name="EnergyConsumption" type="s:boolean"/>
                <s:element name="EntityPerformance" type="s:boolean"/>
                <s:element name="PowerOnTime" type="s:boolean"/>
                <s:element name="Capacity" type="s:boolean"/>
                <s:element name="Area" type="s: integer"/>
                <s:element name="MeterReplacementEvent" type="s:string"/>
                <s:element name="PowerOnTimeCounterStart" type="s:string"/>
                <s:element name="EnergyMeterCounterStart" type="s:string"/>
```

```
                    <s:element name="FiltrationDegree" type="s: integer"/>
                    <s:element name="MaxWorkingPressure" type="s:string"/>
                    <s:element name="MinWorkingPressure" type="s:string"/>
                    <s:element name="InjectorNumber" type="s:integer"/>
                    <s:element name="FertigationMethod" type="tns:FertigationMethod"/>
                </s:sequence>
            </s:complexType>
            <s:element name="ReadEntityDataResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ReadEntityDataResponse" type="tns:ReadEntityDataResult"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
        </s:schema>
    </wsdl:types>
    <wsdl:message name="CreateRecipeSoapIn">
        <wsdl:part name="parameters" element="tns:CreateRecipe"/>
    </wsdl:message>
    <wsdl:message name="CreateRecipeSoapOut">
        <wsdl:part name="parameters" element="tns:CreateRecipeResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadProceduralIDsSoapIn">
        <wsdl:part name="parameters" element="tns:ReadProceduralIDs"/>
    </wsdl:message>
    <wsdl:message name="ReadProceduralIDsSoapOut">
        <wsdl:part name="parameters" element="tns:ReadProceduralIDsResponse"/>
    </wsdl:message>
    <wsdl:message name="GetProceduralIDsSoapIn">
        <wsdl:part name="parameters" element="tns:GetProceduralIDs"/>
    </wsdl:message>
    <wsdl:message name="GetProceduralIDsSoapOut">
        <wsdl:part name="parameters" element="tns:GetProceduralIDsResponse"/>
    </wsdl:message>
    <wsdl:message name="WriteSoapIn">
        <wsdl:part name="parameters" element="tns:Write"/>
    </wsdl:message>
    <wsdl:message name="WriteSoapOut">
        <wsdl:part name="parameters" element="tns:WriteResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadSoapIn">
        <wsdl:part name="parameters" element="tns:Read"/>
    </wsdl:message>
    <wsdl:message name="ReadSoapOut">
        <wsdl:part name="parameters" element="tns:ReadResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadStandardHistSoapIn">
        <wsdl:part name="parameters" element="tns:ReadStandardHist"/>
    </wsdl:message>
    <wsdl:message name="ReadStandardHistSoapOut">
        <wsdl:part name="parameters" element="tns:ReadStandardHistResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadEventSoapIn">
        <wsdl:part name="parameters" element="tns:ReadEvent"/>
    </wsdl:message>
    <wsdl:message name="ReadEventSoapOut">
        <wsdl:part name="parameters" element="tns:ReadEventResponse"/>
```

```
    </wsdl:message>
    <wsdl:message name="StopRecipeSoapIn">
        <wsdl:part name="parameters" element="tns:StopRecipe"/>
    </wsdl:message>
    <wsdl:message name="StopRecipeSoapOut">
        <wsdl:part name="parameters" element="tns:StopRecipeResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadReportSoapIn">
        <wsdl:part name="parameters" element="tns:ReadReport"/>
    </wsdl:message>
    <wsdl:message name="ReadReportSoapOut">
        <wsdl:part name="parameters" element="tns:ReadReportResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadEntityIDsSoapIn">
        <wsdl:part name="parameters" element="tns:ReadEntityIDs"/>
    </wsdl:message>
    <wsdl:message name="ReadEntityIDsSoapOut">
        <wsdl:part name="parameters" element="tns:ReadEntityIDsResponse"/>
    </wsdl:message>
    <wsdl:message name="ReadEntityDataSoapIn">
        <wsdl:part name="parameters" element="tns:ReadEntityData"/>
    </wsdl:message>
    <wsdl:message name="ReadEntityDataSoapOut">
        <wsdl:part name="parameters" element="tns:ReadEntityDataResponse"/>
    </wsdl:message>
    <wsdl:portType name="ManagementServicesPortType">
        <wsdl:operation name="CreateRecipe">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Creates a procedural
element</wsdl:documentation>
            <wsdl:input message="tns:CreateRecipeSoapIn"/>
            <wsdl:output message="tns:CreateRecipeSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="ReadProceduralIDs">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns the operations loaded at subsystem
level</wsdl:documentation>
            <wsdl:input message="tns:ReadProceduralIDsSoapIn"/>
            <wsdl:output message="tns:ReadProceduralIDsSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="GetProceduralIDs">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns the operations loaded at coordination level
</wsdl:documentation>
            <wsdl:input message="tns:GetProceduralIDsSoapIn"/>
            <wsdl:output message="tns:GetProceduralIDsSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="Write">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Writes a value for an entity's property that allows
it</wsdl:documentation>
            <wsdl:input message="tns:WriteSoapIn"/>
            <wsdl:output message="tns:WriteSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="Read">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Reads the current value of an entity's
property</wsdl:documentation>
            <wsdl:input message="tns:ReadSoapIn"/>
            <wsdl:output message="tns:ReadSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="ReadStandardHist">
```

```
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Reads the standard history of an entity's property </wsdl:documentation>
                <wsdl:input message="tns:ReadStandardHistSoapIn"/>
                <wsdl:output message="tns:ReadStandardHistSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadEvent">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Reads all the events in an entity</wsdl:documentation>
                <wsdl:input message="tns:ReadEventSoapIn"/>
                <wsdl:output message="tns:ReadEventSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="StopRecipe">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Stops a procedural element</wsdl:documentation>
                <wsdl:input message="tns:StopRecipeSoapIn"/>
                <wsdl:output message="tns:StopRecipeSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadReport">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Obtains a report of the execution of a procedural element</wsdl:documentation>
                <wsdl:input message="tns:ReadReportSoapIn"/>
                <wsdl:output message="tns:ReadReportSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadEntityIDs">
                <wsdl:documentation>Reads the IDs of all entities loaded at coordination level</wsdl:documentation>
                <wsdl:input message="tns:ReadEntityIDsSoapIn"/>
                <wsdl:output message="tns:ReadEntityIDsSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadEntityData">
                <wsdl:documentation>Reads the statical data from an entity</wsdl:documentation>
                <wsdl:input message="tns:ReadEntityDataSoapIn"/>
                <wsdl:output message="tns:ReadEntityDataSoapOut"/>
            </wsdl:operation>
        </wsdl:portType>
            <wsp:Policy wsu:Id="UserNameTokenPasswordHashOverSSL">
             <wsp:ExactlyOne>
                <wsp:All>
                    <sp:TransportBinding>
                        <wsp:Policy>
                            <sp:TransportToken>
                                <wsp:Policy>
                                    <sp:HttpsToken>
                                        <wsp:Policy />
                                    </sp:HttpsToken>
                                </wsp:Policy>
                            </sp:TransportToken>
                            <sp:Layout>
                                <wsp:Policy>
                                    <sp:Lax />
                                </wsp:Policy>
                            </sp:Layout>
                            <sp:IncludeTimestamp />
                            <sp:AlgorithmSuite>
                                <wsp:Policy>
                                    <sp:Basic128 />
                                </wsp:Policy>
                            </sp:AlgorithmSuite>
```

```
                    </wsp:Policy>
                </sp:TransportBinding>
                <sp:SupportingTokens>
                    <wsp:Policy>
                        <wsp:SecurityTokenReference >
                            <wsp:Policy>
                                <wsp:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" />
                            </wsp:Policy>
                        </wsp:SecurityTokenReference >
                    </wsp:Policy>
                </sp:SupportingTokens>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsdl:binding name="ManagementServicesSoapBinding" type="tns:ManagementServicesPortType">
        <wsp:PolicyReference URI="#UserNameTokenPasswordHashOverSSL"/>
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="CreateRecipe">
            <soap12:operation soapAction="http://www.water.domain/CreateRecipe" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ReadProceduralIDs">
            <soap12:operation soapAction="http://www.water.domain/ReadProceduralIDs" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="GetProceduralIDs">
            <soap12:operation soapAction="http://www.water.domain/GetProceduralIDs" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Write">
            <soap12:operation soapAction="http://www.water.domain/Write" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Read">
            <soap12:operation soapAction="http://www.water.domain/Read" style="document"/>
```

```
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="ReadStandardHist">
                <soap12:operation soapAction="http://www.water.domain/ReadStandardHist" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="ReadEvent">
                <soap12:operation soapAction="http://www.water.domain/ReadEvent" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="StopRecipe">
                <soap12:operation soapAction="http://www.water.domain/StopRecipe" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="ReadReport">
                <soap12:operation soapAction="http://www.water.domain/ReadReport" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="ReadEntityIDs">
                <soap12:operation soapAction="http://www.water.domain/ReadEntityIDs" soapActionRequired="true"
style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="ReadEntityData">
                <soap12:operation soapAction="http://www.water.domain/ReadEntityData" soapActionRequired="true"
style="document"/>
                <wsdl:input>
```

```
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="ManagementServices">
            <wsdl:port name="ManagementServicesPort" binding="tns:ManagementServicesSoapBinding">
                <soap12:address location="http://IP:PORT/ManagementServices"/>
            </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Annex B
(normative)

Subsystem interface with SOAP 1.2

B.1 Overview

This annex outlines the information needed to implement the subsystems interface through Web Services and SOAP technology. This implementation permits the upper control level (where the coordination brokers are located) to retrieve data from the lower control level (where the subsystems are located).

This annex concerns subsystems and coordination brokers adapted to this document. The interface between these architecture elements is addressed by a Web Service:

— Upper control level: the coordination broker acts as a Web Services client and implements all described methods; and

— Lower control level: the subsystem level publishes a Web Service as a server. The subsystem implements all described methods for the control of the irrigation entity that it controls.

These methods allow, among others:

— to execute actions in the irrigation entities;

— to create and read the status of the procedural elements to execute in the subsystem; and

— to obtain reports, standard histories and events.

B.2 Requirements

B.2.1 General

The names of the classes and the methods presented in this annex shall by used for the implementation of the subsystem interface when using SOAP 1.2. Likewise, shall be used the following protocols defined by the Web Services Protocol Stack:

— HTTP 1.1 (Hypertext Transfer Protocol) RFC 2616;

— TLS within HTTP/1.1 RFC 2817;

— TLS 1.2 RFC 5246;

— SOAP 1.2. (Simple Object Access Protocol). The specifications of this protocol are available at http://www.w3.org/TR/soap/;

— WSDL 2.0 (Web Services Description Language). Language of web service's definition, developed by the Web Services Description Working Group. The specifications of this language are available at http://www.w3.org/2002/ws/desc/; and

— XML (Extensible Markup Language). The language of definition of the data contained on the SOAP is at http://www.w3.org/XML/

B.2.2 Data protection

Data protection regulations in the jurisdiction of use can apply.

B.2.3 Security

The SOAP interface described in this document shall use SecurityTokenReference security method, as specified in WS Security 1.1. This token works similarly to the Oauth2 framework. Therefore, a request shall first be made to the authorization server. This authorization server provides a token to access the services described in this document.

The servers shall have a digital certificate for ensure the security through the communication channel defined, using HTTPS over TLS 1.2.

The security criteria defined in this annex shall be understood as minimal, being possible to stablish extra criteria to access data.

B.2.4 Header

The header of all requests and responses shall be encapsulated in an HTTP 1.1 transaction; the HTTP 1.1 header shall include the following attribute as specified by the SOAP 1.2 standard:

— Content-Type: application / soap + xml; Charset = utf-8; action = "http: //www.water.domain/ <op>"

Where <op> is the operation to invoke with the SOAP 1.2 protocol. E.g. to invoke the Read method, the following HTTP header shall be included in the HTTP 1.1 transaction:

— Content-Type: application / soap + xml; Charset = utf-8; action = http: //www.water.domain/Read

B.2.5 Server requirements

The server shall provide the URL (uniform resource locator) to access to the Web Services described in this annex. Additionally, any application located at the lower control level, shall provide the URI (Uniform Resource Identifier) of the server for each Web Service. The structure of the Web Services is the following URI:

— Http://<domain_name>: <port>/SubSystemCommunication

It is possible to retrieve the WSDL describing each web service using the following URI:

— Http://<domain_name>: <port>/SubSystemCommunication?Wsdl

The structure of the authorization server is the following URI:

— Http://<domain_name>:<port>/Security

It is possible to retrieve the WSDL describing each web service using the following URI:

— Http: // < domain_name >: <port>/Security?Wsdl

B.2.6 Web Services Description Language (WSDL) contract

The WSDL is a description of the web service published by the server that establishes a contract with all its possible clients. It specifies the interface through which a client can gain access to the service and details about how it shall be used.

The server and the clients shall fully respect the WSDL described in this annex (even the sequence of the parameters) to ensure communication through the interface. The WSDL is composed by the classes defined in B.3 and B.4.

B.2.7 Classes and enumerations

The published Web Services can be case sensitive for enumerated values. The UpperCamelCase convention shall be used in all classes and enumerations.

B.3 Implementation classes for web services

B.3.1 Implementation criteria

This subclause describes the methods and classes required to implement the subsystem interface between the lower control level (subsystem level) and the upper control level (coordination level), defined at the document architecture clause (see Clause 5). The following subclauses present the structure and attributes of the defined classes.

B.3.2 SubSystemCommunication class

This class contains all defined methods.

Table B.1 — SubSystemCommunication class

Methods			
Name	**Parameters**	**Return**	**Description**
Write	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. PropertyName: PropertyName. Contains one of the values of B.3.22. Value: StringValue. Value of the property to be established.	WriteResponse	This method allows the writing of a property of an irrigation entity. This method in only valid for properties that can be modified.
Read	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. PropertyName: Contains one of the values of B.3.22.	ReadResponse	This method allows the reading of a property of a irrigation entity.

Methods			
Name	**Parameters**	**Return**	**Description**
ReadStandardHist	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. PropertyName: Contains one of the values of B.3.22. Date: DateTimeValue. Date from which the data are requested, in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format. NumHist: IntegerValue. Number of samples contained in the standard history of the requested Date. This parameter can use three values: 24, 48 o 96. Statistics: Statistics (B.3.23 Statistics enumeration).	ReadStandardHistResponse	This method allows the reading of the standard history of a property of an irrigation entity. Selection of the Statistics parameter that defines the result obtained by the method.
SubscribeEvent	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. Path: StringValue. Path to access to the server where the events shall be sent.	SubscribeEventResponse	This method allows to subscribe the coordination broker to the events occurred in an entity.
UnsusbscribeEvent	ActionID: StringValue. ID of the action. EntityID: StringValue ID of the entity where the action shall be executed.	UnsubscribeEventResponse	This method allows to unsubscribe the coordination broker to the events occurred in an entity.
CreateRecipe	ActionID: StringValue. ID of the action. Operation: Operation	CreateRecipeResponse	This method allows creating a procedural element (procedure, unit procedure or operation). All the procedural element parameters are contained in the Operation class.

Methods			
Name	**Parameters**	**Return**	**Description**
StopRecipe	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. ProceduralID: StringValue. ID of the element of the procedural model where the user wishes to execute the action.	StopRecipeResponse	This method allows stopping the procedural element identified by its ProceduralID. It also requires identifying the irrigation entity.
ReadReport	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. ProceduralID: StringValue. ID of the procedural element where the action shall be executed. InstanceNumber: IntegerValue. Number of the instance for which the report is required.	ReadReportResponse	This method allows obtaining the report of a procedural element using its ProceduralID. It also requires identifying the irrigation entity.
ReadProceduralIDs	ActionID: StringValue. ID of the action. EntityID: StringValue. ID of the entity where the action shall be executed. Date: DateTimeValue. Date from which the data are requested in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format. Default value: current day. Status: Status.	ReadProceduralIDsResponse	This method allows obtainingthe list of ProceduralIDs loaded at the subsystem level. If Date parameter is empty, the default value is the current day. If Status parameter is empty, the default value is “Running”.

B.3.3 Response class

Table B.2 — Response class

Parameter	Type	Description
ResponseCode	IntegerValue	Returns an IntegerValue, according to the list below: 0, 1, 2, 3, 4, 5, 6
ResponseMessage	StringValue	Returns the message explaining the ResponseCode. 0 - Action successfully executed 1 - Execution error 2 - Lexical error 3 - Not supported 4 - Communication error 6 - Unavailable The minimum message shall be one of the listed above. The message can be extended (using a hyphen) with as many explanatory texts as errors can be discriminated by a subsystem or a coordination broker.

The value 6 is only applicable to Read and ReadStandardHist methods.

B.3.4 WriteResponse class

Table B.3 — WriteResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.

B.3.5 ReadResponse class

Table B.4 — ReadResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.
Value	StringValue	Current value of the property requested. The value respects the format defined for the property.

B.3.6 ReadStandardHistResponse class

Table B.5 — ReadStandardHistResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.
HistStandardValues	StringValue	Contains a string of fields separated by commas with the value corresponding to the Statistic and NumHist parameters requested. Each piece of data contains two fields: the value and its timestamp of the value at source, in UTC ISO8601

		format YYYYMMDDhhmmss±hhmm.

B.3.7 ReadReportResponse class

Table B.6 — ReadReportResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.
Report	StringValue	Contains the report of the procedural element according to the data existing at the time of the request. The structure of the report shall conform to what is defined for each type of procedural element. Regardless of the state of the report (complete or partial), the structure shall be maintained by returning with null value all the fields that have no value at the time of the request.
Status	Status	Contains one of the values of B.3.16.

B.3.8 SubscribeEventResponse class

Table B.7 — SubscribeEventResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.

B.3.9 UnsubscribeEventResponse class

Table B.8 — UnsubscribeEventResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.

B.3.10 CreateRecipeResponse class

Table B.9 — CreateRecipeResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.

B.3.11 StopRecipeResponse class

Table B.10 — StopRecipeResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.

B.3.12 Operation class

Table B.11 — Operation class

Parameter	Type	Description
ProceduralID	StringValue	Contains the ID of the procedural element that is being to be created.
EntityID	StringValue	Contains the ID of the entity where the procedural element is going to be executed.
EntityType	EntityType	Contains one of the values of B.3.25.
RecipeType	RecipeType	Contains one of the values of B.3.17.
OperationParameters	ArrayOfOperationParameter	Array of parameters of the procedural element.

B.3.13 OperationParameter class

Table B.12 — OperationParameter class

Parameter	Type	Description
RecipeParameter	RecipeParameterType	Contains one of the values of B.3.18.
MonitoringParameterName	MonitoringParameterType	Contains one of the values of B.3.19.
RepetitionParameterName	RepetitionParameterType	Contains one of the values of B.3.20.
WeekCalendarParameterName	WeekCalendarParameterType	Contains one of the values of B.3.21.
ParameterType	ParameterType	Contains one of the values of B.3.15.
ParameterValue	StringValue	Value for the parameter, according to ParameterType format.

B.3.14 ReadProceduralIDsResponse class

Table B.13 — ReadProceduralIDsResponse class

Parameter	Type	Description
ActionID	StringValue	Contains the ActionID that has been introduced on the request.
Response	Response	Contains one of the values of B.3.3.
ProceduralIDs	StringValue	Contains the ProceduralIDs that meet with the parameterization of the request, separated by commas.

B.3.15 ParameterType enumeration

Table B.14 contains the data type that can be contained in a parameter.

Table B.14 — ParameterType enumeration

Value	Description
StringValue	Plain text StringValue.
IntegerValue	Integer number.
DecimalValue	Real number, with dot as a decimal separator.
DateTimeValue	Date in YYYYMMDDhhmmss±hhmm UTC ISO 8601 format.
BooleanValue	1 if true, 0 if false.

B.3.16 Status enumeration

Table B.15 contains the list of possible values of the parameter.

Table B.15 — Status enumeration

Value	Description
Idle	The procedural element execution is pending.
Running	The procedural element is being executed.
Completed	The procedural element is finished.
Stopped	The procedural element has been halted.
Unknown	The status of the procedural element is unknown because the status of the irrigation entity where it is executed is unknown.

B.3.17 RecipeType enumeration

Table B.16 contains a value from the enumeration, that can be extended with new values.

Table B.16 — RecipeType enumeration

RecipeType	Description	Applicable to
IrrigationRecipe1	Supply of water (or irrigation) by time or volume.	HYS, IHG/VIH and SSS
IrrigationRecipe2	Supply of water (or irrigation) without end condition.	HYS
IrrigationRecipe3	Supply of water (or irrigation) based on a calendar and executions by time or volume.	HYS, IHG/VIH and SSS
IrrigationRecipe4	Supply of water (or irrigation) based on thresholds.	HYS and SSS
PumpingRecipe1	Pumping of irrigation water according to a flow rate set point.	PMS
PumpingRecipe2	Pumping of irrigation water according to flow rate and level set points, typically related to RSV.	PMS
PumpingRecipe3	Pumping of irrigation water according to a pressure set point, typically required by a INT.	PMS

RecipeType	Description	Applicable to
PumpingRecipe4	Pumping of irrigation wate according to flow rate and pressure set points, typically required by a INT.	PMS
PumpingRecipe5	Pumping of irrigation water attending to time habilitation.	PMS
NetworkBranchRecipe1	Programming of the NBU Operation, attending to the programmed Operations in the HYSs and HYGs contained.	NBU
ControlPointRecipe1	Programming of the NCP Operation, attending to the programmed Operations in its successor entities.	NCP
ControlPointRecipe2	Programming of a control point to provide specific flow rate or pressure to the irrigation water.	NCP
ControlPointRecipe3	Programming of a control point to monitorize the water input and/or water output.	NCP
FertigationRecipe1	Addition of different fertilizers to the irrigation water by injection of a proportional quantity of fertilizer to a volume of irrigation water.	FTS
FertigationRecipe2	Addition of different fertilizers to the irrigation water by injection of a proportional quantity of fertilizer throughout the Operation.	FTS
FertigationRecipe3	Addition of different fertilizers to the irrigation water by injection of a bulk of fertilizer based on a time criteria during the Operation.	FTS
FertigationRecipe4	Addition of different fertilizers to the irrigation water by injection of a bulk of fertilizer based on a volume criterion during the Operation.	FTS
FertigationRecipe5	Addition of a premixed fertilizer to the irrigation water.	FTS
FiltrationRecipe1	Physical treatment of irrigation water to reduce the amount of solid particles present.	WSF
IrrigationNetworkUnitProcedure1	Procedure of water delivering to one or more recipients, guaranteeing the pressure and flow requirements.	INT
SolidSetUnitProcedure1	Procedure for water delivering to one or more irrigation blocks, guaranteeing the pressure and flow requirements.	SSS
PumpingUnitProcedure1	Procedure for pumping irrigation water to provide flow with the right pressure conditions.	PMS
ReservoirUnitProcedure1	Procedure for monitoring the storagement of irrigation water.	RSV
FiltrationUnitProcedure1	Procedure for irrigation water filtration.	WSF
IrrigationSectorProcedure1	General control of the main process to be performed (to irrigate), based on the activities developed in the irrigation entities (and its dependences) composing the irrigation sector.	IRA

B.3.18 RecipeParameterType enumeration

Table B.17 contains the list of possible parameter type values for an operation recipe, according to the procedural model parameter types defined on this document.

Table B.17 — RecipeParameterType enumeration

Parameter	Type	Description
StartDate	DateTimeValue	Date/time consigned for the start of the procedural element. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.
MaxDuration	IntegerValue	Maximum time of duration of the procedural element. Expressed in seconds. No default value, its inclusion is required.
MaxVolume	DecimalValue	Maximum volume during the procedural element. Expressed in m^3. No default value.
OpeningDegree	IntegerValue	Opening degree of the entity, expressed as a percentage. Dead zone ±5. Default value 100.
MonCummulativeVolumeOut	BooleanValue	Activate/deactivate the monitoring of the output volume. Default value 0.
MonInternalFlowOut	BooleanValue	Activate/deactivate the monitoring of the output flow. Default value 0.
MonPressureOut	BooleanValue	Activate/deactivate the monitoring of the output pressure. Default value 0.
MonCumulativeVolumeIn	BooleanValue	Activate/deactivate the monitoring of the input volume. Default value 0.
MonInternalFlowIn	BooleanValue	Activate/deactivate the monitoring of the input flow. Default value 0.
MonPressureIn	BooleanValue	Activate/deactivate the monitoring of the input pressure. Default value 0.
Repetition	BooleanValue	Activate/deactivate the recipe repetition. Default value 0.
WeekCalendar	StringValue	Contains 7 fields, representing a week (first field is for Monday and the last for Sunday according to ISO 8601). Each field can take one of the next values: — Value=1, when the operation should be executed in that day. — Value=0, when the operation should not be executed in that day. Without default values.
InternalFlowOutSetPoint	DecimalValue	Flow value, with a decimal point, to maintain during the procedural element execution, in m^3/s.
SourceLevel_InternalFlowOutSetPoint	StringValue	Level-FlowRate values to accomplish during the procedural element execution. The parameter contains one or more Level-FlowRate relations where: — Value 1: Is the minimum level required at source to use the FlowRate setpoint, expressed in metres. — Value 2: Is the flow rate setpoint to follow, expressed in m^3/s. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.

Parameter	Type	Description
DestinationLevel_InternalFlowOutSetPoint	StringValue	Level-FlowRate values to accomplish during the procedural element execution. The parameter contains one or more Level-FlowRate relations where: — Value 1: Is the minimum level required at destination to use the FlowRate setpoint, expressed in metres. — Value 2: Is the flow rate setpoint to follow, expressed in m^3/s. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.
PressureOutSetPoint	DecimalValue	Pressure value, with a decimal point, to maintain during the procedural element execution. Expressed in Pa.
InternalFlowOutSetPoint_PressureOutSetPoint	StringValue	FlowRate-Pressure values to accomplish during the procedural element execution. The parameter contains one or more FlowRate-Pressure relations where: — Value 1: Is the flow rate required for a certain value 2, expressed in m^3/s. — Value 2: Is the pressure setpoint to follow for the corresponding value 1, expressed in Pa. The value 1 shall be measured to define which value 2 is used as setpoint. Each pair of values is separated by commas. There is no limitation to the number of pairs included in the parameter.
MaxRainfall	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum rainfall, with a decimal point, that stops the Operation during its execution. Expressed in $m^3/m^2/s$. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MaxWindSpeed	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum wind speed, with a decimal point, obtained from the WindParameters property that stops the Operation during its execution. Expressed in m/s. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MinSoilWaterContent	StringValue	Contains 2 fields separated by commas: — Field 1. Minimum value of the WaterSoilContent property, with a decimal point, that starts the Operation. Expressed in %. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.

Parameter	Type	Description
MaxSoilWaterContent	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum value of the WaterSoilContent, with a decimal point, that stops the Operation during its execution. Expressed in %. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MinSoilWaterPotential	StringValue	Contains 2 fields separated by commas: — Field 1. Minimum value of the WaterSoilPotential property, with a decimal point, that starts the Operation. Expressed in Pa. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MaxSoilWaterPotential	StringValue	Contains 2 fields separated by commas: — Field 1. Maximum value of the WaterSoilPotential, with a decimal point, that stops the Operation during its execution. Expressed in Pa. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
SolarRadiationTriggerOn	StringValue	Contains 2 fields separated by commas: — Field 1. Value of the SolarRadiation property, with a decimal point, that starts the Operation. Expressed in W/m^2. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
SolarRadiationTriggerOff	StringValue	Contains 2 fields separated by commas: — Field 1. Value of the SolarRadiation property, with a decimal point, that stops the Operation during its execution. Expressed in W/m^2. No default value. — Field 2. Time in seconds where the field 1 shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
MaxLevel	DecimalValue	Level setpoint which determines that the entity is at full capacity for the closing of its water inputs. With a decimal point and expressed in m.
MinLevel	DecimalValue	Level setpoint which determines that the entity is at empty capacity for the closing of its water outputs. With a decimal point and expressed in m.
MaxTime	IntegerValue	Maximum time between filter flushing, in s.
DifferentialPressure	DecimalValue	Pressure difference, measured between filter inlet and outlet, which activates the filter flushing. Expressed in Pa. No default value.

Parameter	Type	Description
QuantityRatioSetPoint	StringValue	String composed of one or more groups of four fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. The fields 3 and 4 of each group are mutually exclusive: may both be empty but not filled in simultaneously. Each group has: — Value1: DSM number, expressed as an integer. — Value2: quantity of fertilizer, expressed in m^3, to be proportionally dosified per volume unit of irrigation water and directly related to the total volume to be fertilized (MaxVolume). — Value3: pH setpoint for dosifying adjust, expressed as a real. — Value4: electrical conductivity for dosifying adjust, expressed as a real. The value2 can be adapted attending to value3 or value4 setpoint.
TimeRatioSetPoint	StringValue	String composed of one or more groups of four fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. The fields 3 and 4 of each group are mutually exclusive: may both be empty but not filled in simultaneously. Each group has: — Value1: DSM number, expressed as an integer. — Value2: quantity of time, expressed in seconds, during which the fertilizer is going to be proportionally dosified until the Operation reaches its end by time (MaxDuration). — Value3: pH setpoint for dosifying adjust, expressed as a real. — Value4: electrical conductivity for dosifying adjust, expressed as a real. The value2 can be adapted attending to value3 or value4 setpoint.
pHSetPoint	DecimalValue	Acidity value, with a decimal point, to maintain during the Operation execution.
ECSetPoint	DecimalValue	Electrical conductivity value, with a decimal point, to maintain during the Operation execution, in S/m.
FertigationTime	StringValue	String composed of one or more groups of two fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. Each group has: — Value1: DSM number, expressed as an integer. — Value2: Time to apply the fertilizer bulk, expressed in s.

Parameter	Type	Description
FertigationVolume	StringValue	String composed of one or more groups of two fields separated by commas. Each group corresponds to an injector parameterization. The string can contain as many groups as DSM has the entity. Each group has: — Value1: DSM number, expressed as an integer. — Value2: Volume of fertilizer to apply as fertigation bulk, expressed in m^3.

B.3.19 MonitoringParameterType enumeration

Table B.18 contains a list of parameters to introduce when a monitoring is enabled.

Table B.18 — MonitoringParameterType enumeration

Parameter	Type	Description
ThresholdHH	DecimalValue	Very high threshold value of the variable for which monitoring is activated. When the ThresholdHH is exceeded, an event is generated. This is a required parameter when MonCummulativeVolumeOut and/or MonCummulativeVolumeIn are activated. This parameter can also be required when the ThresholdLL are not established and MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn are activated.
ThresholdLL	DecimalValue	Very low threshold value of the variable for which monitoring is activated. When the ThresholdLL is exceeded, an event is generated. This parameter is required when the ThresholdHH are not established and MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn are activated.
CmdHH	BooleanValue	Its activation forces the Operation to stop when ThresholdHH is reached during the configured time. Parameter conditioned by ThresholdHH. Default value 0. In case this parameter is used with MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn it shall be used with TimeHH.
CmdLL	BooleanValue	Its activation forces the Operation to stop when ThresholdLL is reached during the configured time. Parameter conditioned by ThresholdLL. Default value 0. In case this parameter is used with MonInternalFlowOut and/or MonPressureOut and/or MonInternalFlowIn and/or MonPressureIn it shall be used with TimeLL.
TimeHH	IntegerValue	Time in seconds where the ThresholdHH shall be exceeded. Parameter conditioned by ThresholdHH. Default value 30 s.
TimeLL	IntegerValue	Time in seconds where the ThresholdLL shall be exceeded. Parameter conditioned by ThresholdLL. Default value 30 s.

B.3.20 RepetitionParameterType enumeration

Table B.19 contains a list of parameters to introduce when the Repetition parameter is enabled.

Table B.19 — RepetitionParameterType enumeration

Parameter	Type	Description
RepetitionInterval	IntegerValue	Time interval between repetitions, expressed in seconds and added to the StartDate parameter of the last execution.
RepetitionNumber	IntegerValue	Number of times that the RepetitionInterval parameter shall be repeated. Without limitation.

B.3.21 WeekCalendarParameterType enumeration

Table B.20 contains a list of parameters to introduce when the WeekCalendar is enabled.

Table B.20 — WeekCalendarParameterType enumeration

Parameter	Type	Description
FinalDate	DateTimeValue	Date/time consigned for the end of the procedural element. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm. No default value, its inclusion is required.

B.3.22 PropertyName enumeration

Table B.21 contains the list of properties defined in this document for the irrigation entities.

Table B.21 — PropertyName enumeration

Value	Description
ActivityStatus	Entity activity status
SystemStatus	Subsystem status
Mode	Entity functioning mode
CumulativeVolumeOut	Entity cumulative output volume
InternalFlowOut	Entity output flow
PressureOut	Entity output pressure
OpeningDegree	Entity opening degree
Temperature	Temperature registered at the entity
RelativeHumidity	Relative humidity registered at the entity
SolarRadiation	Solar radiation registered at the entity
WindParameters	Wind speed and direction registered at the entity
Rainfall	Rainfall registered at the entity
CumulativeVolumeIn	Entity cumulative input volume
InternalFlowIn	Entity input flow
PressureIn	Entity input pressure
InstantPower	Entity absorbed instant power during the execution of its functionalities
EnergyConsumption	Entity energy consumption
EntityPerformance	Entity hydraulic performance
PowerOnTime	Entity accumulated activity time
Capacity	Water contained by the entity
SoilWaterContent	Water contained by the soil in a reference area

Value	Description
SoilWaterPotential	Energy state of water in the soil in a reference area
SoilTemperature	Temperature in the soil in a reference area
SoilConductivity	Conductivity in the soil in a reference area
DifferentialPressure	Differential pressure registered between the input and output of the entity
WaterAcidity	Acidity of the irrigation water registered in the entity
WaterConductivity	Electrical conductivity of the irrigation water registered in the entity

B.3.23 Statistics enumeration

Table B.22 contains the type of statistical for the calculation of the standard history. The enumeration contains the following listed values.

Table B.22 — Statistics enumeration

Value	Description
Last	Returns the last value registered for each time slot of a standard history.
Average	Returns the average value registered for each time slot of a standard history.
Maximum	Returns the maximum value registered for each time slot of a standard history.
Minimum	Returns the minimum value registered for each time slot of a standard history.

B.3.24 EntityLevel enumeration

Table B.23 contains the levels of the physical model to which an irrigation entity can belong.

Table B.23 — EntityLevel enumeration

Value
Empty
ProcessCell
Unit
EquipmentModule

B.3.25 EntityType enumeration

Table B.24 contains the listed entities included in the physical model.

Table B.24 — EntityType enumeration

List of values				
IRA	PMS	INT	RSV	SSS
PML	HYS	SSB	NCP	NBU
IHG	VIH	MIH	WFS	FTL
FTS	DSM	DLT	FRT	Empty

B.4 Implementation classes for authorization server

B.4.1 ManagementSecurity class

Table B.25 — ManagementSecurity class

METHODS			
Name	**Name**	**Name**	**Name**
GetToken	grant_type: StringValue. This parameter shall have the next fixed value: "Password". username: StringValue. User to be authenticated by the server. password: StringValue. Password related to the user to be authenticated. scope: StringValue. Optional parameter to define limitations for the requested token, for example for read only operations.	TokenResponse	This method allows the obtaining of the token to access the different services.
RefreshToken	grant_type: StringValue. This parameter shall have the next fixed value: "Refresh_token". refresh_token: StringValue. Value received in the TokenResponse request as Refresh_token value. scope: StringValue. Optional parameter to define limitations for the requested token, for example for read only operations.	TokenResponse	This method allows the token to be updated if it has expired.

B.4.2 TokenResponse class

Table B.26 — TokenResponse class

Parameter	Type	Description
access_token	StringValue	Token that shall be added to requests in order to access services.
token_type	StringValue	This parameter shall have the next fixed value: "Bearer", corresponding to the token type to be used.
expires_in	StringValue	Validity time of the token provided, expressed in s.
refresh_token	StringValue	When the expiration time has been reached, if the token is used an error message shall be received. This value can be used to avoid this error and receive a new token.
scope	StringValue	The same value included in the request.

B.5 WSDL

```
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://www.water.domain" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.water.domain">
    <wsdl:types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://www.water.domain">
            <s:simpleType name="PropertyName">
                <s:restriction base="s:string">
                    <s:enumeration value="ActivityStatus"/>
                    <s:enumeration value="SystemStatus"/>
                    <s:enumeration value="Mode"/>
                    <s:enumeration value="CumulativeVolumeOut"/>
                    <s:enumeration value="InternalFlowOut"/>
                    <s:enumeration value="PressureOut"/>
                    <s:enumeration value="OpeningDegree"/>
                    <s:enumeration value="Temperature"/>
                    <s:enumeration value="RelativeHumidity"/>
                    <s:enumeration value="SolarRadiation"/>
                    <s:enumeration value="WindParameters"/>
                    <s:enumeration value="Rainfall"/>
                    <s:enumeration value="CumulativeVolumeIn"/>
                    <s:enumeration value="InternalFlowIn"/>
                    <s:enumeration value="PressureIn"/>
                    <s:enumeration value="InstantPower"/>
                    <s:enumeration value="EnergyConsumption"/>
                    <s:enumeration value="EntityPerformance"/>
                    <s:enumeration value="PowerOnTime"/>
                    <s:enumeration value="Capacity"/>
                    <s:enumeration value="SoilWaterContent"/>
                    <s:enumeration value="SoilWaterPotential"/>
                    <s:enumeration value="SoilTemperature"/>
                    <s:enumeration value="SoilConductivity"/>
                    <s:enumeration value="DifferentialPressure"/>
                    <s:enumeration value="WaterAcidity"/>
                    <s:enumeration value="WaterConductivity"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="RecipeType">
                <s:restriction base="s:string">
                    <s:enumeration value="IrrigationRecipe1"/>
                    <s:enumeration value="IrrigationRecipe2"/>
                    <s:enumeration value="IrrigationRecipe3"/>
                    <s:enumeration value="IrrigationRecipe4"/>
                    <s:enumeration value="PumpingRecipe1"/>
                    <s:enumeration value="PumpingRecipe2"/>
                    <s:enumeration value="PumpingRecipe3"/>
                    <s:enumeration value="PumpingRecipe4"/>
                    <s:enumeration value="PumpingRecipe5"/>
                    <s:enumeration value="NetworkBranchRecipe1"/>
                    <s:enumeration value="ControlPointRecipe1"/>
                    <s:enumeration value="ControlPointRecipe2"/>
                    <s:enumeration value="ControlPointRecipe3"/>
```

```
                    <s:enumeration value="FertigationRecipe1"/>
                    <s:enumeration value="FertigationRecipe2"/>
                    <s:enumeration value="FertigationRecipe3"/>
                    <s:enumeration value="FertigationRecipe4"/>
                    <s:enumeration value="FertigationRecipe5"/>
                    <s:enumeration value="FiltrationRecipe1"/>
                    <s:enumeration value="IrrigationNetworkUnitProcedure1"/>
                    <s:enumeration value="SolidSetUnitProcedure1"/>
                    <s:enumeration value="PumpingUnitProcedure1"/>
                    <s:enumeration value="ReservoirUnitProcedure1"/>
                    <s:enumeration value="FiltrationUnitProcedure1"/>
                    <s:enumeration value="FertigationUnitProcedure1"/>
                    <s:enumeration value="IrrigationSectorProcedure1"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="EntityLevel">
                <s:restriction base="s:string">
                    <s:enumeration value="Empty"/>
                    <s:enumeration value="Area"/>
                    <s:enumeration value="ProcessCell"/>
                    <s:enumeration value="Unit"/>
                    <s:enumeration value="EquipmentModule"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="EntityType">
                <s:restriction base="s:string">
                    <s:enumeration value="IRA"/>
                    <s:enumeration value="PMS"/>
                    <s:enumeration value="RSV"/>
                    <s:enumeration value="INT"/>
                    <s:enumeration value="SSS"/>
                    <s:enumeration value="PML"/>
                    <s:enumeration value="HYS"/>
                    <s:enumeration value="SSB"/>
                    <s:enumeration value="NCP"/>
                    <s:enumeration value="VIH"/>
                    <s:enumeration value="IHG"/>
                    <s:enumeration value="NBU"/>
                    <s:enumeration value="MIH"/>
                    <s:enumeration value="WFS"/>
                    <s:enumeration value="FTL"/>
                    <s:enumeration value="FTS"/>
                    <s:enumeration value="DSM"/>
                    <s:enumeration value="DLT"/>
                    <s:enumeration value="FRT"/>
                    <s:enumeration value="Empty"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="MonitoringParameterType">
                <s:restriction base="s:string">
                    <s:enumeration value="ThresholdHH"/>
                    <s:enumeration value="CmdHH"/>
                    <s:enumeration value="TimeHH"/>
                    <s:enumeration value="ThresholdLL"/>
                    <s:enumeration value="TimeLL"/>
                    <s:enumeration value="CmdLL"/>
                </s:restriction>
```

```
        </s:simpleType>
        <s:simpleType name="RepetitionParameterType">
            <s:restriction base="s:string">
                <s:enumeration value="RepetitionInterval"/>
                <s:enumeration value="RepetitionNumber"/>
            </s:restriction>
        </s:simpleType>

        <s:simpleType name="WeekCalendarParameterType">
            <s:restriction base="s:string">
                <s:enumeration value="FinalDate"/>
            </s:restriction>
        </s:simpleType>
        <s:simpleType name="RecipeParameterType">
            <s:restriction base="s:string">
                <s:enumeration value="StartDate"/>
                <s:enumeration value="MaxDuration"/>
                <s:enumeration value="MaxVolume"/>
                <s:enumeration value="OpeningDegree"/>
                <s:enumeration value="MonCumulativeVolumeOut"/>
                <s:enumeration value="MonInternalFlowOut"/>
                <s:enumeration value="MonPressureOut"/>
                <s:enumeration value="MonCumulativeVolumeIn"/>
                <s:enumeration value="MonInternalFlowIn"/>
                <s:enumeration value="MonPressureIn"/>
                <s:enumeration value="Repetition"/>
                <s:enumeration value="WeekCalendar"/>
                <s:enumeration value="InternalFlowOutSetPoint"/>
                <s:enumeration value="SourceLevel_InternalFlowOutSetPoint"/>
                <s:enumeration value="DestinationLevel_InternalFlowOutSetPoint"/>
                <s:enumeration value="InternalFlowOutSetPoint_PressureOutSetPoint"/>
                <s:enumeration value="PressureOutSetPoint"/>
                <s:enumeration value="MaxRainfall"/>
                <s:enumeration value="MaxWindSpeed"/>
                <s:enumeration value="MinSoilWaterContent"/>
                <s:enumeration value="MaxSoilWaterContent"/>
                <s:enumeration value="MinSoilWaterPotential"/>
                <s:enumeration value="MaxSoilWaterPotential"/>
                <s:enumeration value="SolarRadiationTriggerOn"/>
                <s:enumeration value="SolarRadiationTriggerOff"/>
                <s:enumeration value="MaxLevel"/>
                <s:enumeration value="MinLevel"/>
                <s:enumeration value="MaxTime"/>
                <s:enumeration value="DifferentialPressure"/>
                <s:enumeration value="QuantityRatioSetPoint"/>
                <s:enumeration value="pHSetPoint"/>
                <s:enumeration value="ECSetPoint"/>
                <s:enumeration value="TimeRatioSetPoint"/>
                <s:enumeration value="FertigationTime"/>
                <s:enumeration value="FertigationVolume"/>
            </s:restriction>
        </s:simpleType>
        <s:simpleType name="ParameterType">
            <s:restriction base="s:string">
                <s:enumeration value="StringValue"/>
                <s:enumeration value="IntegerValue"/>
                <s:enumeration value="DecimalValue"/>
```

```
                    <s:enumeration value="DateTimeValue"/>
                    <s:enumeration value="BooleanValue"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="Status">
                <s:restriction base="s:string">
                    <s:enumeration value="Idle"/>
                    <s:enumeration value="Running"/>
                    <s:enumeration value="Stopped"/>
                    <s:enumeration value="Completed"/>
                    <s:enumeration value="Unknown"/>
                </s:restriction>
            </s:simpleType>
            <s:simpleType name="Statistics">
                <s:restriction base="s:string">
                    <s:enumeration value="Last"/>
                    <s:enumeration value="Average"/>
                    <s:enumeration value="Maximum"/>
                    <s:enumeration value="Minimum"/>
                </s:restriction>
            </s:simpleType>
            <s:complexType name="Operation">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="ProceduralID" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="EntityID" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="EntityType" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="RecipeType" type="tns:RecipeType"/>
                    <s:element minOccurs="1" maxOccurs="1" name="OperationParameters"
type="tns:ArrayOfOperationParameter"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ArrayOfMonitoringParameter">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="unbounded" name="MonitoringParameter" nillable="true"
type="tns:MonitoringParameter"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="MonitoringParameter">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="MonitoringParameterName"
type="tns:MonitoringParameterType"/>
                    <s:element minOccurs="1" maxOccurs="1" name="MonitoringParameterValue" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="MonitoringParameterType" type="tns:ParameterType"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ArrayOfRepetitionParameter">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="2" name="RepetitionParameter" nillable="true"
type="tns:RepetitionParameter"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="RepetitionParameter">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="RepetitionParameterName"
type="tns:RepetitionParameterType"/>
                    <s:element minOccurs="1" maxOccurs="1" name="RepetitionParameterValue" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="RepetitionParameterType"  type="tns:ParameterType"/>
```

```
                </s:sequence>
            </s:complexType>
            <s:complexType name="WeekCalendarParameter">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="WeekCalendarParameterName"
type="tns:WeekCalendarParameterType"/>
                    <s:element minOccurs="1" maxOccurs="1" name="WeekCalendarParameterValue"  type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="WeekCalendarParameterType"   type="tns:ParameterType"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ArrayOfOperationParameter">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="unbounded" name="OperationParameter" nillable="true"
type="tns:OperationParameter"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="OperationParameter">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="RecipeParameter" type="tns:RecipeParameterType"/>
                    <s:element minOccurs="1" maxOccurs="1" name="ParameterValue" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="ParameterType" type="tns:ParameterType"/>
                    <s:element minOccurs="0" maxOccurs="1" name="MonitoringParameters"
type="tns:ArrayOfMonitoringParameter"/>
                    <s:element minOccurs="0" maxOccurs="1" name="RepetitionParameters"
type="tns:ArrayOfRepetitionParameter"/>
                    <s:element minOccurs="0" maxOccurs="1" name="WeekCalendarParameter"
type="tns:WeekCalendarParameter"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="Response">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="ResponseCode" type="s:int"/>
                    <s:element minOccurs="0" maxOccurs="1" name="ResponseMessage" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:element name="CreateRecipe">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="ActionID" type="s:string"/>
                        <s:element minOccurs="1" maxOccurs="1" name="Operation" type="tns:Operation"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="CreateRecipeResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="CreateRecipeResponse" type="tns:CreateRecipeResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="CreateRecipeResult">
                <s:sequence>
                    <s:element name="ActionID" type="s:string" minOccurs="1" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" minOccurs="1" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="ReadProceduralIDs">
```

```
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="Date" type="s:string" maxOccurs="1"/>
                    <s:element name="Status" type="tns:Status" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadProceduralIDsResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadProceduralIDsResponse" type="tns:ReadProceduralIDsResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadProceduralIDsResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="ProceduralIDs" type="s:string" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="Write">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="PropertyName" type="s:string" maxOccurs="1"/>
                    <s:element name="Value" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="WriteResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="WriteResponse" type="tns:WriteResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="WriteResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="Read">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="PropertyName" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadResponse">
            <s:complexType>
```

```
                <s:sequence>
                    <s:element name="ReadResponse" type="tns:ReadResult" minOccurs="0" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="Value" type="s:string" maxOccurs="1"/>
                <s:element name="TimeStamp" type="s:string" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="ReadStandardHist">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="PropertyName" type="s:string" maxOccurs="1"/>
                    <s:element name="Date" type="s:string" maxOccurs="1"/>
                    <s:element name="NumHist" type="s:int" maxOccurs="1"/>
                    <s:element name="Statistics" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="ReadStandardHistResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="ReadStandardHistResponse" type="tns:ReadStandardHistResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ReadStandardHistResult">
            <s:sequence>
                <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                <s:element name="HistStandardValues" type="s:string" maxOccurs="1"/>
            </s:sequence>
        </s:complexType>
        <s:element name="StopRecipe">
            <s:complexType>
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    <s:element name="ProceduralID" type="s:string" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="StopRecipeResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="StopRecipeResponse" type="tns:StopRecipeResult" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="StopRecipeResult">
            <s:sequence>
```

```
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="ReadReport">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                        <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                        <s:element name="ProceduralID" type="s:string" maxOccurs="1"/>
                        <s:element name="InstanceNumber" type="s:int" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="ReadReportResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ReadReportResponse" type="tns:ReadReportResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="ReadReportResult">
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                    <s:element name="Report" type="s:string" maxOccurs="1"/>
                    <s:element name="Status" type="tns:Status" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="SubscribeEvent">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                        <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                        <s:element name="Path" type="s:string" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="SubscribeEventResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="SubscribeEventResponse" type="tns:SubscribeEventResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="SubscribeEventResult">
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
            <s:element name="UnsubscribeEvent">
                <s:complexType>
                    <s:sequence>
                        <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                        <s:element name="EntityID" type="s:string" maxOccurs="1"/>
                    </s:sequence>
```

```
                    </s:complexType>
                </s:element>
                <s:element name="UnsubscribeEventResponse">
                    <s:complexType>
                        <s:sequence>
                            <s:element name="UnsubscribeEventResponse" type="tns:UnsubscribeEventResult" maxOccurs="1"/>
                        </s:sequence>
                    </s:complexType>
                </s:element>
                <s:complexType name="UnsubscribeEventResult">
                        <s:sequence>
                            <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                            <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                        </s:sequence>
                </s:complexType>
            </s:schema>
        </wsdl:types>
        <wsdl:message name="CreateRecipeSoapIn">
            <wsdl:part name="parameters" element="tns:CreateRecipe"/>
        </wsdl:message>
        <wsdl:message name="CreateRecipeSoapOut">
            <wsdl:part name="parameters" element="tns:CreateRecipeResponse"/>
        </wsdl:message>
        <wsdl:message name="ReadProceduralIDsSoapIn">
            <wsdl:part name="parameters" element="tns:ReadProceduralIDs"/>
        </wsdl:message>
        <wsdl:message name="ReadProceduralIDsSoapOut">
            <wsdl:part name="parameters" element="tns:ReadProceduralIDsResponse"/>
        </wsdl:message>
        <wsdl:message name="WriteSoapIn">
            <wsdl:part name="parameters" element="tns:Write"/>
        </wsdl:message>
        <wsdl:message name="WriteSoapOut">
            <wsdl:part name="parameters" element="tns:WriteResponse"/>
        </wsdl:message>
        <wsdl:message name="ReadSoapIn">
            <wsdl:part name="parameters" element="tns:Read"/>
        </wsdl:message>
        <wsdl:message name="ReadSoapOut">
            <wsdl:part name="parameters" element="tns:ReadResponse"/>
        </wsdl:message>
        <wsdl:message name="ReadStandardHistSoapIn">
            <wsdl:part name="parameters" element="tns:ReadStandardHist"/>
        </wsdl:message>
        <wsdl:message name="ReadStandardHistSoapOut">
            <wsdl:part name="parameters" element="tns:ReadStandardHistResponse"/>
        </wsdl:message>
        <wsdl:message name="StopRecipeSoapIn">
            <wsdl:part name="parameters" element="tns:StopRecipe"/>
        </wsdl:message>
        <wsdl:message name="StopRecipeSoapOut">
            <wsdl:part name="parameters" element="tns:StopRecipeResponse"/>
        </wsdl:message>
        <wsdl:message name="ReadReportSoapIn">
            <wsdl:part name="parameters" element="tns:ReadReport"/>
        </wsdl:message>
        <wsdl:message name="ReadReportSoapOut">
```

```
            <wsdl:part name="parameters" element="tns:ReadReportResponse"/>
        </wsdl:message>
        <wsdl:message name="SubscribeEventSoapIn">
            <wsdl:part name="parameters" element="tns:SubscribeEvent"/>
        </wsdl:message>
        <wsdl:message name="SubscribeEventSoapOut">
            <wsdl:part name="parameters" element="tns:SubscribeEventResponse"/>
        </wsdl:message>
        <wsdl:message name="UnsubscribeEventSoapIn">
            <wsdl:part name="parameters" element="tns:UnsubscribeEvent"/>
        </wsdl:message>
        <wsdl:message name="UnsubscribeEventSoapOut">
            <wsdl:part name="parameters" element="tns:UnsubscribeEventResponse"/>
        </wsdl:message>
        <wsdl:portType name="SubSystemCommunicationSoap">
            <wsdl:operation name="CreateRecipe">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Creates a procedural
element</wsdl:documentation>
                <wsdl:input message="tns:CreateRecipeSoapIn"/>
                <wsdl:output message="tns:CreateRecipeSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadProceduralIDs">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns the operations loaded at subsystem
level</wsdl:documentation>
                <wsdl:input message="tns:ReadProceduralIDsSoapIn"/>
                <wsdl:output message="tns:ReadProceduralIDsSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="Write">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Writes a value for an entity's
property</wsdl:documentation>
                <wsdl:input message="tns:WriteSoapIn"/>
                <wsdl:output message="tns:WriteSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="Read">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Reads the current value of an entity's
property</wsdl:documentation>
                <wsdl:input message="tns:ReadSoapIn"/>
                <wsdl:output message="tns:ReadSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadStandardHist">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Reads the standard history of an entity's property
</wsdl:documentation>
                <wsdl:input message="tns:ReadStandardHistSoapIn"/>
                <wsdl:output message="tns:ReadStandardHistSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="StopRecipe">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Stops a procedural
element</wsdl:documentation>
                <wsdl:input message="tns:StopRecipeSoapIn"/>
                <wsdl:output message="tns:StopRecipeSoapOut"/>
            </wsdl:operation>
            <wsdl:operation name="ReadReport">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Obtains a execution report of a procedural
element</wsdl:documentation>
                <wsdl:input message="tns:ReadReportSoapIn"/>
                <wsdl:output message="tns:ReadReportSoapOut"/>
            </wsdl:operation>
```

```
        <wsdl:operation name="SubscribeEvent">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Subscribes to an entity events</wsdl:documentation>
            <wsdl:input message="tns:SubscribeEventSoapIn"/>
            <wsdl:output message="tns:SubscribeEventSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="UnsubscribeEvent">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Unsubscribes to an entity events</wsdl:documentation>
            <wsdl:input message="tns:UnsubscribeEventSoapIn"/>
            <wsdl:output message="tns:UnsubscribeEventSoapOut"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsp:Policy wsu:Id="UserNameTokenPasswordHashOverSSL">
        <wsp:ExactlyOne>
            <wsp:All>
                <sp:TransportBinding>
                    <wsp:Policy>
                        <sp:TransportToken>
                            <wsp:Policy>
                                <sp:HttpsToken>
                                    <wsp:Policy />
                                </sp:HttpsToken>
                            </wsp:Policy>
                        </sp:TransportToken>
                        <sp:Layout>
                            <wsp:Policy>
                                <sp:Lax />
                            </wsp:Policy>
                        </sp:Layout>
                        <sp:IncludeTimestamp />
                        <sp:AlgorithmSuite>
                            <wsp:Policy>
                                <sp:Basic128 />
                            </wsp:Policy>
                        </sp:AlgorithmSuite>
                    </wsp:Policy>
                </sp:TransportBinding>
                <sp:SupportingTokens>
                    <wsp:Policy>
                        <wsp:SecurityTokenReference >
                            <wsp:Policy>
                                <wsp:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" />
                            </wsp:Policy>
                        </wsp:SecurityTokenReference >
                    </wsp:Policy>
                </sp:SupportingTokens>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsdl:binding name="SubSystemCommunicationSoap12" type="tns:SubSystemCommunicationSoap">
        <wsp:PolicyReference URI="#UserNameTokenPasswordHashOverSSL"/>
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="CreateRecipe">
            <soap12:operation soapAction="http://www.water.domain/CreateRecipe" style="document"/>
```

```
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ReadProceduralIDs">
            <soap12:operation soapAction="http://www.water.domain/ReadProceduralIDs" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Write">
            <soap12:operation soapAction="http://www.water.domain/Write" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Read">
            <soap12:operation soapAction="http://www.water.domain/Read" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ReadStandardHist">
            <soap12:operation soapAction="http://www.water.domain/ReadStandardHist" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="StopRecipe">
            <soap12:operation soapAction="http://www.water.domain/StopRecipe" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ReadReport">
            <soap12:operation soapAction="http://www.water.domain/ReadReport" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
```

```
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="SubscribeEvent">
                <soap12:operation soapAction="http://www.water.domain/SubscribeEvent" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="UnsubscribeEvent">
                <soap12:operation soapAction="http://www.water.domain/UnsubscribeEvent" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="SubSystemCommunication">
            <wsdl:port name="SubSystemCommunicationSoap12" binding="tns:SubSystemCommunicationSoap12">
                <soap12:address location="http://IP:80/SubSystemCommunication"/>
            </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Annex C
(normative)

Event interface with SOAP 1.2

C.1 Overview

This annex outlines the information needed to implement the event interface through Web Services and SOAP technology. This implementation permits the lower control level (where the subsystems are located) to send the events occurred in an irrigation entity to the upper control level (where the coordination brokers are located).

This annex concerns subsystems and coordination brokers adapted to this document. The interface between these architecture elements is addressed by a Web Service:

— Upper control level: the coordination broker acts as a Web Services server and implements all methods described in this annex; and

— Lower control level: the subsystems acts as clients sending to the upper control level the events using the requests described in this annex.

These methods allow the exchange of events with different relevance, being the communication started by the subsystem when detects an event.

C.2 Requirements

C.2.1 General

The names of the classes and the methods presented in this annex shall be used for the implementation the events interface when using SOAP 1.2. Likewise, shall be used the following protocols defined by the Web Services Protocol Stack:

— HTTP 1.1 (Hypertext Transfer Protocol) RFC 2616[5];

— TLS within HTTP/1.1 RFC 2817[6];

— TLS 1.2 RFC 5246[7];

— SOAP 1.2. (Simple Object Access Protocol). The specifications of this protocol are available at http://www.w3.org/TR/soap/[8];

— WSDL 2.0 (Web Services Description Language). Language of web service's definition, developed by the Web Services Description Working Group. The specifications of this language are available at http://www.w3.org/2002/ws/desc/[9]; and

— XML (Extensible Markup Language). The language of definition of the data contained on the SOAP is at http://www.w3.org/XML/[10].

C.2.2 Data protection

C.2.3 Data protection regulations in the jurisdiction of use can apply.Security

The SOAP interface described in this document shall use SecurityTokenReference security method, as specified in WS Security 1.1. This token works similarly to the Oauth2 framework. Therefore a request shall first be made to the authorization server. This authorization server provides a token to access the services described in this document.

The servers shall have a digital certificate for ensure the security through the communication channel defined, using HTTPS over TLS 1.2.

The security criteria defined in this annex shall to be understood as minimal, being possible to establish extra criteria to access data.

C.2.4 Header

The header of all requests and responses shall be encapsulated in an HTTP 1.1 transaction; the HTTP 1.1 header shall include the following attribute as specified by the SOAP 1.2 standard:

— Content-Type: application / soap + xml; Charset = utf-8; action = "http: //www.water.domain/ <op>"

Where <op> is the operation to invoke with the SOAP 1.2 protocol. E.g. to invoke the newEvent , the following HTTP header shall be included in the HTTP 1.1 transaction:

— Content-Type: application / soap + xml; Charset = utf-8; action = http: //www.water.domain/NewEvent

C.2.5 Server requirements

The server shall provide the Uniform Resource Identifier (URI) of the provider server for each Web Services included in this annex. Additionally, any application located at the upper control level, shall provide the URI (Uniform Resource Identifier) of the server for each Web Service. The structure of the Web Services is the following URI:

— Http://<domain_name>: <port>/EventCommunication

It is possible to retrieve the WSDL describing each web service using the following URI:

— Http://<domain_name>:<port>/EventCommunication?Wsdl

The structure of the authorization server is the following URI:

— Http://<domain_name>:<port>/Security

It is possible to retrieve the WSDL describing each web service using the following URI:

— Http: // < domain_name >: <port>/Security?Wsdl

C.2.6 Web Services Description Language (WSDL) contract

The WSDL is a description of the web service published by the server that establishes a contract between the server and the clients. It specifies the interface through which a client can gain access to the service and details about how it shall be used.

The server and the clients shall fully respect the WSDL described in this annex (even the sequence of the parameters) to ensure communication through the interface. The WSDL is composed by the implementation classes defined in C.3 and C.4.

C.2.7 Classes and enumerations

The published Web Services can be case sensitive for enumerated values. The UpperCamelCase convention shall be used in all classes and enumerations.

C.3 Implementation classes for web services

C.3.1 Implementation criteria

This subclause describes the methods and classes required to implement the management interface between the lower control level (subsystem level) and the upper control level (coordination level), defined at the system architecture clause (see Clause 4). The following clauses present the structure and attributes of the defined classes.

C.3.2 EventCommunication class

Table C.1 — EventCommunication class

Methods			
Name	**Parameters**	**Return**	**Description**
NewEvent	ActionID: StringValue. ID of the action. EntityID: StringValue. Contains the ID of the entity where the action shall be executed. EventValue: EventValue	NewEventResponse	This method allows the event sending. The server informs to the client about the event reception.

C.3.3 Response class

Table C.2 — Response class

Parameter	Type	Description
ResponseCode	IntegerValue	Returns an integer, according to the list below: 0, 1, 2, 3, 4
ResponseMessage	StringValue	Returns the message explaining the ResponseCode. 0 - Action successfully executed 1 - Execution error 2 - Lexical error 3 - Not supported 4 - Communication error The minimum message shall be one of the listed above. The message can be extended (using a hyphen) with as many explanatory texts as errors can be discriminated by a subsystem or a coordination broker.

C.3.4 EventValue class

Table C.3 — EventValue class

Parameter	Type	Description
EventID	StringValue	Contains the ID of the occurred event.
EventName	StringValue	Contains the event description.
TimeStamp	StringValue	Date/time when the event was generated. In coordinated universal time (UTC) ISO 8601 format YYYYMMDDhhmmss±hhmm.
Relevance	IntegerValue	Relevance of the event
ProceduralID	StringValue	ID of the element of the procedural model for which the report is required.
InstanceNumber	IntegerValue	Instance of the ProceduralID where the event belongs.

C.3.5 EventName string

The EventName string shall be composed taking into account the values defining each event. The possible strings are included in Table C.4, that can be extended with new values. A subsystem exchanges the events assigned to the entity that it controls.

Table C.4 — EventName string

EventName-Field1	EventName-Field2	EventName-Field3	Description
ActivityStatusChanged	Previous value	Current value	The ActivityStatus property changes its value.
SystemStatusUnknown	Previous value	Current value	The SystemStatus property changes its value to unknown.
ModeChanged	Previous value	Current value	The Mode property changes its value.
InternalFlowOutIsNot0OpeningDegreeIs0	Active/Inactive		Activation/deactivation of flow other than zero with entity closed event.
GeneralFailure	Active/Inactive		Activation/deactivation of a general failure in an unit entity. The entity is not available when the event is active.
Failure	Active/Inactive		Failure in on status in an equipment module entity. The entity is not available when the event is active.
CumulativeVolumeOutSync	Previous value	Current value	The CumulativeVolumeOut property has been synchronized.
MinimumCapacity	Active/Inactive	Current value	Activation/deactivation of minimum level reached on a source entity. Related to Capacity property.
MaximumCapacity	Active/Inactive	Current value	Activation/deactivation of maximum level reached on a destination entity. Related to Capacity property.
CumulativeVolumeIn≠CumulativeVolumeOut			The CumulativeVolumeOut and CumulativeVolumeIn properties, has not the same value (±5 %).

EventName-Field1	EventName-Field2	EventName-Field3	Description
ContinuosCleaning	Active/Inactive		Activation/deactivation of a failure on in filter flushing.
MaxWorkingPressure	Active/Inactive		Activation/deactivation of maximum working pressure detected.
MinWorkingPressure	Active/Inactive		Activation/deactivation of minimum working pressure detected.
AutonomyWarning	Active/Inactive	Remaining autonomy (voltage)	Activation/deactivation of an event: the energy autonomy of the irrigation entity is near to end.
AccessWarning	Active/Inactive		Activation/deactivation of a warning of physical access to the irrigation entity.
SubsystemEvent	Available to define	Available to define	Open event for proprietary events of a subsystem.
AccumulativePropertySync	Previous value	Current value	The value of a accumulative property has been synchronized.
StartedOperation			A procedural element has changed to the status Running.
StoppedOperation			A running or idle procedural element reaches the Stopped status (following the StopRecipe action).
CompletedOperation			A running procedural element reaches the Completed status.
OperationErrorStarting	Active/Inactive		A procedural element was not correctly executed. Although the start condition was reached, the Operation did not start.
OperationErrorEnding	Active/Inactive		A procedural element was not executed correctly. When the end condition was reached, the Operation had not finished.
InternalFlowOutIs0OpeningDegreeIsNot0	Active/Inactive		Alarm of zero flow with irrigation entity open. To generate the event with Inactive value in field 2, is required that previously the field 2 of the event has been in Active.
ThresholdHHTriggeredOn	PropertyName	Current value	The ThresholdHH of a monitoring property has been reached.
ThresholdHHTriggeredOff	PropertyName	Current value	The ThresholdHH of a monitoring property has been deactivated.
ThresholdLLTriggeredOn	PropertyName	Current value	The ThresholdLL of a monitoring property has been reached.
ThresholdLLTriggeredOff	PropertyName	Current value	The ThresholdLL of a monitoring property has been deactivated.
OperationErrorSetPoint			A procedural element was not correctly executed. The subsystem can not reach or maintain the setpoint introduced.
OperationOnRecipient			A procedural element was received by the destination device.

C.4 Implementation classes for authorization server

C.4.1 ManagementSecurity class

Table C.5 — ManagementSecurity class

METHODS			
Name	**Name**	**Name**	**Name**
GetToken	grant_type: StringValue. This parameter shall have the next fixed value: "Password". username: StringValue. User to be authenticated by the server. password: StringValue. Password related to the user to be authenticated. scope: StringValue. Optional parameter to define limitations for the requested token, for example for read only operations.	TokenResponse	This method allows the obtaining of the token to access the different services.
RefreshToken	grant_type: StringValue. This parameter shall have the next fixed value: "Refresh_token". refresh_token: StringValue. Value received in the TokenResponse request as Refresh_token value. scope: StringValue. Optional parameter to define limitations for the requested token, for example for read only operations.	TokenResponse	This method allows the token to be updated if it has expired.

C.4.2 TokenResponse class

Table C.6 — TokenResponse class

Parameter	Type	Description
access_token	StringValue	Token that shall be added to requests in order to access services.
token_type	StringValue	This parameter shall have the next fixed value: "Bearer", corresponding to the token type to be used.
expires_in	StringValue	Validity time of the token provided, expressed in s.
refresh_token	StringValue	When the expiration time has been reached, if the token is used an error message shall be received. This value can be used to avoid this error and receive a new token.
scope	StringValue	The same value included in the request.

C.5 WSDL

```
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://www.water.domain" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:wsp="http://www.w3.org/ns/ws-policy"xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://www.water.domain">
    <wsdl:types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://www.water.domain">
            <s:complexType name="Response">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="ResponseCode" type="s:int"/>
                    <s:element minOccurs="0" maxOccurs="1" name="ResponseMessage" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="EventValue">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="EventID" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="EventName" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="TimeStamp" type="s:string"/>
                    <s:element minOccurs="1" maxOccurs="1" name="Relevance" type="s:integer"/>
                    <s:element minOccurs="1" maxOccurs="1" name="ProceduralID" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:element name="NewEvent">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="ActionID" type="s:string"/>
                        <s:element minOccurs="1" maxOccurs="1" name="EntityID" type="s:string"/>
                        <s:element minOccurs="1" maxOccurs="1" name="EventValue" type="tns:EventValue"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="NewEventResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="NewEventResponse" type="tns:NewEventResult" maxOccurs="1"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="NewEventResult">
                <s:sequence>
                    <s:element name="ActionID" type="s:string" maxOccurs="1"/>
                    <s:element name="Response" type="tns:Response" maxOccurs="1"/>
                </s:sequence>
            </s:complexType>
        </s:schema>
    </wsdl:types>
    <wsdl:message name="NewEventSoapIn">
        <wsdl:part name="parameters" element="tns:NewEvent"/>
    </wsdl:message>
    <wsdl:message name="NewEventSoapOut">
        <wsdl:part name="parameters" element="tns:NewEventResponse"/>
    </wsdl:message>
    <wsdl:portType name="EventCommunicationSoap">
        <wsdl:operation name="NewEvent">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns an event occurred in an entity</wsdl:documentation>
            <wsdl:input message="tns:NewEventSoapIn"/>
            <wsdl:output message="tns:NewEventSoapOut"/>
```

```
            </wsdl:operation>
        </wsdl:portType>
        <wsp:Policy wsu:Id="UserNameTokenPasswordHashOverSSL">
            <wsp:ExactlyOne>
                <wsp:All>
                    <sp:TransportBinding>
                        <wsp:Policy>
                            <sp:TransportToken>
                                <wsp:Policy>
                                    <sp:HttpsToken>
                                        <wsp:Policy />
                                    </sp:HttpsToken>
                                </wsp:Policy>
                            </sp:TransportToken>
                            <sp:Layout>
                                <wsp:Policy>
                                    <sp:Lax />
                                </wsp:Policy>
                            </sp:Layout>
                            <sp:IncludeTimestamp />
                            <sp:AlgorithmSuite>
                                <wsp:Policy>
                                    <sp:Basic128 />
                                </wsp:Policy>
                            </sp:AlgorithmSuite>
                        </wsp:Policy>
                    </sp:TransportBinding>
                    <sp:SupportingTokens>
                        <wsp:Policy>
                            <wsp:SecurityTokenReference >
                                <wsp:Policy>
                                    <wsp:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" />
                                </wsp:Policy>
                            </wsp:SecurityTokenReference >
                        </wsp:Policy>
                    </sp:SupportingTokens>
                </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>
        <wsdl:binding name="EventCommunicationSoap12" type="tns:EventCommunicationPortType">
            <wsp:PolicyReference URI="#UserNameTokenPasswordHashOverSSL"/>
            <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
            <wsdl:operation name="NewEvent">
                <soap12:operation soapAction="http://www.water.domain/NewEvent" style="document"/>
                <wsdl:input>
                    <soap12:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap12:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="EventCommunication">
            <wsdl:port name="EventCommunicationSoap12" binding="tns:EventCommunicationSoap12">
                <soap12:address location="http://IP:80/EventCommunication"/>
```

```
            </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Annex D
(informative)

Interoperability test protocol

D.1 Overview

This annex defines and describes the tools and tests required to verify the adaptation of a remote monitoring and control system of this document.

This test protocol is intended for any of the components that are part of an interoperable architecture:

— MIS applications for management;

— Coordination brokers; and

— Subsystems.

The application or system to be tested should be submitted to the battery of tests assigned to it. For each test, the type of component to be tested should be identified.

Any compatible component with this document should conform to one or more of the following groups of verifications:

1) Interface verification tests. Messages (both requests and responses) should conform to this document. Applicable to MIS applications, coordination brokers and subsystems.

2) Verification tests of actions execution and interfaces. Verify the correct understanding of the semantics defined by this document. Applicable to MIS applications, coordination brokers and subsystems.

3) Verification functionality verification tests. Check the performing of the SUT functionalities execution. Applicable to subsystems.

Any product that successfully completes the applicable tests should be considered adapted to this document.

D.2 General description

Test facilities should have two parts:

1) IT infrastructure to check the interfaces. The test facilities should have available testers for all the interfaces mentioned in this document. These testers should use the communication protocols described in the implementation annexes for the interfaces (Annexes A or F for management interface, Annexes B or G for subsystems interface and Annexes C or H for events interface).

2) Real or simulated hydraulic and sensory infrastructure to check the System Under Test (SUT) functionalities. The test bed have one or more of the irrigation entities included. The group of devices that make up an irrigation entity of the standard's physical model are going to be referred to as a test module hereinafter. A test bed could have as many different test modules as entity types are represented in. A test module can be a hydraulic or an electronic test module. In the first case, the test bed should have all the hydraulic components defined for the entity type controlled or managed by the SUT. In the case of a hydraulic test bed, it is need a water supply. In the second, an electronic system can simulate the hydraulic components of the irrigation entity controlled or managed by the SUT. In this case there is no need for a water supply.

The tests to perform are be based on ISO 9646 series, using the "local test method" architecture strategy.

D.3 IT infraestructure

The required IT equipment allows the installation of the software components of the SUT and the testing tools. The SUT and the testers may be located in different operating systems, in different computers or in different virtual machines. In all these situations, all elements should be located at the same LAN. The IT equipment should:

— have a switch to connect all the devices;

— have all its machines, virtual or physical, connected to the local LAN using Ethernet connection;

— have internet connection; and

— allow the establishment of point to point connections with serial devices (RS232/RS485 interface) or others. This connection could be done using Ethernet converters from the communication interface required by each device.

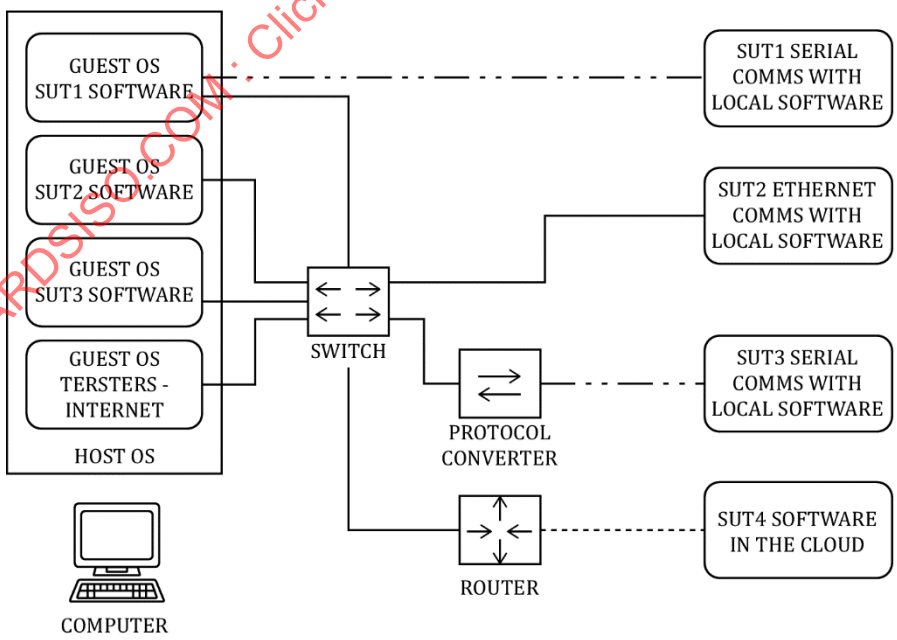


Figure D.1 — Testing architecture

"The local test method" test architecture states a Point of Control and Observation (PCO) between the SUT and the tester. The tester should be compatible with the communication protocol implemented by the SUT. If possible, tools developed by third parties can be used for testing the implementation of this document.

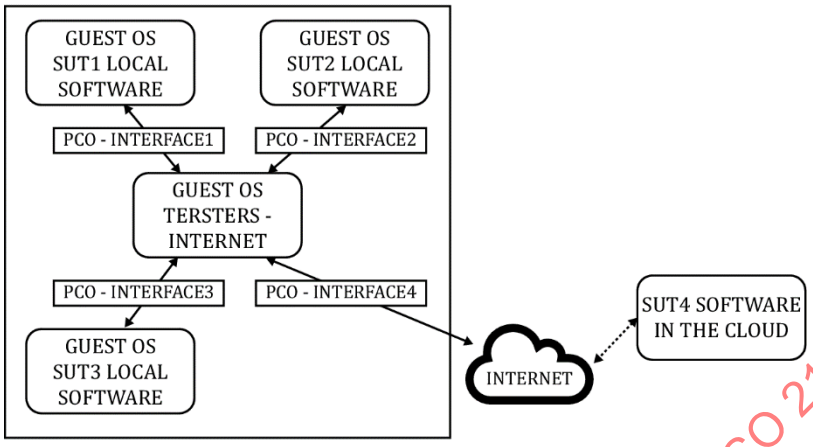


Figure D.2 — Interaction between SUTs and testers

D.4 Test bed description

D.4.1 General

The test bed can have modules to represent the different types of irrigation entities included in this document. Some modules can be used to test different types of these entities.

The test modules are intended to verify that the SUTs are able to control and execute the processes associated to an entity type. The modules should have all the components required to execute these processes.

To guarantee correct hydraulic conditions, the pressure supplied to the test module should be higher or equal than the pressure required by the cut-out element. Pilot regulators for flow rate and pressure should be used to ensure that the cut-out element and the volume totalizer operate within their hydraulic design parameters.

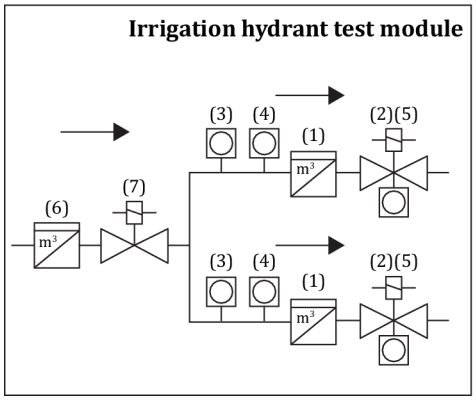
The test bed can be entirely simulated using electronic devices, designed to simulate the hydraulic behavior of the irrigation entity that represent.

This version of the test protocol only includes the definition of the irrigation hydrant test module. The test modules of the rest of irrigation entities included in the document can be defined in further versions.

D.4.2 Irrigation hydrant test module

This module can be used to test a subsystem that controls one or more of the irrigation hydrant types defined in this document. This test module can be used to test any type of hydrant, using a specific set of irrigation devices.

The irrigation hydrant test module consisnts of devices to check their basic functionalities, as shown in Figure D.3.



Key

1	volume totalizer	5	cut-out element position detector (optional)
2	cut-out element	6	master volume totalizer
3	pressure gauge (optional)	7	master cut-out
4	water flow detector (optional)		

Figure D.3 — Diagram of irrigation hydrant test module components

Repeated numbers indicate that the module could have more sets of these components.

These devices can be built in different physical elements (i.e. totalizer and valve) or they can be part of a combined device (hydrometer). A subsystem should operate these elements, therefore:

a) The volume totalizers have a reed relay type of pulse emitter connected to a digital input of the SUT control device.

b) The cut-out elements should be operated by a solenoid connected to a digital output of the control device, designed to operate this kind of actuators. For HYS, the typical solenoid is latch type.

The module can include additional devices to check other properties of the entity: pressure transducers and other sensors using analogue standard signal. The sensor signal should be compatible with the signal type supported by the control device.

All devices can be directly connected to the control device or, preferably, to a connection box with the next available signals to connect:

1) three (3) totalizer digital input;

2) three (3) digital output to a solenoid two or three wire;

3) four (4) general purpose digital inputs;

4) two (2) analogue input; and

5) two (2) or more connections available for other analogue inputs (environmental and soil moisture sensors).

The control device should be connected at least to the minimum devices composing the irrigation entity to be managed or controlled.

The module should also have connection blocks for the power supply, as well as an interface for a point to point (PTP) connection. This interface may be defined by the control device connection requirements.

According to the specifications of the different irrigation hydrant types, the devices required for the tests are those given in Table D.1.

Table D.1 — Devices required for irrigation hydrant test

	HYS		MIH		VIH		IHG	
Device	Required	Optional	Required	Optional	Required	Optional	Required	Optional
(1)	x		x			x		
(2)	x					x		
(3)		x		x		x		
(4)		x		x		x		
(5)		x				x		
(6)		x		x	x	x	x	x
(7)		x		x	x	x	x	x

By its definition, the VIH always acts as a part of an IHG. The double character of the devices in an IHG and VIH tests means that:

— if (6) is not in use because it is optional, (7) is required; and

— if (7) is not in use because it is optional, (6) is required.

At least (6) or (7) should be present in the IHG tests.

D.5 Test procedure

D.5.1 General

The following methodology is established to perform the tests, attending to the requirements established by this document to run and control each type of irrigation entity. The owner shall provide an Implementation conformance statement (ICS) identifying the SUT requirements and which entities can be controlled by it.

The set of tests to be passed by a SUT is defined according the product specs delivered by the manufacturer. The defined procedure performs, from the next verifications, the ones that apply to the SUT:

a) Interface verification testing. Messages (both requests and responses) comply with this document.

b) Interface and action execution verification testing. Verify the correct understanding of the semantic defined by this document.

c) SUT functionalities verification testing. Check the behavior of the SUT executing its functionalities over the test module.

D.5.2 Implementation conformance statement (ICS)

The implementation conformance statement (ICS) provided by the owner, defines the set of tests to perform on the SUT during the test procedure.

Table D.2 — ICS declaration

Parameter	Definition												
Brandname of the owner	Name of the SUT owner.												
Tradename of the SUT	Comercial name of the system under test.												
SUT type	Type of product to be tested: MIS application, coordination broker or subsystem.												
Installation requirements: hardware	Minimum requirements to run the software of the SUT (processor, memory, HD) and pyshical interface required to connect the IT tools of the test bench with the SUT hadware (serial port, ethernet port, others). Applicable to any kind of SUT.												
Installation requirements: software	Operating system, frameworks, database, others.												
Functionalities	Description of the system under test. In the case of a software application, the management functionalities should be defined (irrigation programming, billing, modelling of the hydraulic and/or energy function, or others). The methods of the standard implemented by the SUT to perform its funcionalities should be explicitly identified. In the case of a subsystem, the characterisation should be completed in the following sections of the implementation conformance statement.												
Installation requierements: electronic devices	Applicable to subsystems. With regard to the electronic components of the SUT, should be informed the architecture required to execute the tests and specifically the minimum requirements of all electronic devices; remote terminal units and, if existing, intermediate elements: — power supply requirements and alternatives (voltage, type of accumulator, type of generator); — available communication interfaces (serial port RS232, RS485, Ethernet, others); — communication media settings and requirements; and — specification of inputs and outputs (range and electrical signal for the analogue inputs and other data required for Read sensors and for act over actuators).												
Classification statement attending to SystemStatus definition	Applicable to subsystems. Classification code of the SUT, attending to the standard SystemStatus property definition.												
Characterization statement attending to SystemStatus definition	Applicable to subsystems. Characterization code of the SUT, attending to the standard SystemStatus property definition.												
Which time between communications is used by the SUT?	Applicable to subsystems. Time required for the complete updating of a SUT device data, expressed in s.												
Time required to determine a communication error	Applicable to subsystems. Time required to determine a communication error, expressed in s.												
Which interoperability protocol implements? (Yes/No) Applicable to eny kind of SUT.	SOAP 1.2							REST					
Which ISO 21622	HYS	IRA	VIH	IHG	PMS	INT	RSV	SSS	FTS	NBU	WFS	NCP	MIH

Parameter	Definition												
entity types can the SUT control/manage? (Yes/No) Applicable to eny kind of SUT.													
Which properties are supported by the SUT? (Yes/No) Note: Only identify extended properties. The mandatory ones are defined by the entity types controlled. Applicable to eny kind of SUT.	ActivityStatus		SystemStatus		Mode	
	CumulativeVolumeOut		InternalFlowOut		PressureOut	
	OpeningDegree		Temperature		RelativeHumidity	
	SolarRadiation		WindParameters		Rainfall	
	CumulativeVolumeIn		InternalFlowIn		PressureIn	
	InstantPower		EnergyConsumption		EntityPerformance	
	PowerOnTime		Capacity		SoilWaterContent	
	SoilWaterPotential		SoilTemperature		SoilConductivity	
	DifferentialPressure		WaterAcidity		WaterConductivity	
Which types of procedural elements are executable by the SUT? (Yes/No) See the list of acronyms in the 7.5 and 7.6 of the standard. Applicable to eny kind of SUT.	IOR1	IOR2	IOR3	POR1	POR2	POR3	POR4	POR5	SOR1	IGR1	NBR1	CPR1	IUP1
	SUP1	PUP1	RUP1	FOR1	FTR1	FTR2	FTR3	FTR4	FTR5	FUP1	GUP1		
SUT specifications (1) (Yes/No) Applicable to subsystems.	The SUT supports procedural element repetitions?		If the SUT supports repetitions, it has any restriccition associated?		The SUT supports procedural element calendars?		If the SUT supports calendars, it can attend to the FinalDate hour setup?	
SUT specifications (2) (Yes/No) Applicable to subsystems.	The SUT supports standard histories of 48 values?		The SUT supports standard histories of 96 values?		Has the SUT restrictions about the inmediacy in the start date of a procedural element?		The SUT supports multiple procedural element sending and has a queue management?	
SUT specifications (3) (Yes/No) Applicable to subsystems.	The SUT supports volume monitoring during procedural element execution?		The SUT supports pressure monitoring during procedural element execution?		The SUT supports flow monitoring during procedural element execution?			
Which are the restrictions associated to procedural element execution? (Description)	Applicable to subsystems. Informs about the restrictions to take into account when executing a procedural element (maximum admisible duration, capability to execute it when the execution starts in a day and ends in the next, others).							

Parameter	Definition
Has the SUT owner provide its hardware and software installation documents and user manuals? (Yes/No) (Identification of the provided documents)	Applicable to eny kind of SUT. Informs if the owner provides the documentation required to install and run the SUT, identifiying which are these documents and which is its utility.
The SUT hardware components has its wire connection schema? (Yes/No) (Identification of the provided documents)	Applicable to subsystems. Informs if the wire connection schema has been provided for each of its hardware components, identifying which are these documents and which is its utility.

D.5.3 Common criteria applied to the conformance tests

Syntactically speaking, all tests should use the methods and the parameters described. When parameter <<Value>> is specified, it should be understood as adjusted to the specification of this document. In case a <<Random value>> is specified in a test, it should be understood as an incompatible value with the definition established by this document. If there is a parameter specified as <<Irrelevant>>, this is interpreted as any valid value according to the standard for this parameter but not being part of the test verifications.

The names E00X and P000X are used for the EntityID and ProceduralID parameters as generic identifiers to adapt to the structure of identifiers defined, and to the reality of each test bed.

The timestamps expected in the responses are expressed as “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.

D.5.4 Security criteria

All the testers to be used in the test cases should be adapted to the security methods that the SUT implements in all its standard interfaces.

D.5.5 Results

The SUT should pass the conformance tests linked to the declaration of the owner in the ICS. The tests are passed when the responses are coincident with the expected response parameters, in accordance with the format established for the value of each property. In the negative tests, the response should not contain values other than ActionID and ResponseCode. The SUT does not pass the test if it does not comply with the aforementioned conditions.

In all test cases, all the requests and responses are validated as a previous step to evaluate their content, according to the definition of each interface.

D.6 Tests over subsystems

D.6.1 Test application guide

According to the ICS, the Table D.3 defines the applicable tests for a SUT when it is a subsystem.

Table D.3 — Test guide application over subsystems based on ICS

SUT specifications	Entity type	Test case
All SUTs	All entity types	TC1.1, TC1.3, TC1.5 to TC1.7, TC2.1 to TC2.4, TC3.2 to TC3.6, TC4.1 to TC4.4, TC5.1 to TC5.4, TC6.1 to TC6.4, TC7.1, TC7.5, TC7.7 to TC7.14, TC8.1 to TC8.2, TC9.1 to TC9.2, TC10.1 to TC10.3, TC11.1, TC12.1 to TC12.2, TC13.1 to TC13.4, TC14.1 to TC14.2, TC15.1, TC16.1
SUTs that supports extended properties	All entity types	TC1.2, TC1.4, TC7.4, TC7.6, TC18.1
All SUTs	All entity types	TC3.1, TC17.1, TC17.2
SUTs that support IrrigationRecipe2	HYS, VIH	TC3.10, TC11.2
SUTs that support procedural element s with calendar	HYS, VIH, PMS	TC3.1, TC3.15, TC3.16, TC6.5
SUTs that support procedural elements with calendar attending to FinalDate	HYS, VIH, PMS	TC3.11, TC3.12
SUTs that support procedural elementd with calendar not attending to FinalDate	HYS, VIH, PMS	TC3.13, TC3.14
SUTs that support multiple procedural element sending	All entity types	TC3.7 to TC3.9
SUTs that support repetitions on its procedural elements	All entity types	TC3.17 to TC3.19, TC6.6
SUTs that support histories with 48 values	All entity types	TC7.2
SUTs that support histories with 96 values	All entity types	TC7.3
SUTs that support monitorings during its procedural elements execution	All entity types	TC17.3

D.6.2 Interface and action execution verification tests

D.6.2.1 General

Tests should be performed in accordance with ISO 9646 whereby:

a) A group of conformance test defined as abstract, where the objective of each group is specified with independence on the technology to be applied.

b) Previously to the conformance tests, should be performed the validation of requests and responses structures. These should respect the implementation annex applicable to the SUT. The structure validation should be a general prerequirement for all conformance tests.

c) Using the abstract definition as the basis, a set of conformance tests has been defined. In order to verify the conformity of the SUT, it should pass all applicable test cases (positive and negative). The test bench should have the testing tools required for the SUT.

D.6.2.2 Test "Reading a property of an entity (Read method)"

D.6.2.2.1 Abstract conformance test

Table D.4 — Test description Read method

Test purpose	To verify that the implementation of the Read method at the SUT is compatible with the standard definition.
System under test	Subsystem or coordination broker acting as subsystem.
Test sequence	A tester acting as coordination broker sends a Read request to the SUT. The SUT generates a response according to the execution test case. The value included in the response is confirmed using the SUT user interface.
Request parameters	ActionID, EntityID, PropertyName.
Response parameters	ActionID, ResponseCode, Value, TimeStamp.

D.6.2.2.2 Conformance tests

A set of tests made up of the following are performed:

1) Four (4) positive tests; and

2) Two (2) negative tests.

Table D.5 — Test case 1.1

Identifier	TC1.1	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify the ability to Read all mandatory property that exists in the entity type controlled by the SUT.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed" and the value of each mandatory property. 3. The value obtained by the action shall match (applying unit conversion if required) the one available in the SUT user interface. In case a property is not available in the user interface, this step is not be taked into account. This sequence shall be repeated for all mandatory properties of the irrigation entity. For the properties of string type value, the test shall be repeated to obtain all its possible values.	
Request 1 parameter 1	ActionID (string)	"Test_1.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those mandatory properties belonging to the entity type>>
Response 1 parameter 1	ActionID (string)	"Test_1.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value	<<Value>>

Table D.6 — Test case 1.2

Identifier	TC1.2	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The SUT shall support the extended property under test.	
Test purpose	To verify the ability to Read all extended properties of the irrigation entity supported by the SUT.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed" and the value of each supported extended property. 3. The value obtained by the action shall match (applying unit conversion if required) the one available in the SUT user interface. In case where a property is not Readable from the user interface, this step is not be taked into account. This sequence shall be repeated for all extended properties of the irrigation entity supported by the SUT. For the properties of string type value, the test shall be repeated to obtain all its possible values.	
Request 1 parameter 1	ActionID (string)	"Test_1.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All extended properties of the entity type supported by the SUT>>
Response 1 parameter 1	ActionID (string)	"Test_1.2"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value	<<Value>>

Table D.7 — Test case 1.3

Identifier	TC1.3	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The SUT shall not support the extended property under test.	
Test purpose	To verify the ability to Read an extended property not supported by the SUT.	
Test sequence	1. The tester sends a Read request to the SUT. 2. The SUT returns a "3 – Not supported" for each non supported extended property. This sequence shall be repeated for all extended properties not supported by the SUT.	
Request 1 parameter 1	ActionID (string)	"Test_1.3"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All extended properties of the entity type not supported by the SUT>>
Response 1 parameter 1	ActionID (string)	"Test_1.3"
Response 1 parameter 2	ResponseCode (string)	"3 – Not supported"

Table D.8 — Test case 1.4

Identifier	TC1.4	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The SUT shall support the extended property under test.	
Test purpose	To verify the ability to Read an extended property supported by the SUT but not available in the irrigation entity "E00X".	
Test sequence	1. The SUT shall be configured to lack the sensor required to obtain the property value. 2. The tester sends a Read request (1) to the SUT. 3. The SUT returns a "6 – Not avaliable" for each supported but not present extended property. This sequence shall be repeated for all extended properties supported by the SUT.	
Request 1 parameter 1	ActionID (string)	"Test_1.4"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All extended properties belonging to the entity and supported by the SUT but that do not exist in E00X>>
Response 1 parameter 1	ActionID (string)	"Test_1.4"
Response 1 parameter 2	ResponseCode (string)	"6 – Not available"

Table D.9 — Test case 1.5

Identifier	TC1.5	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker.	
Prerequirement 1	The entity "E00X" shall not exist in the SUT.	
Test purpose	To verify the ability of the SUT to filter by identifiers, returning an error message when the identifier does not exist.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a "2 - Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_1.5"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	"Test_1.5"
Response 1 parameter 2	ResponseCode (string)	"2 - Lexical error"

Table D.10 — Test case 1.6

Identifier	TC1.6	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Test purpose	To verify the ability of the SUT to filter by PropertyName, returning an error message when the PropertyName introduced is not included in this document.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a “2 - Lexical error”.	
Request 1 parameter 1	ActionID (string)	“Test_1.6”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	<<Random value>>
Response 1 parameter 1	ActionID (string)	“Test_1.6”
Response 1 parameter 2	ResponseCode (string)	“2 - Lexical error”

D.6.2.3 Test “Write a property of an entity (Write method)”

D.6.2.3.1 Abstract conformance test

Table D.11 — Test description for Write method

Test purpose	To verify that the implementation of the Write method at the SUT is compatible with the standard definition.
System under test	Subsystem or coordination broker acting as subsystem.
Test sequence	A tester acting as coordination broker sends a Write request to the SUT. The SUT generates a response according to the execution test case. In some cases, the SUT should update the property value with the one introduced by the tester. The new value is verified using the Read action.
Request parameters	ActionID, EntityID, PropertyName, Value.
Response parameters	ActionID, ResponseCode.

D.6.2.3.2 Conformance tests

A set of tests made up of the following are performed:

1) Two (2) positive test; and

2) Two (2) negative tests.

Table D.12 — Test case 2.1

Identifier	TC2.1
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem or coordination broker acting as subsystem.
Prerequirement 1	The entity “E00X” shall exist in the SUT.

Prerequirement 2	The Read method has been successfully tested previously.	
Test purpose	To verify the ability to Write a writable property in the entity controlled by the SUT.	
Test sequence	1. The tester sends a Read request (1) to the SUT to determine the current value of a writable property. 2. The tester sends a Write request (2) to the SUT, introducing a new value for the writable property. The new value shall respect the standard definition. 3. The SUT returns a "0 – Action successfully executed". 4. The tester sends a Read request (3) to the SUT to determine the value of the writable property. 5. The SUT returns a "0 – Action successfully executed" and the value introduced in the step 2. This sequence shall be repeated for all writable properties (mandatory and extended) supported by the SUT. For the properties with a list of possible values, the sequence shall be repeated for all its possible values.	
Request 1 parameter 1	ActionID (string)	"Test_2.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Writable property>>
Response 1 parameter 1	ActionID (string)	"Test_2.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value	<<Value>>
Request 2 parameter 1	ActionID (string)	"Test_2.1"
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	PropertyName (string)	<<Writable property>>
Request 2 parameter 4	Value	<<Valid value>>
Response 2 parameter 1	ActionID (string)	"Test_2.1"
Response 2 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Request 3 parameter 1	ActionID (string)	"Test_2.1"
Request 3 parameter 2	EntityID (string)	"E00X"
Request 3 parameter 3	PropertyName (string)	<<Writable property>>
Response 3 parameter 1	ActionID (string)	"Test_2.1"
Response 3 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 3 parameter 3	Value	<<Value of request 2>>

Table D.13 — Test case 2.2

Identifier	TC2.2
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Positive
System under test	Subsystem or coordination broker acting as subsystem.
Prerequirement 1	The entity "E00X" shall exist in the SUT.
Prerequirement 2	The Read method has been tested previously successfully.
Test purpose	To verify that the SUT do not allow the writing of any property defined as non-writable.

Test sequence	1. The tester sends a Read request (1) to the SUT to determine the current value. 2. The tester sends a Write request (2) to the SUT, introducing a new value for a non-writable property. 3. The SUT returns a "3 – Not supported". 4. The tester sends a Read request (3) to the SUT to verifiy that the property value has not been changed. This sequence shall be repeated for all non-writable properties (mandatory and extended) supported by the SUT. The same response shall be obtained for any non-supported extended property.	
Request 1 parameter 1	ActionID (string)	"Test_2.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Non-writable property>>
Response 1 parameter 1	ActionID (string)	"Test_2.2"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value	<<Value>>
Request 2 parameter 1	ActionID (string)	"Test_2.2"
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	PropertyName (string)	<<Non-writable property>>
Request 2 parameter 4	Value	<<Irrelevant>>
Response 2 parameter 1	ActionID (string)	"Test_2.2"
Response 2 parameter 2	ResponseCode (string)	"3 – Not supported"
Request 3 parameter 1	ActionID (string)	"Test_2.2"
Request 3 parameter 2	EntityID (string)	"E00X"
Request 3 parameter 3	PropertyName (string)	<<Non-writable property>>
Response 3 parameter 1	ActionID (string)	"Test_2.2"
Response 3 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 3 parameter 3	Value	<<Value>>

Table D.14 — Test case 2.3

Identifier	TC2.3	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify that the SUT filters by the parameter PropertyName.	
Test sequence	1. The tester sends a Write request (1) to the SUT, introducing a new value for a random PropertyName. The PropertyName shall not be compatible with the standard definition. 2. The SUT returns a "1 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_2.3"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Random value>>
Request 1 parameter 4	Value	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	"Test_2.3"

Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

Table D.15 — Test case 2.4

Identifier	TC2.4	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify that the SUT validates the format introduced for the writable properties.	
Test sequence	1. The tester sends a Write request (1) to the SUT, introducing a random value for a writable property. The value shall not be compatible with the standard definition. 2. The SUT returns a "1 – Lexical error". This sequence shall be repeated for all writable properties (mandatory and extended) supported by the SUT.	
Request 1 parameter 1	ActionID (string)	"Test_2.4"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<PropertyName>>
Request 1 parameter 4	Value	<<Random value>>
Response 1 parameter 1	ActionID (string)	"Test_2.4"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

D.6.2.4 Test "Create a procedural element (CreateRecipe method)"

D.6.2.4.1 Abstract conformance test

Table D.16 — Test description CreateRecipe method

Test purpose	To verify that the implementation of the CreateRecipe method at the SUT is compatible with the standard definition.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker sends a CreateRecipe request to the SUT. The SUT generates a response according to the execution test case. In the positive cases, the SUT should to generate and execute the procedural element attending to its parameterezation.
Request parameters	ActionID, ProceduralID, EntityID, EntityType, RecipeType and OperationParameters (StartDate, MaxDuration, WeekCalendar and Final Date, Repetition, RepetitionNumber and RepetitionInterval).
Response parameters	ActionID, ResponseCode.

D.6.2.4.2 Specific conformance tests for procedural elements with and without calendar

This first test is required to run any other for this method. A set of tests made up of the following are performed.

1) One (1) positive test.

Table D.17 — Test case 3.1

Identifier	TC3.1	
Applicable to	HYS, VIH	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural elements with time end condition or calendar.	
Test purpose	To verify the ability to create an procedural element to execute in the entity controlled by the SUT. The start condition and end conditions are time type conditions.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make a procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “0 – Action successfully executed”. 3. If possible, to confirm at the SUT user interface the presence of the procedural element. 4. If not possible, directly verify its execution, monitoring at the SUT user interface possible changes between the start and the end conditions.	
Request 1 parameter 1	ActionID (string)	“Test_3.1”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_3.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

D.6.2.4.3 Common conformance tests

A set of tests made up of the following are performed:

1) Eight (8) negative tests.

2) One (1) positive test.

Table D.18 — Test case 3.2

Identifier	TC3.2
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Negative
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The procedural element “P00X” shall previously exist in the SUT.
Test purpose	To verify the ability to filter requests with bad parameters: duplicated ProceduralID.

Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a "2 – Lexical error". 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	"Test_3.2"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Response 1 parameter 1	ActionID (string)	"Test_3.2"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

Table D.19 — Test case 3.3

Identifier	TC3.3	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall not exist in the SUT.	
Test purpose	To verify the ability to filter requests with bad parameters: unknown EntityID.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a "2 – Lexical error". 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	"Test_3.3"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	<<Random value>>
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Response 1 parameter 1	ActionID (string)	"Test_3.3"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

Table D.20 — Test case 3.4

Identifier	TC3.4
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Negative

System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Test purpose	To verify the ability to filter requests with bad parameters: StartDate with random value.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “2 – Lexical error”. 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	“Test_3.4”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	<<Random value>>
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_3.4”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.21 — Test case 3.5

Identifier	TC3.5	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Test purpose	To verify the ability to filter requests with bad parameters: StartDate previous to the current time.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. The StartDate is referred to a previous moment to the current time. 2. The SUT returns a “2 – Lexical error”. 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	“Test_3.5”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_3.5”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.22 — Test case 3.6

Identifier	TC3.6	
Applicable to	HYS, VIH	
Verification of	Subsystem interface and action execution	
Test type	negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural elements with time end condition or calendar.	
Test purpose	To verify the ability to create an procedural element to execute in the entity controlled by the SUT. The duration of the operation is 0.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “2 – Lexical error”. 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	“Test_3.6”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“0”
Response 1 parameter 1	ActionID (string)	“Test_3.6”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.23 — Test case 3.7

Identifier	TC3.7	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall be working with its design communication interval.	
Test purpose	To verify the ability to filter requests with bad parameters: StartDate is setup to be before the SUT communication.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate is after the last communication and before the next scheduled. 2. The SUT returns a “1 – Execution error”. 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	“Test_3.7”

Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_3.7”
Response 1 parameter 2	ResponseCode (string)	“1 – Execution error”

Table D.24 — Test case 3.8

Identifier	TC3.8	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural elements “P00X” and “P00Y” shall not exist in the SUT.	
Prerequirement 3	The SUT shall be working with its design communication interval.	
Prerequirement 4	The SUT shall support multiple procedural element sending.	
Test purpose	To verify the ability to filter múltiple procedural elements with the same start condition.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “0 – Action successfully executed”. 3. If possible, to confirm at the SUT user interface the presence of the procedural element. 4. The tester sends a CreateRecipe request (2) to the SUT. The StartDate condition of the request (2) shall be the same than the preceding request. 5. The SUT returns a “1 – Execution error” or “3 – Not supported”	
Request 1 parameter 1	ActionID (string)	“Test_3.87”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“8600”
Response 1 parameter 1	ActionID (string)	“Test_3.8”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_3.8”
Request 2 parameter 2	ProceduralID (string)	“P00Y”
Request 2 parameter 3	EntityID (string)	“E00X
Request 2 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 2 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 2 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.

Request 2 parameter 7	MaxDuration (integer)	“600”
Response 2 parameter 1	ActionID (string)	“Test_3.8”
Response 2 parameter 2	ResponseCode (string)	“1 – Execution error” or “3 – Not supported”

Table D.25 — Test case 3.9

Identifier	TC3.9	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” and “P00Y” shall not exist in the SUT.	
Prerequirement 3	The SUT shall be working with its design communication interval.	
Prerequirement 4	The SUT shall not support multiple procedural element sending.	
Test purpose	To verify the ability to filter requests of procedural elements overlapping in time.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make a procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “0 – Action successfully executed”. 3. If possible, to confirm at the SUT user interface the presence of the procedural element. 4. The tester sends a CreateRecipe request (2) to the SUT. The StartDate condition of the request (2) shall be overlapping in time but not equal to the preceding request. 5. The SUT returns a “1 – Execution error” or “3 – Not supported”.	
Request 1 parameter 1	ActionID (string)	“Test_3.9”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_3.9”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_3.9”
Request 2 parameter 2	ProceduralID (string)	“P00Y”
Request 2 parameter 3	EntityID (string)	“E00X
Request 2 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 2 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 2 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 7	MaxDuration (integer)	“600”
Response 2 parameter 1	ActionID (string)	“Test_3.9”
Response 2 parameter 2	ResponseCode (string)	“1 – Execution error” or “3 – Not supported”

Table D.26 — Test case 3.10

Identifier	TC3.10	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” and “P00Y” shall not exist in the SUT.	
Prerequirement 3	The SUT shall be working with its design communication interval.	
Prerequirement 4	The SUT shall support multiple procedural elements sending.	
Prerequirement 5	The StartDate of “P00X” and P00Y” are not the same.	
Prerequirement 6	The ReadProceduralIDs method shall be tested previously.	
Test purpose	To verify the ability to manage requests of procedural elements overlapping in time.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “0 – Action successfully executed”. 3. If possible, to confirm at the SUT user interface the presence of the procedural element. 4. The tester sends a CreateRecipe request (2) to the SUT. The StartDate condition of the request (2) shall be overlapping the request (1) StartDate. 5. The SUT returns a “0 – Action successfully executed”. 6. When the StartDate of “P00X,0” has been reached, the tester sends a ReadProceduralIDs request (3) for Running status. 7. The SUT shall return a “0 – Action successfully executed” and “P00X,0”. 8. After the StartDate of “P00Y” has been reached, the tester sends a ReadProceduralIDs request (4) for Running status. 9. The SUT shall return a “0 – Action successfully executed” and “P00Y,0”. 10. The tester sends ReadProceduralIDs request (5) for Completed status, before the theoretical End condition of “P00X” has been reached. 11. The SUT shall return a “0 – Action successfully executed” and “P00X,0”.	
Request 1 parameter 1	ActionID (string)	“Test_3.10”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_3.10”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_3.10”
Request 2 parameter 2	ProceduralID (string)	“P00Y”
Request 2 parameter 3	EntityID (string)	“E00X
Request 2 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 2 parameter 5	RecipeType (string)	<<Any type of procedural element>>
Request 2 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.

Request 2 parameter 7	MaxDuration (integer)	“600”
Response 2 parameter 1	ActionID (string)	“Test_3.10”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 3 parameter 1	ActionID (string)	“Test_3.10”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 3 parameter 4	Status (Status)	“Running”
Response 3 parameter 1	ActionID (string)	“Test_3.10”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 4 parameter 1	ActionID (string)	“Test_3.10”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	“Running”
Response 4 parameter 1	ActionID (string)	“Test_3.10”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	ProceduralIDs (string)	“P00Y,0”
Request 5 parameter 1	ActionID (string)	“Test_3.10”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 5 parameter 4	Status (Status)	“Completed”
Response 5 parameter 1	ActionID (string)	“Test_3.10”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	ProceduralIDs (string)	“P00X,0”

D.6.2.4.4 Specific conformance tests for IrrigationRecipe2

A set of tests made up of the following are performed:

1) One (1) positive test.

Table D.27 — Test case 3.11

Identifier	TC3.11
Applicable to	HYS, VIH
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The operation “P00X” shall not exist in the SUT.
Prerequirement 3	The SUT shall support IrrigationRecipe2 procedural elements.
Test purpose	To verify the ability to create an operation to execute in the entity controlled by the SUT and without an end condition (IrrigationRecipe2).

Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an operation executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a "0 – Action successfully executed". 3. If possible, to confirm at the SUT user interface the presence of the operation. 4. If not possible, directly verify its execution, monitoring at the SUT user interface possible changes after the start condition.	
Request 1 parameter 1	ActionID (string)	"Test_3.11"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	"HYS" or "VIH"
Request 1 parameter 5	RecipeType (string)	"IrrigationRecipe2"
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Response 1 parameter 1	ActionID (string)	"Test_3.11"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"

D.6.2.4.5 Specific conformance tests procedural elements with calendar

A set of tests made up of the following are performed:

1) Three (3) positive test; and

2) Three (3) negative tests.

Table D.28 — Test case 3.12

Identifier	TC3.12	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural elements with calendar.	
Prerequirement 4	The SUT shall be able to attend to FinalDate to end the procedural element.	
Test purpose	To verify the ability to execute a procedural element with calendar in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a "0 – Action successfully executed". 3. If possible, to confirm at the SUT user interface the presence of the procedural element. 4. If not possible, directly verify its execution, monitoring at the SUT user interface possible changes between the start and the end conditions.	
Request 1 parameter 1	ActionID (string)	"Test_3.12"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	"HYS" or "VIH" or "PMS"

Request 1 parameter 5	RecipeType (string)	<<Procedural element type with calendar>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Request 1 parameter 8	WeekCalendar (string)	<<A possible combination of values>>
Request 1 parameter 9	FinalDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Response 1 parameter 1	ActionID (string)	"Test_3.12"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"

Table D.29 — Test case 3.13

Identifier	TC3.13	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem able to attend to FinalDate to end the procedural element.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural elements with calendar. The SUT has not restrictions on FinalDate parameterization.	
Prerequirement 4	The SUT shall be able to attend to FinalDate to end the procedural element.	
Prerequirement 5	The SUT has been passed the TC3.8 test.	
Test purpose	To verify the ability to manage the FinalDate parameter of an procedural element with calendar to execute in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The FinalDate parameter is set up to reach before a repetition of the calendar its completed. 3. The SUT returns a "0 – Action successfully executed" and execute the procedural element attending to the defined FinalDate. 4. Verify its execution.	
Request 1 parameter 1	ActionID (string)	"Test_3.13"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	"HYS" or "VIH" or "PMS"
Request 1 parameter 5	RecipeType (string)	<<Procedural element with calendar>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Request 1 parameter 8	WeekCalendar (string)	<<A possible combination of values>>
Request 1 parameter 9	FinalDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Response 1 parameter 1	ActionID (string)	"Test_3.13"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"

Table D.30 — Test case 3.14

Identifier	TC3.14	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem not able to attend FinalDate hour to end the procedural element.	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural element with calendar. The SUT has restrictions in FinalDate parameterization.	
Prerequirement 4	The SUT shall be able to attend to FinalDate to end the procedural element.	
Test purpose	To verify the ability to manage the FinalDate parameter of an procedural element with calendar to execute in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The FinalDate parameter is set up to reach after a repetition of the calendar its completed. 3. The SUT returns a “0 – Action successfully executed” and execute the procedural element attending to the end condition of the last repetition included at the FinalDate parameter (the SUT should attend to the date, but not the hour of FinalDate). 4. Verify its execution.	
Request 1 parameter 1	ActionID (string)	“Test_3.14”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	“HYS” or “VIH” or “PMS”
Request 1 parameter 5	RecipeType (string)	<<Procedural element with calendar>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Request 1 parameter 8	WeekCalendar (string)	<<A possible combination of values>>
Request 1 parameter 9	FinalDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Response 1 parameter 1	ActionID (string)	“Test_3.14”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Table D.31 — Test case 3.5

Identifier	TC3.15
Applicable to	HYS, VIH, PMS
Verification of	Subsystem interface
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.
Prerequirement 3	The SUT shall support procedural elements with calendar. The SUT has restrictions in FinalDate parameterization.

Prerequirement 4	The SUT is not able to attend FinalDate hour to end the procedural element.	
Test purpose	To verify the ability to manage the FinalDate parameter of a procedural element to execute in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The FinalDate parameter is set up to reach before a repetition of the calendar its completed. 3. The SUT returns a "1 – Execution error".	
Request 1 parameter 1	ActionID (string)	"Test_3.15"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	"HYS" or "VIH" or "PMS"
Request 1 parameter 5	RecipeType (string)	<<Procedural element with calendar>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Request 1 parameter 8	WeekCalendar (string)	<<A possible combination of values>>
Request 1 parameter 9	FinalDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Response 1 parameter 1	ActionID (string)	"Test_3.15"
Response 1 parameter 2	ResponseCode (string)	"1 – Execution error"

Table D.32 — Test case 3.16

Identifier	TC3.16	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural elements with calendar.	
Test purpose	To verify the ability to filter requests with bad parameters: WeekCalendar parameter has a non-valid value.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The FinalDate parameter is a random value. 2. The SUT returns a "2 – Lexical error". 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	"Test_3.16"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X
Request 1 parameter 4	EntityType (string)	"HYS" or "VIH" or "PMS"
Request 1 parameter 5	RecipeType (string)	<<Procedural element with calendar>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Request 1 parameter 8	WeekCalendar (string)	<<Random value>>
Request 1 parameter 9	FinalDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.

Response 1 parameter 1	ActionID (string)	“Test_3.16”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.33 — Test case 3.17

Identifier	TC3.17	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural elements with calendar.	
Test purpose	To verify the ability to filter requests with bad parameters: FinalDate parameter has a non-valid value.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate does not respect the time between communications. 2. The SUT returns a “2 – Lexical error”. 3. If possible, to confirm at the SUT user interface the absence of the procedural element.	
Request 1 parameter 1	ActionID (string)	“Test_3.17”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	“HYS” or “VIH” or “PMS”
Request 1 parameter 5	RecipeType (string)	<<Procedural element with calendar>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Request 1 parameter 8	WeekCalendar (string)	<<Three possible combination of values>>
Request 1 parameter 9	FinalDate (DateTime)	<<Random value>>
Response 1 parameter 1	ActionID (string)	“Test_3.17”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

D.6.2.4.6 Specific conformance tests for procedural elements with repetitions

A set of tests made up of the following are performed:

1) One (1) positive test.

2) Two (2) negative tests.

Table D.34 — Test case 3.18

Identifier	TC3.18
Applicable to	HYS, VIH, PMS
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.

Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural element than allows repetitions.	
Test purpose	To verify the ability to create a procedural element with repetitions to execute in the entity controlled by the SUT (IrrigationRecipe1).	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “0 – Action successfully executed”. 3. If possible, to confirm at the SUT user interface the existence of the repetitions included in the procedural element. 4. If not possible, directly verify its execution, monitoring at the SUT user interface possible changes between the start and the end conditions.	
Request 1 parameter 1	ActionID (string)	“Test_3.18”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	“HYS” or “VIH” or “PMS”
Request 1 parameter 5	RecipeType (string)	<<Procedural element with repetitions enabled>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Request 1 parameter 8	Repetition (boolean)	“1”
Request 1 parameter 9	RepetitionInterval (integer)	“600”
Request 1 parameter 10	RepetitionNumber (integer)	“5”
Response 1 parameter 1	ActionID (string)	“Test_3.18”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Table D.35 — Test case 3.19

Identifier	TC3.19	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Prerequirement 3	The SUT shall support procedural element than allows repetitions.	
Test purpose	To verify the ability to filter requests with bad parameters: RepetitionInterval parameter has a non-valid value.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make a procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a “2 – Lexical error”.	
Request 1 parameter 1	ActionID (string)	“Test_3.19”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	“HYS” or “VIH” or “PMS”
Request 1 parameter 5	RecipeType (string)	<<Procedural element with repetitions enabled>>

Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Request 1 parameter 8	Repetition (boolean)	"1"
Request 1 parameter 9	RepetitionInterval (integer)	"RandomValue"
Request 1 parameter 10	RepetitionNumber (integer)	"5"
Response 1 parameter 1	ActionID (string)	"Test_3.19"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

Table D.36 — Test case 3.20

Identifier	TC3.20	
Applicable to	HYS, VIH, PMS	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall not exist in the SUT.	
Prerequirement 3	The SUT shall execute procedural elements with repetitions.	
Test purpose	To verify the ability to filter requests with bad parameters: RepetitionNumber parameter has a non-valid value.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make a procedural element executable, attending to the restrictions established by its design time between communications. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_3.20"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	"HYS" or "VIH" or "PMS"
Request 1 parameter 5	RecipeType (string)	<<Procedural element with repetitions enabled>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Request 1 parameter 8	Repetition (boolean)	"1"
Request 1 parameter 9	RepetitionInterval (integer)	"600"
Request 1 parameter 10	RepetitionNumber (integer)	"RandomValue"
Response 1 parameter 1	ActionID (string)	"Test_3.20"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

D.6.2.5 Test "Read the procedural elements loaded in the SUT (ReadProceduralIDs method)"

D.6.2.5.1 Abstract conformance test

Table D.37 — Test description for ReadProceduralIDs method

Test purpose	To verify that the implementation of the ReadProceduralIDs method in the SUT is compatible with the standard definition.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker sends a ReadProceduralIDs request to the SUT. The SUT generates a response according to the execution test case.
Request parameters	ActionID, ProceduralID, Date, Status.

Response parameters	ActionID, ResponseCode, ProceduralIDs.

D.6.2.5.2 Conformance tests

A set of tests made up of the following are performed:

1. Three (3) positive test; and
2. One (1) negative test.

Table D.38 — Test case 4.1

Identifier	TC4.1	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The CreateRecipe method shall be tested previously.	
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT.	
Test purpose	To verify the ability to Read the IDs of procedural elements in Idle/Running/Completed statuses at the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that are being created shall start at least five minutes after the request. The StartDate of the procedural element shall be accord to the restrictions established between communications in the SUT. 2. The SUT returns a “0 – Action successfully executed”. 3. The tester sends a ReadProceduralIDs request (2) for Idle procedural elements, before the start condition is reached. 4. The SUT returns a “0 – Action successfully executed” and “P00X,0” as a value. 5. When the procedural element start condition is reached, the tester sends a ReadProceduralIDs request (3) for Running Status procedural elements . 6. The SUT returns a “0 – Action successfully executed” and “P00X,0” as a value. 7. When the procedural element end condition is reached, the tester sends a ReadProceduralIDs request (4) for Completed Status procedural elements. 8. The SUT returns a “0 – Action successfully executed” and “P00X,0” as a value. 9. After sept 7, the tester sends a ReadProceduralIDs request (5) for Idle Status procedural elements. 10. The SUT returns a “0 – Action successfully executed” and “P00X,0” does not appear. 11. After sept 9, the tester sends a ReadProceduralIDs request (6) for Running Status procedural elements. 12. The SUT returns a “0 – Action successfully executed” and “P00X,0” does not appear.	
Request 1 parameter 1	ActionID (string)	“Test_4.1”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_4.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_4.1”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Idle”
Response 2 parameter 1	ActionID (string)	“Test_4.1”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Response 2 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 3 parameter 1	ActionID (string)	“Test_4.1”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 3 parameter 4	Status (Status)	“Running”
Response 3 parameter 1	ActionID (string)	“Test_4.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 4 parameter 1	ActionID (string)	“Test_4.1”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	“Completed”
Response 4 parameter 1	ActionID (string)	“Test_4.1”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 5 parameter 1	ActionID (string)	“Test_4.1”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 5 parameter 4	Status (Status)	“Idle”
Response 5 parameter 1	ActionID (string)	“Test_4.1”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	ProceduralIDs (string)	“P00X,0” does not appear.
Request 6 parameter 1	ActionID (string)	“Test_4.1”
Request 6 parameter 2	EntityID (string)	“E00X”
Request 6 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 6 parameter 4	Status (Status)	“Running”
Response 6 parameter 1	ActionID (string)	“Test_4.1”
Response 6 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 6 parameter 3	ProceduralIDs (string)	“P00X,0” does not appear.

Table D.39 — Test case 4.2

Identifier	TC4.2
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The CreateRecipe and StopRecipe methods shall be tested previously.
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT.
Prerequirement 4	The StopRecipe method shall have been executed over “P00X” successfully.
Test purpose	To verify the ability to Read the IDs of procedural elements in status Stopped at the entity controlled by the SUT.

Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element "P00X" that is being created shall start at least five minutes after the request. The StartDate of the procedural element shall be in agreement with the restrictions established between communications at the SUT. 2. When the start condition is reached, the tester sends a ReadProceduralIDs request (2) for Running procedural elements. 3. The SUT returns a "0 – Action successfully executed" and "P00X" as a value. 4. The tester sends a StopRecipe request (3) for "P00X". 5. The SUT returns a "0 – Action successfully executed". 6. The tester sends a ReadProceduralIDs request (4) for Stopped procedural elements. The SUT returns a "0 – Action successfully executed" and "P00X" as a value. 7. The tester sends a ReadProceduralIDs request (5) for Running procedural elements. The SUT returns a "0 – Action successfully executed" and "P00X" does not appear.	
Request 1 parameter 1	ActionID (string)	"Test_4.2"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Response 1 parameter 1	ActionID (string)	"Test_4.2"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Request 2 parameter 1	ActionID (string)	"Test_4.2"
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	Date (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	"Running"
Response 2 parameter 1	ActionID (string)	"Test_4.2"
Response 2 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 2 parameter 3	ProceduralIDs (string)	"P00X,0"
Request 3 parameter 1	ActionID (string)	"Test_5.1"
Request 3 parameter 2	EntityID (string)	"E00X"
Request 3 parameter 3	ProceduralID (string)	"P00X,0"
Response 3 parameter 1	ActionID (string)	"Test_5.1"
Response 3 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Request 4 parameter 1	ActionID (string)	"Test_4.2"
Request 4 parameter 2	EntityID (string)	"E00X"
Request 4 parameter 3	Date (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	"Stopped"
Response 4 parameter 1	ActionID (string)	"Test_4.2"
Response 4 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 4 parameter 3	ProceduralIDs (string)	"P00X,0"
Request 5 parameter 1	ActionID (string)	"Test_4.2"
Request 5 parameter 2	EntityID (string)	"E00X"
Request 5 parameter 3	Date (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 5 parameter 4	Status (Status)	"Running"
Response 5 parameter 1	ActionID (string)	"Test_4.2"
Response 5 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 5 parameter 3	ProceduralIDs (string)	"P00X,0" does not appears.

Table D.40 — Test case 4.3

Identifier	TC4.3	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The CreateRecipe method shall be tested previously.	
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT.	
Prerequirement 4	The CreateRecipe and StopRecipe methods shall be tested previously.	
Prerequirement 5	The SUT shall support multiple not simultaneous procedural elements.	
Test purpose	To verify the ability to Read multiple IDs of procedural elements in all possible status at the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that are being created shall start at least five minutes after the request. 2. The SUT returns a “0 – Action successfully executed”. 3. If the SUT supports multiple procedural elements sending, the tester sends another CreateRecipe request (2). The procedural element “P00Y” that is being created shall start after the end condition of step 1 procedural element. 4. The SUT returns a “0 – Action successfully executed”. 5. The tester sends a ReadProceduralIDs request (3) to the SUT, before the procedural elements starting, with status Idle. 6. The SUT returns a “0 – Action successfully executed” in addition to “P00X” and “P00Y” as values. 7. After the first procedural element reaches its end condition, the tester sends a ReadProceduralIDs request (4) to the SUT, with the status Idle. 8. The SUT returns a “0 – Action successfully executed” and “P00Y” as value. 9. When the two procedural elements have been reached their end condition, the tester sends a ReadProceduralIDs request (5) to the SUT, with status Completed. 10. The SUT returns a “0 – Action successfully executed” and “P00X” and “P00Y” as values.	
Request 1 parameter 1	ActionID (string)	“Test_4.3”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_4.3”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_4.3”
Request 2 parameter 2	ProceduralID (string)	“P00Y”
Request 2 parameter 3	EntityID (string)	“E00X”
Request 2 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 2 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 2 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.

Request 2 parameter 7	MaxDuration (integer)	“600”
Response 2 parameter 1	ActionID (string)	“Test_4.3”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 3 parameter 1	ActionID (string)	“Test_4.3”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 3 parameter 4	Status (Status)	“Idle”
Response 3 parameter 1	ActionID (string)	“Test_4.3”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	ProceduralIDs (string)	“P00X, 0”, “P00Y, 0”
Request 4 parameter 1	ActionID (string)	“Test_4.3”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	“Idle”
Response 4 parameter 1	ActionID (string)	“Test_4.3”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	ProceduralIDs (string)	“P00Y, 0”
Request 5 parameter 1	ActionID (string)	“Test_4.3”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 5 parameter 4	Status (Status)	“Completed”
Response 5 parameter 1	ActionID (string)	“Test_4.3”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	ProceduralIDs (string)	“P00X, 0”, “P00Y, 0”

Table D.41 — Test case 4.4

Identifier	TC4.4	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify the ability to filter by Date parameter.	
Test sequence	1. The tester sends a ReadProceduralIDs request (1) with a random value for Date parameter. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_4.4"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	Date (DateTime)	<<Random value>>
Request 1 parameter 4	Status (Status)	"Idle"
Response 1 parameter 1	ActionID (string)	"Test_4.4"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

D.6.2.6 Test "Stop a procedural element (StopRecipe method)"

D.6.2.6.1 Abstract conformance test

Table D.42 — Test description for StopRecipe method

Test purpose	To verify that the implementation of the StopRecipe method in the SUT is compatible with the standard definition.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker sends a StopRecipe request to the SUT. The SUT generates a response according to the execution test case. In the positive cases, the SUT shall stop the procedural element previously charged.
Request parameters	ActionID, ProceduralID, EntityID.
Response parameters	ActionID, ResponseCode.

D.6.2.6.2 Conformance tests

A set of tests made up of the following are performed:

1) Three (3) positive test; and

2) One (1) negative test.

Table D.43 — Test case 5.1

Identifier	TC5.1
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem

Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The CreateRecipe method shall be tested previously.	
Prerequirement 3	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 4	The procedural element “P00X” shall exist previously in the SUT is in Idle status.	
Test purpose	To verify the ability to stop procedural elements, in Idle or Running status, in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that are being created shall start at least five minutes after the request. 2. The tester sends a ReadProceduralIDs request (2) to the SUT to confirm the status of the procedural element “P00X”. If the returned status is Idle, proceed to the next step. 3. The tester sends a StopRecipe request (3) to the SUT. 4. The SUT returns a “0 – Action successfully executed”. 5. If possible, to confirm at the SUT user interface the absence of the procedural element. 6. The tester sends a ReadProceduralIDs request (4) to the SUT to confirm the status of the procedural element “P00X”. If the returned status is Stopped, proceed to the next step. 7. A second CreateRecipe request (5) shall be send by the tester. The procedural element “P00Y” that are being created shall start at least five minutes after the request. 8. When the start condition is reached, the tester sends a ReadProceduralIDs request (6) to the SUT to confirm the status of the procedural element “P00Y”. If the returned status is Running, proceed to the next step. 9. The tester sends a StopRecipe request (7) to the SUT. 10. The SUT returns a “0 – Action successfully executed”. 11. The tester sends three ReadProceduralIDs requests (8, 9 and 10) to the SUT, one for the next possible status values of the procedural elements (Idle, Running and Completed). "P00Y" and “P00X” does not appear in any response.	
Request 1 parameter 1	ActionID (string)	“Test_5.1”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_5.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_5.1”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Idle”
Response 2 parameter 1	ActionID (string)	“Test_5.1”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 3 parameter 1	ActionID (string)	“Test_5.1”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	ProceduralID (string)	“P00X”
Response 3 parameter 1	ActionID (string)	“Test_5.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Request 4 parameter 1	ActionID (string)	“Test_5.1”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	“Stopped”
Response 4 parameter 1	ActionID (string)	“Test_5.1”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 5 parameter 1	ActionID (string)	“Test_5.1”
Request 5 parameter 2	ProceduralID (string)	“P00Y”
Request 5 parameter 3	EntityID (string)	“E00X”
Request 5 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 5 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 5 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 5 parameter 7	MaxDuration (integer)	“600”
Response 5 parameter 1	ActionID (string)	“Test_5.1”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 6 parameter 1	ActionID (string)	“Test_5.1”
Request 6 parameter 2	EntityID (string)	“E00X”
Request 6 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 6 parameter 4	Status (Status)	“Running”
Response 6 parameter 1	ActionID (string)	“Test_5.1”
Response 6 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 6 parameter 3	ProceduralIDs (string)	“P00Y,0”
Request 7 parameter 1	ActionID (string)	“Test_5.1”
Request 7 parameter 2	EntityID (string)	“E00X”
Request 7 parameter 3	ProceduralID (string)	“P00Y”
Response 7 parameter 1	ActionID (string)	“Test_5.1”
Response 7 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 8 parameter 1	ActionID (string)	“Test_5.1”
Request 8 parameter 2	EntityID (string)	“E00X”
Request 8 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 8 parameter 4	Status (Status)	“Idle”
Response 8 parameter 1	ActionID (string)	“Test_5.1”
Response 8 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 8 parameter 3	ProceduralIDs (string)	"P00Y,0" and “P00X,0” does not appear.
Request 9 parameter 1	ActionID (string)	“Test_5.1”
Request 9 parameter 2	EntityID (string)	“E00X”
Request 9 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 9 parameter 4	Status (Status)	“Running”
Response 9 parameter 1	ActionID (string)	“Test_5.1”

Response 9 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 9 parameter 3	ProceduralIDs (string)	"P00Y,0" and “P00X,0” does not appear.
Request 10 parameter 1	ActionID (string)	“Test_5.1”
Request 10 parameter 2	EntityID (string)	“E00X”
Request 10 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 10 parameter 4	Status (Status)	“Completed”
Response 10 parameter 1	ActionID (string)	“Test_5.1”
Response 10 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 10 parameter 3	ProceduralIDs (string)	"P00Y,0" and “P00X,0” does not appear.

Table D.44 — Test case 5.2

Identifier	TC5.2	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT is in Completed status.	
Test purpose	To verify the ability to stop procedural elements in the entity controlled by the SUT. The SUT can not stop previously completed procedural elements.	
Test sequence	1. The tester sends a ReadProceduralIDs request (1) to the SUT to confirm the status of the procedural element “P00X”. If the returned status is Completed, proceed to the next step. 2. The tester sends a StopRecipe request to the SUT. 3. The SUT returns a “1 – Execution error”.	
Request 1 parameter 1	ActionID (string)	“Test_5.2”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	ProceduralID (string)	“P00X”
Response 1 parameter 1	ActionID (string)	“Test_5.2”
Response 1 parameter 2	ResponseCode (string)	“1 – Execution error”

Table D.45 — Test case 5.3

Identifier	TC5.3
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Negative
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The ReadProceduralIDs method shall be tested previously.
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT and is in Stopped status.
Test purpose	To verify the ability to stop procedural elements in the entity controlled by the SUT. The SUT can not stop previously stopped procedural elements.

Test sequence	1. The tester sends a ReadProceduralIDs request (1) to the SUT to confirm the status of the procedural element "P00X". If the returned status is Stopped, proceed to the next step. 2. The tester sends a StopRecipe request to the SUT. 3. The SUT returns a "1 – Execution error".	
Request 1 parameter 1	ActionID (string)	"Test_5.3"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	ProceduralID (string)	"P00X"
Response 1 parameter 1	ActionID (string)	"Test_5.3"
Response 1 parameter 2	ResponseCode (string)	"1 – Execution error"

Table D.46 — Test case 5.4

Identifier	TC5.4	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall not exist previously in the SUT.	
Test purpose	To verify the ability to filter procedural elements by ProceduralID in the entity controlled by the SUT.	
Test sequence	1. The tester sends a StopRecipe request (1) to the SUT. 2. The SUT returns a "1 – Execution error".	
Request 1 parameter 1	ActionID (string)	"Test_5.4"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	ProceduralID (string)	"P00X"
Response 1 parameter 1	ActionID (string)	"Test_5.4"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error" or "1 - Execution error"

D.6.2.7 Test "Read a procedural element report (ReadReport method)"

D.6.2.7.1 Abstract conformance test

Table D.47 — Test description for ReadReport method

Test purpose	To verify the ReadReport method for the procedural elements executed at the SUT, according to the standard definition.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker sends a ReadReport request to the SUT. The SUT generates a response according to the execution test case.
Request parameters	ActionID, ProceduralID, EntityID
Response parameters	ActionID, ResponseCode, Report.

D.6.2.7.2 Conformance tests

A set of tests made up of the following are performed:

1) Three (3) positive tests; and

2) Two (2) negative tests.

Table D.48 — Test case 6.1

Identifier	TC6.1	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT and is in Idle status.	
Test purpose	To verify the ability to generate the report of an executed procedural element in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that is being created shall start at least five minutes after the request. 2. Before the time start condition, the tester sends a ReadProceduralIDs request (2) to confirm its status. If the returned status is Idle, proceed to the next step. 3. The tester sends a ReadReport request (3) to the SUT. 4. The SUT returns a “0 – Action successfully executed” and the report. 5. When the procedural element start condition is reached, the tester sends a ReadProceduralIDs request (4) to confirm its status. If the returned status is Running, proceed to the next step. 6. The tester sends a ReadReport request (5) to the SUT. 7. The SUT returns a “0 – Action successfully executed” and the report. 8. When the procedural element end condition is reached, the tester sends a ReadProceduralIDs request (6) to confirm its status. If the returned status is Completed, proceed to the next step. 9. The tester sends a ReadReport request (7) to the SUT. 10. The SUT returns a “0 – Action successfully executed” and the report.	
Request 1 parameter 1	ActionID (string)	“Test_6.1”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_6.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_6.1”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Idle”
Response 2 parameter 1	ActionID (string)	“Test_6.1”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	ProceduralIDs (string)	“P00X, 0”

Request 3 parameter 1	ActionID (string)	“Test_6.1”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	ProceduralID (string)	“P00X”
Request 3 parameter 4	Instance (integer)	“0”
Response 3 parameter 1	ActionID (string)	“Test_6.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	Report (string)	“P00X,0, RecipeType,null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null”
Response 3 parameter 4	Status (string)	"Idle"
Request 4 parameter 1	ActionID (string)	“Test_6.1”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	“Running”
Response 4 parameter 1	ActionID (string)	“Test_6.1”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 5 parameter 1	ActionID (string)	“Test_6.1”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	ProceduralID (string)	“P00X”
Request 5 parameter 4	Instance (integer)	“0”
Response 5 parameter 1	ActionID (string)	“Test_6.1”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	Report (string)	“P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), null, null, null, <<irrelevant>>, 100, null, null, null, null, null, null, null, null, null, StartedOperation EventID”
Response 5 parameter 4	Status (string)	"Running"
Request 6 parameter 1	ActionID (string)	“Test_6.1”
Request 6 parameter 2	EntityID (string)	“E00X”
Request 6 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 6 parameter 4	Status (Status)	“Completed”
Response 6 parameter 1	ActionID (string)	“Test_6.1”
Response 6 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 6 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 7 parameter 1	ActionID (string)	“Test_6.1”
Request 7 parameter 2	EntityID (string)	“E00X”
Request 7 parameter 3	ProceduralID (string)	“P00X”
Request 3 parameter 4	Instance (integer)	“0”
Response 7 parameter 1	ActionID (string)	“Test_6.1”
Response 7 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Response 7 parameter 3	Report (string)	“P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 7 parameter 4	Status (string)	"Completed"

Table D.49 — Test case 6.2

Identifier	TC6.2	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 3	The StopRecipe method shall be tested previously.	
Prerequirement 4	The procedural element “P00X” shall exist previously in the SUT in Running status. Its end condition is <<irrelevant>>.	
Test purpose	To verify the ability to generate the report of an executed procedural element in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that is being created shall start at least five minutes after the request. 2. After the “P00X” time start condition, the tester sends a ReadProceduralIDs request (2) to the SUT to confirm its status. If the returned status is Running, proceed to the next step. 3. The tester sends a StopRecipe request (3) to the SUT. 4. The SUT returns a “0 – Action successfully executed”. 5. The tester sends a ReadProceduralIDs request (4) to confirm its status. If the returned status is Stopped, proceed to the next step. 6. The tester sends a ReadReport request (5) to the SUT. 7. The SUT returns a “0 – Action successfully executed” and the report.	
Request 1 parameter 1	ActionID (string)	“Test_6.2”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_6.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_6.2”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Running”
Response 2 parameter 1	ActionID (string)	“Test_6.2”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	ProceduralIDs (string)	“P00X, 0”
Request 3 parameter 1	ActionID (string)	“Test_6.2”
Request 3 parameter 2	EntityID (string)	“E00X”

Request 3 parameter 3	ProceduralID (string)	“P00X”
Response 3 parameter 1	ActionID (string)	“Test_6.2”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 4 parameter 1	ActionID (string)	“Test_6.2”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 4 parameter 4	Status (Status)	“Stopped”
Response 4 parameter 1	ActionID (string)	“Test_6.2”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	ProceduralIDs (string)	“P00X, 0”
Request 5 parameter 1	ActionID (string)	“Test_6.2”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	ProceduralID (string)	“P00X”
Request 5 parameter 4	Instance (integer)	“0”
Response 5 parameter 1	ActionID (string)	“Test_6.2”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	Report (string)	“P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), StopRecipe, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, StoppedOperation EventID”
Response 5 parameter 4	Status (string)	"Stopped"

Table D.50 — Test case 6.3

Identifier	TC6.3	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall not exist in the SUT.	
Test purpose	To verify the ability the parameters filtering in the ReadReport method.	
Test sequence	1. The tester sends a ReadReport request (1) to the SUT. 2. The SUT returns a “2 – Lexical error”.	
Request 1 parameter 1	ActionID (string)	“Test_6.3”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	ProceduralID (string)	“P00X”
Response 1 parameter 1	ActionID (string)	“Test_6.3”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.51 — Test case 6.4

Identifier	TC6.4
Applicable to	All entity types
Verification of	Subsystem interface

Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity “E00X” shall not exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall exist in the SUT.	
Test purpose	To verify the ability the parameters filtering in the ReadReport method.	
Test sequence	1. The tester sends a ReadReport request (1) to the SUT. 2. The SUT returns a “2 – Lexical error”.	
Request 1 parameter 1	ActionID (string)	“Test_6.4”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	ProceduralID (string)	“P00X”
Response 1 parameter 1	ActionID (string)	“Test_6.4”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

D.6.2.7.3 Specific conformance tests procedural elements with calendar

Table D.52 — Test case 6.5

Identifier	TC6.5	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem that supports week calendars.	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT and is in Idle status.	
Test purpose	To verify the ability to generate the report for all instances of an procedural element executed in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that is being created shall start at least five minutes after the request and shall contain a WeekCalendar parameter with at least 3 repetitions. 2. When the procedural element start condition is reached, the tester sends a ReadProceduralIDs request (2) to confirm its status. If the returned status is Running, proceed to the next step. 3. The tester sends a ReadReport request (3) to the SUT for obtain the report of the first instance of the procedural element. 4. The SUT returns a “0 – Action successfully executed” and the requested report. 5. When the procedural element (and all its repetitions) is completed, the tester sends ReadReport requests (4 to 7) to the SUT requesting the report for all the procedural element instances. 6. The SUT returns a “0 – Action successfully executed” and the requested reports.	
Request 1 parameter 1	ActionID (string)	“Test_6.5”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Procedural element with calendar>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”

Request 1 parameter 8	WeekCalendar (string)	<<A possible combination of values>>
Request 1 parameter 9	FinalDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Response 1 parameter 1	ActionID (string)	“Test_6.5”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_6.5”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Running”
Response 2 parameter 1	ActionID (string)	“Test_6.5”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	ProceduralIDs (string)	“P00X, 0”
Request 3 parameter 1	ActionID (string)	“Test_6.5”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	ProceduralID (string)	“P00X”
Request 3 parameter 4	Instance (integer)	“0”
Response 3 parameter 1	ActionID (string)	“Test_6.5”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	Report (string)	“P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), null, null, null, null, null, null, null, null, null, null, null, null, null, StartedOperation EventID,, null”
Response 3 parameter 4	Status (string)	"Running"
Request 4 parameter 1	ActionID (string)	“Test_6.5”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	ProceduralID (string)	“P00X”
Request 4 parameter 4	Instance (integer)	“0”
Response 4 parameter 1	ActionID (string)	“Test_6.5”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	Report (string)	“P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 4 parameter 4	Status (string)	"Completed"
Request 5 parameter 1	ActionID (string)	“Test_6.5”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	ProceduralID (string)	“P00X”
Request 5 parameter 4	Instance (integer)	“1”
Response 5 parameter 1	ActionID (string)	“Test_6.5”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Response 5 parameter 3	Report (string)	“P00X,1, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 5 parameter 4	Status (string)	"Completed"
Request 6 parameter 1	ActionID (string)	“Test_6.5”
Request 6 parameter 2	EntityID (string)	“E00X”
Request 6 parameter 3	ProceduralID (string)	“P00X”
Request 6 parameter 4	Instance (integer)	“2”
Response 6 parameter 1	ActionID (string)	“Test_6.5”
Response 6 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 6 parameter 3	Report (string)	“P00X,2, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 6 parameter 4	Status (string)	"Completed"
Request 7 parameter 1	ActionID (string)	“Test_6.5”
Request 7 parameter 2	EntityID (string)	“E00X”
Request 7 parameter 3	ProceduralID (string)	“P00X”
Request 7 parameter 4	Instance (integer)	“3”
Response 7 parameter 1	ActionID (string)	“Test_6.5”
Response 7 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 7 parameter 3	Report (string)	“P00X,3, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 7 parameter 4	Status (string)	"Completed"

D.6.2.7.4 Specific conformance tests for procedural elements with repetitions

A set of tests made up of the following are performed:

1) One (1) positive test.

Table D.53 — Test case 6.6

Identifier	TC6.6
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem that supports repetitions.

Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 3	The procedural element “P00X” shall exist previously in the SUT and is in Idle status.	
Test purpose	To verify the ability to generate the report for all the instances of an procedural element executed in the entity controlled by the SUT.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00X” that is being created shall start at least five minutes after the request and shall contain at least 3 repetitions of the original procedural element. 2. When the procedural element start condition is reached, the tester sends a ReadProceduralIDs request (2) to confirm its status. If the returned status is Running, proceed to the next step. 3. The tester sends a ReadReport request (3) to the SUT for obtain the report of the first instance. 4. The SUT returns a “0 – Action successfully executed” and the requested report. 5. When the procedural element (and all its repetitions) is completed, the tester sends ReadReport requests (4 to 7) to the SUT requesting the report for all the procedural element instances. 6. The SUT returns a “0 – Action successfully executed” and the requested reports.	
Request 1 parameter 1	ActionID (string)	“Test_6.6”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Procedural element with repetitions enabled>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Request 1 parameter 8	Repetition (boolean)	“1”
Request 1 parameter 9	RepetitionInterval (integer)	“600”
Request 1 parameter 10	RepetitionNumber (integer)	“3”
Response 1 parameter 1	ActionID (string)	“Test_6.6”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_6.6”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Running”
Response 2 parameter 1	ActionID (string)	“Test_6.6”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	ProceduralIDs (string)	“P00X, 0”
Request 3 parameter 1	ActionID (string)	“Test_6.6”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	ProceduralID (string)	“P00X”
Request 3 parameter 4	Instance (integer)	“0”
Response 3 parameter 1	ActionID (string)	“Test_6.6”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Response 3 parameter 3	Report (string)	“P00X,0, RecipeType, initial date (YYYYMMDDhhmmss), null, null, null, null, null, null, null, null, null, null, null, null, null, StartedOperation EventID,, null”
Response 4 parameter 4	Status (string)	"Running"
Request 4 parameter 1	ActionID (string)	“Test_6.6”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	ProceduralID (string)	“P00X”
Request 4 parameter 4	Instance (integer)	“0”
Response 4 parameter 1	ActionID (string)	“Test_6.6”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	Report (string)	“P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 4 parameter 4	Status (string)	"Completed"
Request 5 parameter 1	ActionID (string)	“Test_6.6”
Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	ProceduralID (string)	“P00X”
Request 5 parameter 4	Instance (integer)	“1”
Response 5 parameter 1	ActionID (string)	“Test_6.6”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	Report (string)	“P00X,1, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 5 parameter 4	Status (string)	"Completed"
Request 6 parameter 1	ActionID (string)	“Test_6.6”
Request 6 parameter 2	EntityID (string)	“E00X”
Request 6 parameter 3	ProceduralID (string)	“P00X”
Request 6 parameter 4	Instance (integer)	“2”
Response 6 parameter 1	ActionID (string)	“Test_6.6”
Response 6 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 6 parameter 3	Report (string)	“P00X,2, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 6 parameter 4	Status (string)	"Completed"
Request 7 parameter 1	ActionID (string)	“Test_6.6”
Request 7 parameter 2	EntityID (string)	“E00X”

Request 7 parameter 3	ProceduralID (string)	“P00X”
Request 7 parameter 4	Instance (integer)	“3”
Response 7 parameter 1	ActionID (string)	“Test_6.6”
Response 7 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 7 parameter 3	Report (string)	“P00X,3, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), MaxDuration, duration (s), <<irrelevant>>, 100, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, <<irrelevant>>, StartedOperation EventID, CompletedOperation EventID”
Response 7 parameter 4	Status (string)	"Completed"

D.6.2.7.5 Specific results

The SUT passes the tests if:

1) Its response matches the expected response parameters.

2) In the positive tests, the string <<Report>> has the exact structure established in this document, whereby any information that has not been generated while the ReadReport is being tested shall be assigned an empty field (null).

3) In the negative tests, values other than ActionID and ResponseCode in the response shall not exist.

The SUT does not pass the test if it does not comply with the aforementioned conditions.

D.6.2.8 Test “Read the history of a property (ReadStandardHist method)”

D.6.2.8.1 Abstract conformance test

Table D.54 — Test description for ReadStandardHist method

Test purpose	To verify that the implementation of the ReadStandardHist method in the SUT is compatible with the standard definition.
System under test	Subsystem or coordination broker acting as subsystem.
Test sequence	A tester acting as coordination broker sends a ReadStandardHist request to the SUT. The SUT generates a response according to the execution test case.
Request parameters	ActionID, EntityID, PropertyName, Date, NumHist, Statistics
Response parameters	ActionID, ResponseCode, HistStandardValues

D.6.2.8.2 Conformance tests

A set of tests made up of the following are performed:

1) Four (4) positive test; and

2) Two (2) negative tests.

Table D.55 — Test case 7.1

Identifier	TC7.1	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The property shall be numerical.	
Prerequirement 3	The property is not OpeningDegree	
Test purpose	To verify the ability to read the 24 values standard history for a mandatory property. The reading is performed to obtain the values registered during a natural day in the entity controlled by the SUT.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a “0 – Action successfully executed” and the history values for the requested property. If the property is accumulative and the statistic is different of “Last”, proceed to step 3. 3. For any other statistic, the SUT returns a “1 – Execution error”. This sequence shall be repeated for all mandatory properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	“Test_7.1”
Request 1 to 4 parameter 2	EntityID (string)	“E00X”
Request 1 to 4 parameter 3	PropertyName (string)	<<All those mandatory properties belonging to the entity type>>
Request 1 to 4 parameter 4	Date (DateTime)	Natural day of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	“24”
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined in the standard>>
Response 1 to 4 parameter 1	ActionID (string)	“Test_7.1”
Response 1 to 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed” or “1 – Execution error”.
Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is “0 – Action successfully executed”. The string shall contain 24 fields of the selected statistic, one per hour of the day. Each field is compound by the statistic value requested and it’s timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter of the request.

Table D.56 — Test case 7.2

Identifier	TC7.2
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem or coordination broker acting as subsystem.
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The SUT shall support standard history of 48 values.

Prerequirement 3	The property shall be numerical.	
Test purpose	To verify the ability to read the 48 values standard history for a mandatory property. The reading is performed to obtain the values registered during a natural day in the entity controlled by the SUT.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. If the SUT does not support this type of history, it returns a "3 - Not supported". If the property is accumulative and the statistic is different of "Last", proceed to step 3. 3. For any other statistic, the SUT returns a "1 – Execution error". This sequence shall be repeated for all mandatory properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	"Test_7.2"
Request 1 to 4 parameter 2	EntityID (string)	"E00X"
Request 1 to 4 parameter 3	PropertyName (string)	<<All those mandatory properties belonging to the entity type>>
Request 1 to 4 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	"48"
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined >>
Response 1 to 4 parameter 1	ActionID (string)	"Test_7.2"
Response 1 to 4 parameter 2	ResponseCode (string)	"0 – Action successfully executed", "3 – Not supported" or "1 – Execution error".
Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is "0 – Action successfully executed". The string shall contain 48 fields of the selected statistic, one per half an hour. Each field is compound by the statistic value requested and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter of the request.

Table D.57 — Test case 7.3

Identifier	TC7.3
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem or coordination broker acting as subsystem.
Prerequirement 1	The entity "E00X" shall exist in the SUT.
Prerequirement 2	The SUT shall support standard history of 96 values.
Prerequirement 3	The property shall be numerical.
Test purpose	To verify the ability to read the 96 values standard history for a mandatory property. The reading is performed to obtain the values registered during a natural day in the entity controlled by the SUT.

Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. If the SUT does not support this type of history, it returns a "3 - Not supported". If the property is accumulative and the statistic is different of "Last", proceed to step 3. 3. For any other statistic, the SUT returns a "1 – Execution error". This sequence shall be repeated for all mandatory properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	"Test_7.3"
Request 1 to 4 parameter 2	EntityID (string)	"E00X"
Request 1 to 4 parameter 3	PropertyName (string)	<<All those mandatory properties belonging to the entity type>>
Request 1 to 4 parameter 4	Date (DateTime)	Date of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	"96"
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined >>
Response 1 to 4 parameter 1	ActionID (string)	"Test_7.3"
Response 1 to 4 parameter 2	ResponseCode (string)	"0 – Action successfully executed", "3 – Not supported" or "1 – Execution error".
Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is "0 – Action successfully executed". The string shall contain 96 fields of the selected statistic, one each quarter of an hour. Each field is compound by the statistic value requested and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter of the request.

Table D.58 — Test case 7.4

Identifier	TC7.4	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The property shall be numerical.	
Test purpose	To verify the ability to read the standard history for an extended property. The reading is performed to obtain the values registered during a natural day in the entity controlled by the SUT.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. If the property is accumulative and the statistic is different of "Last", proceed to step 3. 3. For any other statistic, the SUT returns a "1 – Execution error". This sequence shall be repeated for all extended properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	"Test_7.4"
Request 1 to 4 parameter 2	EntityID (string)	"E00X"

Request 1 to 4 parameter 3	PropertyName (string)	<<All those extended properties belonging to the entity type managed by the SUT>>
Request 1 to 4 parameter 4	Date (DateTime)	Date of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	"24"
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined >>
Response 1 to 4 parameter 1	ActionID (string)	"Test_7.4"
Response 1 to 4 parameter 2	ResponseCode (string)	"0 – Action successfully executed" or "1 – Execution error".
Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is "0 – Action successfully executed". The string shall contain 24 fields of the selected statistic, one per hour. Each field is compound by the statistic value requested and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter of the request.

Table D.59 — Test case 7.5

Identifier	TC7.5	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The property shall be represented as a string.	
Test purpose	To verify the ability to read the standard history for a mandatory string type property. The reading is performed to obtain the values registered during a natural day in the entity controlled by the SUT.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. For the statistic "Last", the SUT returns a "0 – Action successfully executed" and the history values for the requested property. If the statistic is different of "Last", proceed to step 3. 3. For any other statistic, the SUT returns a "1 – Execution error". This sequence shall be repeated for all mandatory string properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	"Test_7.5"
Request 1 to 4 parameter 2	EntityID (string)	"E00X"
Request 1 to 4 parameter 3	PropertyName (string)	<<All those mandatory properties belonging to the entity type>>
Request 1 to 4 parameter 4	Date (DateTime)	Date of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	"24"
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined >>
Response 1 to 4 parameter 1	ActionID (string)	"Test_7.5"
Response 1 to 4 parameter 2	ResponseCode (string)	"0 – Action successfully executed" or "1 – Execution error".

Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is “0 – Action successfully executed”. The string shall contain 24 fields of the selected statistic, one per hour. Each field is compound by the statistic value requested and it’s timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter of the request.

Table D.60 — Test case 7.6

Identifier	TC7.6	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The property shall be represented as a string.	
Test purpose	To verify the ability to read the standard history for an extended string type property. The reading is performed to obtain the values registered during a natural day in the entity controlled by the SUT.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a “0 – Action successfully executed” and the history values for the requested property. If the statistic is different of “Last”, proceed to step 3. 3. For any other statistic, the SUT returns a “1 – Execution error”. This sequence shall be repeated for all extended properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	“Test_7.6”
Request 1 to 4 parameter 2	EntityID (string)	“E00X”
Request 1 to 4 parameter 3	PropertyName (string)	<<All those extended properties belonging to the entity type managed by the SUT>>
Request 1 to 4 parameter 4	Date (DateTime)	Date of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	“24”
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined >>
Response 1 to 4 parameter 1	ActionID (string)	“Test_7.6”
Response 1 to 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed” or “1 – Execution error”.
Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is “0 – Action successfully executed”. The string shall contain 24 fields, one per hour. Each field is compound by the statistic value requested and it’s timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter of the request.

Table D.61 — Test case 7.7

Identifier	TC7.7
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Negative

System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The property shall not be supported by the SUT.	
Test purpose	To verify the ability to read the standard history for a non-supported property.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a “3 – Not supported”. This sequence shall be repeated for all extended properties of the entity not supported by the SUT, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	“Test_7.7”
Request 1 to 4 parameter 2	EntityID (string)	“E00X”
Request 1 to 4 parameter 3	PropertyName (string)	<<All those non-supported properties by the SUT>>
Request 1 to 4 parameter 4	Date (DateTime)	Date of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	“24”
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined>>
Response 1 to 4 parameter 1	ActionID (string)	“Test_7.7”
Response 1 to 4 parameter 2	ResponseCode (string)	“3 – Not supported”

Table D.62 — Test case 7.8

Identifier	TC7.8	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The property shall be not available in the SUT.	
Test purpose	To verify the ability to read the standard history for a non-available property.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a “3 – Not available”. This sequence shall be repeated for all non-available properties of the entity, using the four possible Statistics defined (4 possible combinations for each property).	
Request 1 to 4 parameter 1	ActionID (string)	“Test_7.8”
Request 1 to 4 parameter 2	EntityID (string)	“E00X”
Request 1 to 4 parameter 3	PropertyName (string)	<<All those non-available properties in the SUT>>
Request 1 to 4 parameter 4	Date (DateTime)	Date of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	“24”
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined>>
Response 1 to 4 parameter 1	ActionID (string)	“Test_7.8”
Response 1 to 4 parameter 2	ResponseCode (string)	“6 – Not available”

Table D.63 — Test case 7.9

Identifier	TC7.9
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Negative
System under test	Subsystem or coordination broker acting as subsystem.

Prerequirement 1	The entity “E00X” shall not exist in the SUT.	
Test purpose	To verify the ability of the SUT to filter by identifiers, returning an error message when the identifier does not exist.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a “2 – Lexical error”.	
Request 1 parameter 1	ActionID (string)	“Test_7.9”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	<<Irrelevant>>
Request 1 parameter 4	Date (DateTime)	<<Irrelevant>>
Request 1 parameter 5	NumHist (integer)	<<Irrelevant>>
Request 1 parameter 6	Statistics (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	“Test_7.9”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.64 — Test case 7.10

Identifier	TC7.10	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Test purpose	To verify the ability of the SUT to identify a bad parameterization, returning an error message when a parameter has a random value. PropertyName parameter.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a “2 – Lexical error”.	
Request 1 parameter 1	ActionID (string)	“Test_7.10”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	<<Random value>>
Request 1 parameter 4	Date (DateTime)	<<Irrelevant>>
Request 1 parameter 5	NumHist (integer)	<<Irrelevant>>
Request 1 parameter 6	Statistics (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	“Test_7.10”
Response 1 parameter 2	ResponseCode (string)	“2 – Lexical error”

Table D.65 — Test case 7.11

Identifier	TC7.11
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Negative
System under test	Subsystem or coordination broker acting as subsystem.
Prerequirement 1	The entity “E00X” shall exist in the SUT.

Test purpose	To verify the ability of the SUT to identify a bad parameterization, returning an error message when a parameter has a random value. Date parameter. The expected response is the same for any property requested.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_7.11"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Irrelevant>>
Request 1 parameter 4	Date (DateTime)	<<Random value>>
Request 1 parameter 5	NumHist (integer)	<<Irrelevant>>
Request 1 parameter 6	Statistics (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	"Test_7.11"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

Table D.66 — Test case 7.12

Identifier	TC7.12	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify the ability of the SUT to identify a bad parameterization, returning an error message when a parameter has a random value. NumHist parameter. The expected response is the same for any property requested.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "2 – Lexical error" or " 3 – Not supported".	
Request 1 parameter 1	ActionID (string)	"Test_7.12"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Irrelevant>>
Request 1 parameter 4	Date (DateTime)	<<Irrelevant>>
Request 1 parameter 5	NumHist (integer)	<<Random numeric value>>
Request 1 parameter 6	Statistics (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	"Test_7.12"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error" or "3 – Not supported"

Table D.67 — Test case 7.13

Identifier	TC7.13
Applicable to	All entity types
Verification of	Subsystem interface
Test type	Negative
System under test	Subsystem or coordination broker acting as subsystem.
Prerequirement 1	The entity "E00X" shall exist in the SUT.

Test purpose	To verify the ability of the SUT to identify a bad parameterization, returning an error message when a parameter has a random value. Statistics parameter. The expected response is the same for any property requested.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_7.13"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Irrelevant>>
Request 1 parameter 4	Date (DateTime)	<<Irrelevant>>
Request 1 parameter 5	NumHist (integer)	<<Irrelevant>>
Request 1 parameter 6	Statistics (string)	<<Random value>>
Response 1 parameter 1	ActionID (string)	"Test_7.13"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

Table D.68 — Test case 7.14

Identifier	TC7.14	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem or coordination broker acting as subsystem.	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify the ability of the SUT to identify a bad parameterization, returning an error message when the parameter Date has as value the one of the current day or a future day.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_7.14"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<Irrelevant>>
Request 1 parameter 4	Date (DateTime)	Date of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	<<Irrelevant>>
Request 1 parameter 6	Statistics (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	"Test_7.14"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

D.6.2.8.3 Specific results

The SUT passes the test if:

1) Its response matches the expected response parameters.

2) In the positive tests, the string <<HistStandardValues>> has the exact structure established in this document, whereby any information that has not been generated shall be assigned an empty field (null).

3) In the negative tests, shall not exist other values than ActionID and ResponseCode in the response.

The SUT does not pass the test if it does not comply with the aforementioned conditions.

D.6.2.9 Test "Subscription to an entity events (SubscribeEvent method)"

D.6.2.9.1 Abstract conformance test

Table D.69 — Test description for SubscribeEvent method

Test purpose	To verify that the implementation of the SubscribeEvent method in the SUT is compatible with the standard definition.
System under test	Subsystem.
Requirements	This test requires the establishment of a suscriptor to the SUT events. This suscriptor shall be a tester.
Test sequence	A tester acting as coordination broker sends a SubscribeEvent request to the SUT. The request includes a path for event suscription. This path is been Ready to receive events. The SUT generates a response according to the execution test case.
Request parameters	ActionID, EntityID, Path.
Response parameters	ActionID, ResponseCode.

D.6.2.9.2 Conformance tests

A set of tests made up of the following are performed:

1) One (1) positive test; and

2) One (1) negative test.

Table D.70 — Test case 8.1

Identifier	TC8.1
Applicable to	All entity types
Verification of	Subsystem interface and action execution
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity "E00X" shall exist in the SUT.
Prerequirement 2	The Path to subscribe shall be set in the tester.
Test purpose	To verify the ability of the SUT to accept subscriptions to the events occurred in the entity that it controls.

Test sequence	1. The tester sends a SubscribeEvent request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed". 3. The test concludes when an event has been generated and the tester has been receive it.	
Request 1 parameter 1	ActionID (string)	"Test_8.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	Path (string)	Location where the tester is waiting the entity events.
Response 1 parameter 1	ActionID (string)	"Test_8.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"

Table D.71 — Test case 8.2

Identifier	TC8.2	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall not exist in the SUT.	
Test purpose	To verify the ability to filter the subscriptions by inexistent entityIDs.	
Test sequence	1. The tester sends a SubscribeEvent request (1) to the SUT. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_8.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	Path (string)	<<Irrelevant>>
Response 1 parameter 1	ActionID (string)	"Test_8.2"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

D.6.2.10 Test "Unsubscription to an entity events (UnSubscribeEvent method)"

D.6.2.10.1 Abstract conformance test

Table D.72 — Test description for UnSubscribeEvent method

Test purpose	To verify that the implementation of the UnSubscribeEvent method in the SUT is compatible with the standard definition.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker sends a UnSubscribeEvent request to the SUT. The SUT generates a response according to the execution test case.
Request parameters	ActionID, EntityID.
Response parameters	ActionID, ResponseCode.

D.6.2.10.2 Conformance tests

A set of tests made up of the following are performed:

1) One (1) positive test; and

2) One (1) negative test.

Table D.73 — Test case 9.1

Identifier	TC9.1	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The SubscribeEvent method shall be tested previously and the tester shall be subscribed to "E00X" events.	
Test purpose	To verify the ability of the SUT to accept unsubscriptions to the events occurred in the entity that it controls.	
Test sequence	1. The tester sends a UnSubscribeEvent request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed". 3. The test concludes when a new event has been generated and the tester does not receive it. The same tester had receive events when testing SubscribeEvent.	
Request 1 parameter 1	ActionID (string)	"Test_9.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Response 1 parameter 1	ActionID (string)	"Test_9.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"

Table D.74 — Test case 9.2

Identifier	TC9.2	
Applicable to	All entity types	
Verification of	Subsystem interface	
Test type	Negative	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall not exist in the SUT.	
Test purpose	To verify the ability to filter the subscriptions by inexistent entityIDs.	
Test sequence	1. The tester sends a UnSubscribeEvent request (1) to the SUT. 2. The SUT returns a "2 – Lexical error".	
Request 1 parameter 1	ActionID (string)	"Test_9.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Response 1 parameter 1	ActionID (string)	"Test_9.2"
Response 1 parameter 2	ResponseCode (string)	"2 – Lexical error"

D.6.2.11 Test "Generation of an entity event (NewEvent method)"

D.6.2.11.1 Abstract conformance test

Table D.75 — Test description for NewEvent method

Test purpose	To verify that the implementation of the NewEvent method in the SUT is compatible with the standard definition.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker is waiting for events in the Path set using the SubscribeEvent method. The events are received as a SUT request.

Request parameters	ActionID, EntityID, EventValue (EventID, EventName, TimeStamp, Relevance and ProceduralID).
Response parameters[a]	ActionID, ResponseCode.
[a] The response parameters are irrelevant in these tests because they are generated by the tester.	

D.6.2.11.2 Conformance tests

A set of tests made up of the following are performed three (3) positive tests.

Table D.76 — Test case 10.1

Identifier	TC10.1	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall exist previously in the SUT.	
Test purpose	To verify the event generation during a procedural element execution in the entity controlled by the SUT. The expected events are StartedOperation and CompletedOperation. The test also includes the events associated to the property Mode.	
Test sequence	1. When the start condition of "P00X" has been reached, the SUT shall send a NewEvent request (1) with StartedOperation event. 2. The SUT sends a NewEvent request (2) with ModeChanged event. 3. When the end condition of "P00X" has been reached, the SUT sends a NewEvent request (3) with CompletedOperation event. 4. The SUT sends a NewEvent request (4) with ModeChanged event.	
Request 1 parameter 1	ActionID (string)	ActionID assigned by the SUT to the request.
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 1 parameter 4	EventName	"StartedOperation"
Request 1 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 6	Relevance	<<Irrelevant>>
Request 1 parameter 7	ProceduralID	"P00X"
Response 1 parameter 1	ActionID (string)	<<Irrelevant>>
Response 1 parameter 2	ResponseCode (string)	<<Irrelevant>>
Request 2 parameter 1	ActionID (string)	ActionID assigned by the SUT to the request.
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 2 parameter 4	EventName	"ModeChanged, previous value, current value"
Request 2 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 2 parameter 6	Relevance	<<Irrelevant>>
Request 2 parameter 7	ProceduralID	"null"
Response 2 parameter 1	ActionID (string)	<<Irrelevant>>

Response 2 parameter 2	ResponseCode (string)	<<Irrelevant>>
Request 3 parameter 1	ActionID (string)	ActionID assigned by the SUT to the request.
Request 3 parameter 2	EntityID (string)	"E00X"
Request 3 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 3 parameter 4	EventName	"CompletedOperation"
Request 3 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 3 parameter 6	Relevance	<<Irrelevant>>
Request 3 parameter 7	ProceduralID	"P00X"
Response 3 parameter 1	ActionID (string)	<<Irrelevant>>
Response 3 parameter 2	ResponseCode (string)	<<Irrelevant>>
Request 4 parameter 1	ActionID (string)	ActionID assigned by the SUT to the request.
Request 4 parameter 2	EntityID (string)	"E00X"
Request 4 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 4 parameter 4	EventName	"ModeChanged, previous value, current value"
Request 4 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 4 parameter 6	Relevance	<<Irrelevant>>
Request 4 parameter 7	ProceduralID	"null"
Response 4 parameter 1	ActionID (string)	<<Irrelevant>>
Response 4 parameter 2	ResponseCode (string)	<<Irrelevant>>

Table D.77 — Test case 10.2

Identifier	TC10.2	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The procedural element "P00X" shall exist previously in the SUT.	
Test purpose	To verify the event generation during a procedural element execution in the entity controlled by the SUT. The expected events are StartedOperation and StoppedOperation.	
Test sequence	1. When the start condition of "P00X" has been reached, the SUT shall send a NewEvent request (1) with StartedOperation event. 2. Before the end condition of "P00X" has been reached, the tester sends a StopRecipe request (2) to the SUT. 3. The SUT returns a "0 – Action successfully executed". 4. The SUT sends a NewEvent request (3) with StoppedOperation event.	
Request 1 parameter 1	ActionID (string)	"Test_10.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 1 parameter 4	EventName	"StartedOperation"
Request 1 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.

Request 1 parameter 6	Relevance	<<Irrelevant>>
Request 1 parameter 7	ProceduralID	"P00X"
Response 1 parameter 1	ActionID (string)	<<Irrelevant>>
Response 1 parameter 2	ResponseCode (string)	<<Irrelevant>>
Request 2 parameter 1	ActionID (string)	ActionID assigned by the SUT to the request.
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 2 parameter 4	EventName	"StoppedOperation"
Request 2 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 2 parameter 6	Relevance	<<Irrelevant>>
Request 2 parameter 7	ProceduralID	"P00X"
Response 2 parameter 1	ActionID (string)	<<Irrelevant>>
Response 2 parameter 2	ResponseCode (string)	<<Irrelevant>>

Table D.78 — Test case 10.3

Identifier	TC10.3	
Applicable to	All entity types	
Verification of	Subsystem interface and action execution	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Prerequirement 2	The Write method shall be tested previously.	
Test purpose	To verify the event generation occurred by user interaction with the entity controlled by the SUT. The expected events are ActivityStatusChanged and CumulativeVolumeOutSync.	
Test sequence	1. The tester sends a Read request (1) to obtain the value of the writable property. 2. The SUT returns a "0 – Action successfully executed" and the value of the property. 3. The tester sends a Write request (2) to modify a writable property with another valid value. 4. The SUT returns a "0 – Action successfully executed". 5. The SUT shall send a NewEvent request (3) with the generated event and according to the standard definition. This sequence shall be repeated for all events generated by the Write method that according to the standard can occur in the entity controlled by the SUT.	
Request 1 parameter 1	ActionID (string)	"Test_10.3"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	"ActivityStatus"
Response 1 parameter 1	ActionID (string)	"Test_10.3"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value	<<Initial value>>
Request 2 parameter 1	ActionID (string)	"Test_10.3"
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	PropertyName (string)	"ActivityStatus"
Request 2 parameter 4	PropertyName (string)	<<Final value>>

Response 2 parameter 1	ActionID (string)	"Test_10.3"
Response 2 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Request 3 parameter 1	ActionID (string)	ActionID assigned by the SUT to the request.
Request 3 parameter 2	EntityID (string)	"E00X"
Request 3 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 3 parameter 4	EventName	"ActivityStatusChanged, <<Initial value>>, <<Final value>>"
Request 3 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 3 parameter 6	Relevance	<<Irrelevant>>
Request 3 parameter 7	ProceduralID	null
Response 3 parameter 1	ActionID (string)	<<Irrelevant>>
Response 3 parameter 2	ResponseCode (string)	<<Irrelevant>>
Request 4 parameter 1	ActionID (string)	"Test_10.3"
Request 4 parameter 2	EntityID (string)	"E00X"
Request 4 parameter 3	PropertyName (string)	"CumulativeVolumeOut"
Response 4 parameter 1	ActionID (string)	"Test_10.3"
Response 4 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 4 parameter 3	Value	<<Initial value>>
Request 5 parameter 1	ActionID (string)	ActionID assigned by the SUT to the event.
Request 5 parameter 2	EntityID (string)	"E00X"
Request 5 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 5 parameter 4	EventName	"CumulativeVolumeOutSync, <<Initial value>>, <<Final value>>"
Request 5 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 5 parameter 6	Relevance	<<Irrelevant>>
Request 5 parameter 7	ProceduralID	null
Response 5 parameter 1	ActionID (string)	<<Irrelevant>>
Response 5 parameter 2	ResponseCode (string)	<<Irrelevant>>

D.6.3 Functionality verification tests

D.6.3.1 Property Mode

D.6.3.1.1 Abstract conformance test

Table D.79 — Test description for functional tests on property Mode

Test purpose	To verify the functionality implementation of the property Mode.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT in different situations the value of the property Mode.
Request parameters	ActionID, EntityID, PropertyName.
Response parameters	ActionID, ResponseCode, Value, TimeStamp.

D.6.3.1.2 Conformance tests

The conformance tests are composed by two (2) positive test cases.

Table D.80 — Test case 11.1

Identifier	TC11.1	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The Read method shall be tested previously.	
Prerequirement 3	The CreateRecipe method shall be tested previously.	
Prerequirement 4	The SUT does not be executing a procedural element.	
Test purpose	To verify the correct behaviour of the SUT managing the property.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a “0 – Action successfully executed” and the value of the property. 3. The value obtained shall be “Monitoring, reserved”. 4. The tester sends a CreateRecipe request (2). The procedural element “P00X” that are being created shall start at least five minutes after the request. 5. When the start condition is reached the tester sends a Read request (3) to the SUT. 6. The SUT returns a “0 – Action successfully executed” and the value of the property. 7. The value obtained shall be “Programmed, reserved”. 8. When the end condition is reached the tester sends a Read request (4) to the SUT. 9. The SUT returns a “0 – Action successfully executed” and the value of the property. 10. The value obtained shall be “Monitoring, reserved”.	
Request 1 parameter 1	ActionID (string)	“Test_11.1”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	Mode
Response 1 parameter 1	ActionID (string)	“Test_11.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 1 parameter 3	Value	“Monitoring, reserved” or “Monitoring,”
Request 2 parameter 1	ActionID (string)	“Test_11.1”
Request 2 parameter 2	ProceduralID (string)	“P00X”
Request 2 parameter 3	EntityID (string)	“E00X”
Request 2 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 2 parameter 5	RecipeType (string)	<<Any type of procedural element>>
Request 2 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 7	MaxDuration (integer)	“600”
Response 2 parameter 1	ActionID (string)	“Test_11.1”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 3 parameter 1	ActionID (string)	“Test_11.1”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	PropertyName (string)	Mode

Response 3 parameter 1	ActionID (string)	“Test_11.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	Value	“Programmed, reserved” or “Programmed,”
Request 4 parameter 1	ActionID (string)	“Test_11.1”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	PropertyName (string)	Mode
Response 4 parameter 1	ActionID (string)	“Test_11.1”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 4 parameter 3	Value	“Monitoring, reserved” or “Monitoring,”

D.6.3.1.3 Specific conformance tests for SUTs supporting IrrigationRecipe2 operations

Table D.81 — Test case 11.2

Identifier	TC11.2	
Applicable to	HYS, VIH	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The Read method shall be tested previously.	
Prerequirement 3	The CreateRecipe method shall be tested previously.	
Prerequirement 4	The SUT does not be executing a procedural element.	
Prerequirement 5	The SUT shall support IrrigationRecipe2 procedural elements.	
Test purpose	To verify the correct behaviour of the SUT managing the property.	
Test sequence	1. The tester sends a CreateRecipe request (1). The procedural element “P00Y” that is being created shall start at least five minutes after the request. 2. The tester sends a Read request (2) to the SUT before the IrrigationRecipe2 procedural element type starts. 3. The SUT returns a “0 – Action successfully executed” and the value of the property. 4. The value obtained shall be “Monitoring, reserved”. 5. The tester sends a Read request (3) to the SUT after the IrrigationRecipe2 procedural element type starts. 6. The SUT returns a “0 – Action successfully executed” and the value of the property. 7. The value obtained shall be “OnDemand, reserved”.	
Request 1 parameter 1	ActionID (string)	“Test_11.1”
Request 1 parameter 2	ProceduralID (string)	“P00Y”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	“HYS” or “VIH”
Request 1 parameter 5	RecipeType (string)	“IrrigationRecipe2”
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_11.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_11.1”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	PropertyName (string)	Mode
Response 2 parameter 1	ActionID (string)	“Test_11.1”

Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	Value	“Monitoring, reserved” or “Monitoring,”
Request 3 parameter 1	ActionID (string)	“Test_11.1”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	PropertyName (string)	Mode
Response 3 parameter 1	ActionID (string)	“Test_11.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	Value	“OnDemand, reserved” or “OnDemand,”

D.6.3.2 Property ActivityStatus

D.6.3.2.1 Abstract conformance test

Table D.82 — Test description for functional tests on property ActivityStatus

Test purpose	To verify the functionality implementation of the property Mode.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT in different situations the value of the property Mode.
Request parameters	ActionID, EntityID, PropertyName.
Response parameters	ActionID, ResponseCode, Value, TimeStamp.

D.6.3.2.2 Conformance tests

The conformance tests are composed by two (2) positive test cases.

Table D.83 — Test case 12.1

Identifier	TC12.1
Applicable to	All entity types
Verification of	SUT functionality
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The Read method shall be tested previously.
Prerequirement 3	The Write method shall be tested previously.
Prerequirement 4	The CreateRecipe method shall be tested previously.
Prerequirement 5	The SUT does not be executing a procedural element.
Test purpose	To verify the correct behaviour of the SUT managing the property.

Test sequence	1. The tester sends a Write request (1) to the SUT, sending a “Active, Ready”. 2. The SUT returns a “0 – Action successfully executed”. 3. The tester sends a Read request (2) to the SUT. 4. The SUT returns a “0 – Action successfully executed” and the value of the property. 5. The value obtained shall be “Active, Ready”. 6. The tester sends a CreateRecipe request (3). The procedural element shall be executed succesfully by the SUT. 7. The tester sends a Write request (4) to the SUT, sending a “Inactive, Ready”. 8. The SUT returns a “0 – Action successfully executed”. 9. The tester sends a Read request (5) to the SUT. 10. The SUT returns a “0 – Action successfully executed” and the value of the property. 11. The value obtained shall be “Inactive, Ready”. 12. The tester sends a CreateRecipe request (6) to the SUT. 13. The SUT returns a “1 – Execution error”. 14. The tester sends a Write request (7) to the SUT, sending a “Inactive, OutOfOrder” to the SUT. 15. The SUT returns a “0 – Action successfully executed”. 16. The tester sends a Read request (8) to the SUT. 17. The SUT returns a “0 – Action successfully executed” and the value of the property. 18. The value obtained shall be “Inactive, OutOfOrder”. 19. The tester sends a CreateRecipe request (9). 20. The SUT returns a “1 – Execution error”.	
Request 1 parameter 1	ActionID (string)	“Test_12.1”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	ActivityStatus
Request 1 parameter 4	Value (string)	“Active, Ready”
Response 1 parameter 1	ActionID (string)	“Test_12.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_12.1”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	PropertyName (string)	ActivityStatus
Response 2 parameter 1	ActionID (string)	“Test_12.1”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	Value	“Active, Ready”
Request 3 parameter 1	ActionID (string)	“Test_12.1”
Request 3 parameter 2	ProceduralID (string)	“P00X”
Request 3 parameter 3	EntityID (string)	“E00X”
Request 3 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 3 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 3 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 3 parameter 7	MaxDuration (integer)	“600”
Response 3 parameter 1	ActionID (string)	“Test_12.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 4 parameter 1	ActionID (string)	“Test_12.1”
Request 4 parameter 2	EntityID (string)	“E00X”
Request 4 parameter 3	PropertyName (string)	ActivityStatus
Request 4 parameter 4	Value (string)	“Inactive, Ready”
Response 4 parameter 1	ActionID (string)	“Test_12.1”
Response 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 5 parameter 1	ActionID (string)	“Test_12.1”

Request 5 parameter 2	EntityID (string)	“E00X”
Request 5 parameter 3	PropertyName (string)	ActivityStatus
Response 5 parameter 1	ActionID (string)	“Test_12.1”
Response 5 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 5 parameter 3	Value	“Inactive, Ready”
Request 6 parameter 1	ActionID (string)	“Test_12.1”
Request 6 parameter 2	ProceduralID (string)	“P00Y”
Request 6 parameter 3	EntityID (string)	“E00X”
Request 6 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 6 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 6 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 6 parameter 7	MaxDuration (integer)	“600”
Response 6 parameter 1	ActionID (string)	“Test_12.1”
Response 6 parameter 2	ResponseCode (string)	“1 – Execution error”
Request 7 parameter 1	ActionID (string)	“Test_12.1”
Request 7 parameter 2	EntityID (string)	“E00X”
Request 7 parameter 3	PropertyName (string)	ActivityStatus
Request 7 parameter 4	Value (string)	“Active, OutOfOrder”
Response 7 parameter 1	ActionID (string)	“Test_12.1”
Response 7 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 8 parameter 1	ActionID (string)	“Test_12.1”
Request 8 parameter 2	EntityID (string)	“E00X”
Request 8 parameter 3	PropertyName (string)	ActivityStatus
Response 8 parameter 1	ActionID (string)	“Test_12.1”
Response 8 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 8 parameter 3	Value	“Inactive, OutOfOrder”
Request 9 parameter 1	ActionID (string)	“Test_12.1”
Request 9 parameter 2	ProceduralID (string)	“P00Z”
Request 9 parameter 3	EntityID (string)	“E00X”
Request 9 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 9 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 9 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 9 parameter 7	MaxDuration (integer)	“600”
Response 9 parameter 1	ActionID (string)	“Test_12.1”
Response 9 parameter 2	ResponseCode (string)	“1 – Execution error”

Table D.84 — Test case 12.2

Identifier	TC12.2
Applicable to	All entity types
Verification of	SUT functionality
Test type	Positive
System under test	Subsystem
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The Read method shall be tested previously.
Prerequirement 3	The NewEvent method shall be tested previously.

Prerequirement 4	The tester shall be subscribed to the events of the entity controlled by the SUT.	
Prerequirement 5	The SUT does not be executing a procedural element.	
Test purpose	To verify the correct behaviour of the SUT managing the property when a communication error occurs.	
Test sequence	1. A communication error situation shall be forced in the SUT. The SUT electronic shall remain shutted off at least during the time required by the SUT to determine the communication failure. When the failure status has been reached at the SUT, proceed to step 2. 2. The tester sends a Read request (1) to the SUT. 3. The SUT returns a "0 – Action successfully executed" and the value of the property. 4. The value obtained shall be "Unknown". 5. The SUT sends a NewEvent request (3) corresponding to a ActivityStatusChange event.	
Request 1 parameter 1	ActionID (string)	"Test_12.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	ActivityStatus
Response 1 parameter 1	ActionID (string)	"Test_12.2"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value (string)	"Active, Unknown"
Request 2 parameter 1	ActionID (string)	ActionID assigned by the SUT.
Request 2 parameter 2	EntityID (string)	"E00X"
Request 2 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 2 parameter 4	EventName	"ActivityStatusChanged,<<PreviousValue>>, Unknown"
Request 2 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 2 parameter 6	Relevance	<<Irrelevant>>
Request 2 parameter 7	ProceduralID	null

D.6.3.3 Property SystemStatus

D.6.3.3.1 Abstract conformance test

Table D.85 — Test description for functional tests on property SystemStatus

Test purpose	To verify the functionality implementation of the property SystemStatus.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT in different situations the value of the property SystemStatus.
Request parameters	ActionID, EntityID, PropertyName.
Response parameters	ActionID, ResponseCode, Value, TimeStamp.

D.6.3.3.2 Conformance tests

The conformance tests are composed by one (1) positive test cases.

Table D.86 — Test case 13.1

Identifier	TC13.1
Applicable to	All entity types
Verification of	SUT functionality

Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The Read method shall be tested previously.	
Prerequirement 3	The NewEvent method shall be tested previously.	
Prerequirement 4	The tester shall be subscribed to the events of the entity controlled by the SUT.	
Prerequirement 5	The SUT does not be executing a procedural element.	
Test purpose	To verify the correct behaviour of the SUT managing the property when a communication error occurs.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a “0 – Action successfully executed” and the value of the property. 3. A communication error situation shall be forced in the SUT. The SUT electronic shall remain shutted off at least during the time required by the SUT to determine the communication failure. When the failure status has been reached at the SUT, proceed to step 4. 4. The tester sends a Read request (2) to the SUT. 5. The SUT returns a “0 – Action successfully executed” and the value of the property. 6. The value obtained shall be “Unknown, Unknown, Unknown”. 7. The SUT sends a NewEvent request (3) corresponding to a SystemStatusUnknown event.	
Request 1 parameter 1	ActionID (string)	“Test_13.4”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	SystemStatus
Response 1 parameter 1	ActionID (string)	“Test_13.4”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 1 parameter 3	Value (string)	<<Evaluation,Characterization,Classification>>
Request 2 parameter 1	ActionID (string)	“Test_13.4”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	PropertyName (string)	SystemStatus
Response 2 parameter 1	ActionID (string)	“Test_13.4”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	Value (string)	“Unknown, unknown, unknown”
Request 2 parameter 1	ActionID (string)	ActionID assigned by the SUT.
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 2 parameter 4	EventName	“SystemStatusChanged, <<PreviousEvaluation, PreviousCharacterization, PreviousClassification>>, Unknown, Unknown, Unknown”
Request 2 parameter 5	TimeStamp	Date of the event “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 6	Relevance	<<Irrelevant>>
Request 2 parameter 7	ProceduralID	null

D.6.3.4 Property OpeningDegree

D.6.3.4.1 Abstract conformance test

Table D.87 — Test description for functional tests on property OpeningDegree

Test purpose	To verify the functionality implementation of the property OpeningDegree.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT in different situations the value of the property OpeningDegree.
Request parameters	ActionID, EntityID, PropertyName.
Response parameters	ActionID, ResponseCode, Value, TimeStamp.

D.6.3.4.2 Conformance tests

The conformance tests are composed by two (2) positive test cases.

Table D.88 — Test case 14.1

Identifier	TC14.1	
Applicable to	All entity types where OpeningDegree is a property.	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The CreateRecipe method shall be tested previously.	
Prerequirement 3	The SUT does not be executing a procedural element.	
Test purpose	To verify the correct behaviour of the SUT managing the property.	
Test sequence	1. The tester sends a Read request (1) to the SUT to obtain the value of the property OpeningDegree. 2. The SUT returns a “0 – Action successfully executed” and the value of the property. 3. The tester sends a CreateRecipe request (2). 4. When the start condition of the procedural element is reached, the tester sends a Read request (3) to the SUT to obtain the value of the property OpeningDegree. 5. The SUT returns a “0 – Action successfully executed” and the value of the property. The expected value is “≠0”.	
Request 1 parameter 1	ActionID (string)	“Test_14.1”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	OpeningDegree
Response 1 parameter 1	ActionID (string)	“Test_14.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 1 parameter 3	Value	“0”
Request 2 parameter 1	ActionID (string)	“Test_14.1”
Request 2 parameter 2	ProceduralID (string)	“P00X”
Request 2 parameter 3	EntityID (string)	“E00X”
Request 2 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 2 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 2 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 7	MaxDuration (integer)	“600”

Response 2 parameter 1	ActionID (string)	“Test_14.1”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 3 parameter 1	ActionID (string)	“Test_14.1”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	PropertyName (string)	OpeningDegree
Response 3 parameter 1	ActionID (string)	“Test_14.1”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	Value	“≠0”

Table D.89 — Test case 14.2

Identifier	TC14.2	
Applicable to	All entity types where OpeningDegree is a property	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Test purpose	To verify the ability to generate standard histories for the OpeningDegree property. The test contains as many requests as NumHist (24, 48 and 96) supported by the SUT.	
Test sequence	1. The tester sends a ReadStandardHist request (1) to the SUT. 2. If the statistic of the request is different to “Average”, the SUT returns a “0 – Action successfully executed” and the history values for the requested property. 3. For “Average” statistic, the SUT returns a “1 – Execution error”. This sequence shall be repeated for all possible NumHist supported by the SUT, using the four possible Statistics defined.	
Request 1 to 4 parameter 1	ActionID (string)	“TC14.2”
Request 1 to 4 parameter 2	EntityID (string)	“E00X”
Request 1 to 4 parameter 3	PropertyName (string)	OpeningDegree
Request 1 to 4 parameter 4	Date (DateTime)	Natural day of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 to 4 parameter 5	NumHist (integer)	“24” or “48” or “96”
Request 1 to 4 parameter 6	Statistics (string)	<<All posible statistics defined>>
Response 1 to 4 parameter 1	ActionID (string)	“TC14.2”
Response 1 to 4 parameter 2	ResponseCode (string)	“0 – Action successfully executed” or “1 – Execution error”.
Response 1 to 4 parameter 3	HistStandardValues (string)	This parameter only appears if response is “0 – Action successfully executed”. The string shall contain 24 fields of the selected statistic, one per hour. Each field is compound by the value of the property and it’s timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

D.6.3.5 Flow with closed valve event

D.6.3.5.1 Abstract conformance test

Table D.90 — Test description for functional tests on InternalFlowOutIsNot0OpeningDegreeIs0 event

Test purpose	To verify the generation of the flow with closed valve event.
System under test	Subsystem.
Test sequence	An event of flow with closed valve is generated or simulated on the test module. The SUT sends the event to the subscribed tester.
Request parameters	ActionID, EntityID, EventValue (EventID, EventName, TimeStamp, Relevance and ProceduralID).
Response parameters[a]	ActionID, ResponseCode.
[a] The response parameters are irrelevant in these tests because they are generated by the tester.	

D.6.3.5.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.91 — Test case 15.1

Identifier	TC15.1	
Applicable to	All entity types where flow with closed valve has been defined as event.	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The NewEvent method shall be tested previously.	
Test purpose	To verify the generation of an event of detected flow with closed valve.	
Test sequence	1. After the event is generated at the test module, the SUT sends a NewEvent request (1) with a InternalFlowOutIsNot0OpeningDegreeIs0 event.	
Request 1 parameter 1	ActionID (string)	ActionID assigned by the SUT.
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 1 parameter 4	EventName	“InternalFlowOutIsNot0OpeningDegreeIs0,,”
Request 1 parameter 5	TimeStamp	Date of the event “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 6	Relevance	<<Irrelevant>>
Request 1 parameter 7	ProceduralID	null

D.6.3.6 No flow with open valve event

D.6.3.6.1 Abstract conformance test

Table D.92 — Test description for functional tests on InternalFlowOutIs0OpeningDegreeIsNot0 event

Test purpose	To verify the generation of the no flow with open valve event.
System under test	Subsystem.
Test sequence	An event of flow absence with open valve is generated on the test module. The SUT sends the event to the subscribed tester.
Request parameters	ActionID, EntityID, EventValue (EventID, EventName, TimeStamp, Relevance and ProceduralID).
Response parameters[a]	ActionID, ResponseCode.
[a] The response parameters are irrelevant in these tests because they are generated by the tester.	

D.6.3.6.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.93 — Test case 16.1

Identifier	TC16.1	
Applicable to	All entity types where no flow with open valve had been defined as an event.	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The procedural element “P00X” shall exist previously in the SUT.	
Prerequirement 3	The NewEvent method shall be tested previously.	
Test purpose	To verify the generation of an event of flow absence with open valve in the SUT.	
Test sequence	1. After the event is generated at the test module (P00X in execution without CumulativeVolumeOut variation), the SUT shall send a request (1) with a InternalFlowOutIs0OpeningDegreeIsNot0 event.	
Request 1 parameter 1	ActionID (string)	ActionID assigned by the SUT.
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 1 parameter 4	EventName	“InternalFlowOutIs0OpeningDegreeIsNot0,,”
Request 1 parameter 5	TimeStamp	Date of the event “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 6	Relevance	<<Irrelevant>>
Request 1 parameter 7	ProceduralID	“P00X”

D.6.3.7 Execution of procedural elements

D.6.3.7.1 Abstract conformance test

Table D.94 — Test description for functional tests for irrigation procedural elements

Test purpose	To verify the execution of the procedural elements.
System under test	Subsystem
Test sequence	A tester acting as coordination broker requests to the SUT the execution of a procedural element with a certain parameterization.
Request parameters	ActionID, ProceduralID, EntityID, EntityType, RecipeType and OperationParameters.
Response parameters	ActionID, ResponseCode.

D.6.3.7.2 Conformance tests

The conformance tests are composed by three (3) positive test cases.

Table D.95 — Test case 17.1

Identifier	TC17.1	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Test purpose	To verify the behaviour of the SUT executing a procedural element ended by time.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make a procedural element executable, attending to the restrictions established by its design time between communications. 2. The end condition is MaxDuration. 3. The SUT returns a “0 – Action successfully executed”. 4. The SUT shall execute the procedural element and end it attending to the MaxDuration parameter introduced.	
Request 1 parameter 1	ActionID (string)	“Test_17.1”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“600”
Response 1 parameter 1	ActionID (string)	“Test_17.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”

Table D.96 — Test case 17.2

Identifier	TC17.2
Applicable to	All entity types
Verification of	SUT functionality

Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The CreateRecipe method shall be tested previously.	
Prerequirement 3	The ReadProceduralIDs method shall be tested previously.	
Test purpose	To verify the correct behaviour of the SUT executing a procedural element ended by another end condition different of time.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make a procedural element executable, attending to the restrictions established by its design time between communications. 2. The end condition is another than MaxDuration, so MaxDuration can not been reached before. 3. The SUT returns a “0 – Action successfully executed”. 4. The SUT shall execute the procedural element and end when the end condition parameterized is reached. 5. The tester sends a ReadProceduralIDs request (2) to verify that the Completed Status has been reached before the MaxDuration condition. 6. If the status is not Completed, proceed to a ReadProceduralIDs request (3) to verify that the Status is Running 7. The test ends when Completed Status has been reached.	
Request 1 parameter 1	ActionID (string)	“Test_17.2”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“86400”
Request 1 parameter 8	<<Another end condition allowed by the procedural element selected>>	<<Valid value for the selected end condition>>
Response 1 parameter 1	ActionID (string)	“Test_17.2”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_17.2”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 4	Status (Status)	“Completed”
Response 2 parameter 1	ActionID (string)	“Test_17.2”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	ProceduralIDs (string)	“P00X,0”
Request 3 parameter 1	ActionID (string)	“Test_17.2”
Request 3 parameter 2	EntityID (string)	“E00X”
Request 3 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 3 parameter 4	Status (Status)	“Running”
Response 3 parameter 1	ActionID (string)	“Test_17.2”
Response 3 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 3 parameter 3	ProceduralIDs (string)	“Empty”

Table D.97 — Test case 17.3

Identifier	TC17.3	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The entity “E00X” shall exist in the SUT.	
Prerequirement 2	The CreateRecipe method shall be tested previously.	
Prerequirement 3	The ReadProceduralIDs method shall be tested previously.	
Prerequirement 4	The NewEvent method shall be tested previously.	
Test purpose	To verify the correct behaviour of the SUT executing an procedural element with active monitorizations.	
Test sequence	1. The tester sends a CreateRecipe request (1) to the SUT. The StartDate should respect the minimum time defined at the SUT to make an procedural element executable, attending to the restrictions established by its design time between communications. The request includes one supported monitorization, attending to the ICS document. The monitoring includes the setup of the stop condition (CmdHH or CmdLL=1) to verify the complete functionality. 2. The SUT returns a “0 – Action successfully executed”. 3. The execution does not end before the threshold parameterized has been reached. 4. When the execution ends, the tester sends a ReadProceduralIDs request (2) to verify the status Stopped of the procedural element. 5. The SUT sends a NewEvent request (3) with a StoppedOperation. 6. The SUT sends a NewEvent request (4) with a ThresholdHHTrriggered or ThresholdLLTrriggered event. This test shall be repeated for all SUT supported monitorings and for all configurable thresholds of each monitoring.	
Request 1 parameter 1	ActionID (string)	“Test_17.3”
Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X”
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“86400”
Request 1 parameter 8	MonCumulativeVolumeOut or MonInternalFlowOut or MonPressureOut or MonCumulativeVolumeIn or MonInternalFlowIn or MonPressureIn (boolean)	“1”
Request 1 parameter 9	ThresholdHH or ThresholdLL	<<Value>>
Request 1 parameter 10	CmdHH or CmdLL	“1”
Request 1 parameter 11*	TimeHH or TimeLL	“30”
Response 1 parameter 1	ActionID (string)	“Test_17.3”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_17.3”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	Date (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.

Request 2 parameter 4	Status (Status)	"Stopped"
Request 3 parameter 1	ActionID (string)	ActionID assigned by the SUT.
Request 3 parameter 2	EntityID (string)	"E00X"
Request 3 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 3 parameter 4	EventName	"StoppedOperation,,"
Request 3 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 3 parameter 6	Relevance	<<Irrelevant>>
Request 3 parameter 7	ProceduralID	null
Request 4 parameter 1	ActionID (string)	ActionID assigned by the SUT.
Request 4 parameter 2	EntityID (string)	"E00X"
Request 4 parameter 3	EventID (string)	ID assigned by the SUT to the event.
Request 4 parameter 4	EventName	"ThresholdHHTriggered, PropertyName,CurrentValue" or "ThresholdLLTriggered, PropertyName,CurrentValue"
Request 4 parameter 5	TimeStamp	Date of the event "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 4 parameter 6	Relevance	<<Irrelevant>>
Request 4 parameter 7	ProceduralID	null
* Request 1 parameter 11 is not included in the monitoring of the CumulativeVolumeOut property.		

D.6.3.8 Numerical properties management

D.6.3.8.1 Abstract conformance test

Table D.98 — Test description for functional tests for numerical properties management

Test purpose	To verify the correct management of numerical properties.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT the value of all numerical properties related to the entity type controlled by it. A reference value for each of the numerical properties are generated at the test module. The value returned by the SUT shall be the expected for the measure units defined in this document and for the reference values setted.
Request parameters	ActionID, EntityID, PropertyName.
Response parameters	ActionID, ResponseCode, Value, TimeStamp.

D.6.3.8.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.99 — Test case 18.1

Identifier	TC18.1
Applicable to	All entity types
Verification of	SUT functionality
Test type	Positive
System under test	Subsystem

Prerequirement 1	The entity "E00X" shall exist in the SUT.	
Test purpose	To verify the correct formating of the numerical properties (mandatory and extended) of the entity controlled by the SUT.	
Test sequence	1. The tester sends a Read request (1) to the SUT. 2. The SUT returns a "0 – Action successfully executed" and the value of the requested property. 3. The value obtained shall match (applying if required the unit conversion) with the one available in the SUT user interface. The SUT shall return the value of the property using the unit defined on the standard.	
Request 1 parameter 1	ActionID (string)	"Test_18.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All mandatory numerical properties and all supported extended properties>>
Response 1 parameter 1	ActionID (string)	"Test_18.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	Value	<<Value>>

D.6.3.9 Standard history content validation

D.6.3.9.1 Abstract conformance test

Table D.100 — Test description for standard history validation

Test purpose	To verify the content included on a standard history.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT the standard history of all properties. The value returned by the SUT shall match with the ones expected, attending to the standard definition.
Request parameters	ActionID, EntityID, PropertyName, Date, NumHist, Statistics
Response parameters	ActionID, ResponseCode, HistStandardValues

D.6.3.9.2 Conformance tests

The conformance tests are composed by nine (9) positive test cases.

Table D.101 — Test case 19.1

Identifier	TC19.1
Applicable to	All entity types
Verification of	SUT functionality
Test type	Positive
System under test	Subsystem
Test purpose	To verify the content of the standard history for text properties, whether mandatory or extended.

Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. During the sample day, changes shall be forced in all text type properties. At least ten (10) changes shall be performed, being four (4) of them during the same hour. The previous and the current property values and the timestamp of each change shall be registered. 3. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 4. The SUT returns a "0 – Action successfully executed" and the standard history for the requested property. 5. The standard history values shall match with the ones registered. This aproaching should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All text type properties belonging to the entity controlled by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"24"
Request 1 parameter 6	Statistics (string)	"Last"
Response 1 parameter 1	ActionID (string)	"Test_19.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 24 fields of the selected statistic, one per hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.102 — Test case 19.2

Identifier	TC19.2	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Test purpose	To verify the correct generation of standard history content for accumulative numerical properties, whether mandatory or extended.	
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. The tester sends a CreateRecipe request (1) to the SUT. The created procedural element is executed integrally during the sample day and should have a four (4) hours duration. 3. The SUT returns a "0 - Action successfully executed". 4. The value of each property of the entity controlled by the SUT is registered for the o'clock hour previous to the StartDate parameterization. For example, if the procedural element starts at 12:40pm the property value is registered at 12:00pm. During the procedural element execution, the property value is registered all quarter of hour (following the example, at 12:45pm, 13:00pm, 13:15pm up to 16:30pm). After the end condition has been reached, the value is registered at the posterior o'clock hour (following the example, at 17:00pm). 5. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 6. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 7. The standard history values should be approached to the ones registered. This aproaching should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.2"

Request 1 parameter 2	ProceduralID (string)	“P00X”
Request 1 parameter 3	EntityID (string)	“E00X
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	“YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	“28800”
Response 1 parameter 1	ActionID (string)	“Test_19.2”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Request 2 parameter 1	ActionID (string)	“Test_19.2”
Request 2 parameter 2	EntityID (string)	“E00X”
Request 2 parameter 3	PropertyName (string)	<<All those accumulative numerical properties belonging to the entity type controlled by the SUT>>
Request 2 parameter 4	Date (DateTime)	Natural day of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.
Request 2 parameter 5	NumHist (integer)	“24”
Request 2 parameter 6	Statistics (string)	“Last”
Response 2 parameter 1	ActionID (string)	“Test_19.2”
Response 2 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 2 parameter 3	HistStandardValues (string)	The string shall contain 24 fields of the selected statistic, one per hour. Each field is compound by the value of the property and it’s timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.103 — Test case 19.3

Identifier	TC18.3	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Test purpose	To verify the correct generation of standard history content for non-accumulative numerical type properties, whether mandatory or extended.	
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. During the sample day, changes shall be forced in all numerical properties. At least ten (10) changes times are performed, being four (4) of them during the same hour. The previous and the current property values and the timestamp of each change shall be registered. 3. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 4. The SUT returns a “0 – Action successfully executed” and the history values for the requested property. 5. The standard history values shall be approached to the ones registered. This aproaching should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	“Test_19.3”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	<<All those numerical non-accumulative properties belonging to the entity controlled by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history “YYYYMMDDhhmmss±hhmm” in UTC ISO 8106.

Request 1 parameter 5	NumHist (integer)	"24"
Request 1 parameter 6	Statistics (string)	<<All posible statistics defined>>
Response 1 parameter 1	ActionID (string)	"Test_19.3"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	This parameter only appears if response is "0 – Action successfully executed". The string shall contain 24 fields of the selected statistic, one per hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.104 — Test case 19.4

Identifier	TC18.4	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The SUT shall support standard history of 48 values.	
Test purpose	To verify the correct generation of 48 values standard history content for text properties, whether mandatory or extended.	
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. During the sample day, changes shall be forced in all text properties. At least ten (10) changes times are performed, being four (4) of them during the same hour. The previous and the current property values and the timestamp of each change shall be registered. 3. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 4. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 5. The standard history values should match with the ones registered. It should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.4"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those text properties belonging to the entity type controlled by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"48"
Request 1 parameter 6	Statistics (string)	"Last"
Response 1 parameter 1	ActionID (string)	"Test_19.4"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 48 fields of the selected statistic, one per half an hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.105 — Test case 19.5

Identifier	TC19.5	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Prerequirement 1	The SUT shall support standard history of 96 values.	
Test purpose	To verify the correct generation of 96 values standard history content for text properties, whether mandatory or extended.	
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. During the sample day, changes shall be forced in all text type properties. At least ten (10) changes times are performed, being four (4) of them during the same hour. The previous and the current property values and the timestamp of each change shall be registered. 3. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 4. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 5. The standard history values should match with the ones registered. It should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.5"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those numerical type properties belonging to the entity type controlled by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"96"
Request 1 parameter 6	Statistics (string)	"Last"
Response 1 parameter 1	ActionID (string)	"Test_19.5"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 96 fields of the selected statistic, one each quarter of an hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.106 — Test case 19.6

Identifier	TC19.6
Applicable to	All entity types
Verification of	SUT functionality
Test type	Positive
System under test	Subsystem
Prerequirement 1	The SUT shall support standard history of 48 values.
Test purpose	To verify the correct generation of 48 values standard history content for accumulative numerical properties, whether mandatory or extended.
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. The tester sends a CreateRecipe request (1) to the SUT. The created procedural element is

	executed integrally during the sample day and should have a four (4) hours duration. 3. The SUT returns a "0 - Action successfully executed". 4. The value of each property of the entity controlled by the SUT shall be registered for the o'clock hour previous to the StartDate parameterization. For example, if the procedural element starts at 12:40pm the property value shall be registered at 12:00pm. During the procedural element execution, the property value shall be registered all quarter of hour (following the example, at 12:45pm, 13:00pm, 13:15pm up to 16:30pm). After the end condition has been reached, the value shall be registered at the posterior o'clock hour (following the example, at 17:00pm). 5. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 6. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 7. The standard history values should be approached to the ones registered. This aproaching should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.6"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those numerical type properties belonging to the entity type controlled by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"48"
Request 1 parameter 6	Statistics (string)	"Last"
Response 1 parameter 1	ActionID (string)	"Test_19.6"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 48 fields of the selected statistic, one per half an hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.107 — Test case 19.7

Identifier	TC19.7
Applicable to	All entity types
Verification of	SUT functionality
Test type	Positive
System under test	Subsystem
Prerequirement 1	The SUT shall support standard history of 96 values.
Test purpose	To verify the correct generation of 96 values standard history content for accumulative numerical properties, whether mandatory or extended.
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. The tester sends a CreateRecipe request (1) to the SUT. The created procedural element is executed integrally during the sample day and should have a four (4) hours duration. 3. The SUT returns a "0 - Action successfully executed". 4. The value of each property of the entity controlled by the SUT shall be registered for the o'clock hour previous to the StartDate parameterization. For example, if the procedural element starts at 12:40pm the property value shall be registered at 12:00pm. During the procedural element execution, the property value shall be registered all quarter of hour (following the example, at 12:45pm, 13:00pm, 13:15pm up to 16:30pm). After the end condition has been reached, the value shall be registered at the posterior o'clock hour (following the example, at 17:00pm).

	5. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 6. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 7. The standard history values should be approached to the ones registered. This aproaching should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.7"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those numerical type properties belonging to the entity type managed by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"96"
Request 1 parameter 6	Statistics (string)	"Last"
Response 1 parameter 1	ActionID (string)	"Test_19.7"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 96 fields of the selected statistic, one each quarter of an hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.108 — Test case 19.8

Identifier	TC19.8	
Applicable to	All entity types	
Verification of	SUT functionality	
Prerequirement 1	The SUT shall support standard history of 48 values.	
Test type	Positive	
System under test	Subsystem	
Test purpose	To verify the correct generation of 48 values standard history content for non-accumulative numerical properties, whether mandatory or extended.	
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. During the sample day, changes shall be forced in all numerical properties. At least ten (10) changes times shall be performed, being four (4) of them during the same hour. The previous and the current property values and the timestamp of each change shall be registered. 3. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 4. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 5. The standard history values should match with the ones registered. It should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.8"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those numerical type properties belonging to the entity type managed by the SUT>>

Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"48"
Request 1 parameter 6	Statistics (string)	<<All possible statistic values>>
Response 1 parameter 1	ActionID (string)	"Test_19.8"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 48 fields of the selected statistic, one per half an hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

Table D.109 — Test case 19.9

Identifier	TC19.9	
Applicable to	All entity types	
Verification of	SUT functionality	
Prerequirement 1	The SUT shall support standard history of 96 values.	
Test type	Positive	
System under test	Subsystem	
Test purpose	To verify the correct generation of 96 values standard history content for non-accumulative numerical properties, whether mandatory or extended.	
Test sequence	1. The test scope is the natural day (from 00:00h to 23:59h). 2. During the sample day, changes shall be forced in all numerical properties. At least ten (10) changes times shall be performed, being four (4) of them during the same hour. The previous and the current property values and the timestamp of each change shall be rgistered. 3. Once the sample day had been end, the tester is able to send a ReadStandardHist request (1). 4. The SUT returns a "0 – Action successfully executed" and the history values for the requested property. 5. The standard history values should match with the ones registered. It should be verified by comparison.	
Request 1 parameter 1	ActionID (string)	"Test_19.9"
Request 1 parameter 2	EntityID (string)	"E00X"
Request 1 parameter 3	PropertyName (string)	<<All those numerical type properties belonging to the entity type managed by the SUT>>
Request 1 parameter 4	Date (DateTime)	Natural day of the requested history "YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 5	NumHist (integer)	"96"
Request 1 parameter 6	Statistics (string)	<<All possible statistic values>>
Response 1 parameter 1	ActionID (string)	"Test_19.9"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3	HistStandardValues (string)	The string shall contain 96 fields of the selected statistic, one each quarter of an hour. Each field is compound by the value of the property and it's timestamp. The first field shall correspond to the 0 hour and the last to the 24 hour. The hours correspond to the natural day selected in Date parameter.

D.6.3.10 Report content validation

D.6.3.10.1 Abstract conformance test

Table D.110 — Test description report validation

Test purpose	To verify the content included on the report of a procedural element.
System under test	Subsystem.
Test sequence	A tester acting as coordination broker requests to the SUT the report of a procedural element. The values returned by the SUT in the report shall match with the ones expected, attending to the standard definition.
Request parameters	ActionID, EntityID, PropertyName, Date, NumHist, Statistics
Response parameters	ActionID, ResponseCode, HistStandardValues

D.6.3.10.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.111 — Test case 20.1

Identifier	TC19.1	
Applicable to	All entity types	
Verification of	SUT functionality	
Test type	Positive	
System under test	Subsystem	
Test purpose	To verify the correct generation of an procedural element report.	
Test sequence	1. The test scope is an procedural element execution. 2. The tester sends a CreateRecipe request (1) to the SUT. 3. The SUT returns a "0 – Action successfully executed". 4. During the procedural element execution shall be registered the start and end conditions and dates, its real duration, the water comsumption as well as any other data that should be included in the report according to the standard definition. The report shall include the expected events. 5. If possible, the pressure or the flow values should be modified during the execution to verify the correct registration of maximum, minimum and average statistic values. 6. After the end condition has been reached, the tester sends a ReadReport request (2) to the SUT. 7. The SUT returns a "0 – Action successfully executed" and the report. 8. The values included in the report should be similar to the registered at the step 4.	
Request 1 parameter 1	ActionID (string)	"Test_20.1"
Request 1 parameter 2	ProceduralID (string)	"P00X"
Request 1 parameter 3	EntityID (string)	"E00X"
Request 1 parameter 4	EntityType (string)	<<Any valid entity type>>
Request 1 parameter 5	RecipeType (string)	<<Any valid procedural element>>
Request 1 parameter 6	StartDate (DateTime)	"YYYYMMDDhhmmss±hhmm" in UTC ISO 8106.
Request 1 parameter 7	MaxDuration (integer)	"600"
Response 1 parameter 1	ActionID (string)	"Test_20.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Request 2 parameter 1	ActionID (string)	"Test_20.1"
Request 2 parameter 2	EntityID (string)	"E00X"

Request 2 parameter 3	ProceduralID (string)	"P00X"
Request 2 parameter 4	Instance (integer)	"0"
Response 2 parameter 1	ActionID (string)	"Test_20.1"
Response 2 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 2 parameter 3	Report (string)	"P00X,0, RecipeType,initial date (YYYYMMDDhhmmss), final date (YYYYMMDDhhmmss), end condition reached, duration, volume, <<irrelevant>>, max flow out, max flow out time stamp, min flow out, min flow out time stamp, average flow out, max pressure out, max pressure out time stamp, min pressure out, min pressure out time stamp, started operation eventID, completed operation eventID"

D.7 Tests over coordination brokers

D.7.1 Test application guide

The coordination broker functions as a router when the recipient of the requests is an irrigation entity controlled by a subsystem. In this case, the tests are limited to verifying the interfaces, using all the communication protocols that the SUT supports. When requests are intended for a virtual irrigation entity controlled by the coordination broker, the test battery defined for a subsystem shall be also applied.

D.7.2 Verification interfaces tests

D.7.2.1 Well parameterized requests

D.7.2.1.1 Abstract conformance test

Table D.112 — Test description for well-written request submission

Test purpose	To verify that the SUT correctly implements the management interface as server and the subsystem interface as client.
System under test	Coordination broker
Test sequence	A tester acting as a MIS application sends requests to the SUT. A second tester, acting as a subsystem receives the requests that the SUT should route. It should be verified that the requests send by the SUT to the second tester conform to the implementation of the communications protocol being tested. This test is performed for all methods that the SUT supports in the management interface.

D.7.2.1.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.113 — Test case 21.1

Identifier	TC21.1
Applicable to	All entity types
Verification of	Management and subsystem interface
Test type	Positive
System under test	Coordination broker
Prerequirement 1	The entity "E00X" shall exist in the SUT.
Prerequirement 2	The SUT has access to "E00X" using hte subsystem interface.

Prerequirement 3	The testers shall use one of the communication protocols supported by the SUT in each interface.
Test purpose	To verify the correct processing of well formulated requests in the management interface.
Test sequence	1. The communications protocol to be used at each SUT interface in the test case shall be registered. 2. The tester sends a request (1) to the SUT. 3. The SUT sends a request (2) to the second tester which acts as a subsystem. 4. The tester returns to the SUT a "0 - Action successfully executed" response and a controlled and correct content for the request. 5. The SUT returns a "0 - Action successfully executed" and the content of the response obtained from the second tester to the tester that made the original request. 6. The contents of the request (2) and the response returned by the SUT in step 5 are verified to confirm that both comply with the specifications of the standard for that method and the communications protocol being tested. This sequence should be repeated for the common methods defined for both interfaces supported by the SUT.

D.7.2.2 Bad parameterized requests

D.7.2.2.1 Abstract conformance test

Table D.114 — Test description for bad-written request submission

Test purpose	To verify that the SUT correctly implements the management interface as server and the subsystem interface as client.
System under test	Coordination broker
Test sequence	A tester acting as a MIS application sends bad requests to the SUT. A second tester, acting as a subsystem receives the requests that the SUT should route. It is verified that the SUT do not send bad requests to the second tester and that returns a expected error message to the first tester. This test is performed for all methods that the SUT supports in the management interface.

D.7.2.2.2 Conformance tests

The conformance tests are composed by one (1) negative test case.

Table D.115 — Test case 22.1

Identifier	T022.1
Applicable to	All entity types
Verification of	Management and subsystem interfaces
Test type	Negative
System under test	Coordination broker
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT in each interface.
Test purpose	To verify the filtering of bad parameterized requests in the management interface.

Test sequence	1. The communications protocol to be used at each SUT interface in the test case shall be registered. 2. The tester sends a bad parameterized request (1) to the SUT. 3. The SUT returns a "2-Lexical error" and does not send the request to the subsystem interface. This sequence should be repeated for the common methods defined for both interfaces supported by the SUT. Random values shall be entered in all parameters of all requests that support them.

D.7.2.3 Events tests

D.7.2.3.1 Abstract conformance test

Table D.116 — Test description for events subscription and unsubscription

Test purpose	To verify that the SUT executes the suscription and the unsuscription to an entity events.
System under test	Coordination broker
Test sequence	Acting the SUT as client in the subsystem interface, the subscription and unsubscription methods should be tested, verifying in a tester acting as a subsystem that the subscription and unsubscription requests are correctly received. A second tester is used to act as client of the event interface to generate events and send them to the SUT. The SUT should receive the events as server in the event interface and processes them correctly.

D.7.2.3.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.117 — Test Case 23.1

Identifier	TC23.1
Applicable to	All entity types
Verification of	Subsystem and event interfaces.
Test type	Positive
System under test	Coordination broker
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT in each interface.
Test purpose	To verify the ability of the SUT to subscribe, unsubscribe, receive, and process events.
Test sequence	1. The communications protocol to be used in the SUT subsystem and event interfaces is registered. 2. A SubcribeEvent request (1) is sent to the first tester via the SUT user interface. It is verified that the content of the request (1) respects the definition of the standard for the communication protocol used. The server path to send the events shall be registered. 3. The tester returns a "0 - Action successfully executed" response to the SUT. It is required to verify that the SUT correctly receives and interprets the response. 4. Once step 3 is verified, a NewEvent request (2) is generated in a second tester, with an event addressed to the path indicated in the request (1). The response returned by the SUT to the tester is verified, being the expected "0 - Action successfully executed". 5. The verification is done in the user interface of the SUT where the event has been correctly registered. 6. An UnSubcribeEvent request (3) is sent through the SUT user interface to the first tester. 7. The tester returns a "0 - Action successfully executed" response to the SUT. It is required to verify that the SUT correctly receives and interprets the response.

D.7.2.4 Access and permission tests

D.7.2.4.1 Abstract conformance test

Table D.118 — Test description for access and permission

Test purpose	To verify that the SUT correctly implements access mechanisms and permissions for clients accessing its management or event interface, as well as for client access using the subsystem interface.
System under test	Coordination broker
Test sequence	A tester acting as MIS application requests from the SUT the identifiers of all the irrigation entities it has loaded and to which this tester has access permission.

D.7.2.4.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.119 — Test Case 24.1

Identifier	TC24.1	
Applicable to	All entity types	
Verification of	Management interface	
Test type	Positive	
System under test	Coordination broker	
Prerequirement 1	“E00X”, “E00Y”, “E00Z”, “E00V” y “E00W” shall exist in the SUT.	
Prerequirement 2	The management tester shall have access to the entities "E00X" and "E00Y".	
Prerequirement 3	The management tester only have permission to execute the Read and Write methods.	
Prerequirement 4	The SUT have all the data of access to the testers of the subsystem interface.	
Test purpose	To verify that the SUT manages permits and accesses to irrigation entities correctly.	
Test sequence	1. The tester sends a Read request (1) to the SUT intended for "E00X". 2. The SUT returns a "0 - Action successfully executed" and the value of the property registered in the destination subsystem tester. 3. The tester sends a Read request (2) to the SUT intended for "E00W". 4. The SUT returns a "1 - Execution error". 5. The tester sends a Write request (3) to the SUT for "E00Y". 6. The SUT returns a "0 - Action successfully executed" and updates the property value in the destination subsystem tester. 7. The tester sends a Write request (4) to the SUT for "E00V". 8. The SUT returns a "1 - Execution error". 9. The tester sends a request (5) for a method intended for “E00X” for which it is not authorized. 10. The SUT returns a "1 - Execution error".	
Request 1 parameter 1	ActionID (string)	“Test_24.1”
Request 1 parameter 2	EntityID (string)	“E00X”
Request 1 parameter 3	PropertyName (string)	<<Any PropertyName>>
Response 1 parameter 1	ActionID (string)	“Test_24.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 1 parameter 3	Value	<<Value of the property>>
Request 2 parameter 1	ActionID (string)	“Test_24.1”

Request 2 parameter 2	EntityID (string)	"E00W"
Request 2 parameter 3	PropertyName (string)	<<Any PropertyName>>
Response 2 parameter 1	ActionID (string)	"Test_24.1"
Response 2 parameter 2	ResponseCode (string)	"1 – Execution error"
Request 3 parameter 1	ActionID (string)	"Test_24.1"
Request 3 parameter 2	EntityID (string)	"E00Y"
Request 3 parameter 3	PropertyName (string)	<<Writable PropertyName>>
Request 3 parameter 4	Value (string)	<<Valid value>>
Response 3 parameter 2	ActionID (string)	"Test_24.1"
Response 3 parameter 3	ResponseCode (string)	"0 – Action successfully executed"
Request 4 parameter 1	ActionID (string)	"Test_24.1"
Request 4 parameter 2	EntityID (string)	"E00V"
Request 4 parameter 3	PropertyName (string)	<<Writable PropertyName>>
Response 4 parameter 1	Value (string)	<<Valid value>>
Response 4 parameter 1	ActionID (string)	"Test_24.1"
Response 4 parameter 2	ResponseCode (string)	"1 – Execution error"
Parameters of request 5	Parameters required for the request 5, intended for "E00X".	
Response 5 parameter 1	ActionID (string)	"Test_24.1"
Response 5 parameter 2	ResponseCode (string)	"1 – Execution error"

D.7.3 Specific test for management interface methods

D.7.3.1 Test of "Reading the identifiers of all irrigation entities known by the SUT (ReadEntityIDs method)"

D.7.3.1.1 Abstract conformance test

Table D.120 — Test description for ReadEntityIDs method

Test purpose	To verify that the SUT correctly implements access mechanisms and permissions for clients accessing its management or event interface, as well as for its client access using the subsystem interface.
System under test	Coordination broker
Test sequence	A tester acting as MIS application sends requests to the SUT. Access to some irrigation methods and entities has been assigned to the tester in the SUT user interface. The SUT also have all the data needed to access the irrigation entities that are interoperated through the subsystem interface.

D.7.3.1.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.121 — Test Case 25.1

Identifier	TC25.1
Applicable to	All entity types
Verification of	Management interface
Test type	Positive

System under test	Coordination broker	
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT.	
Prerequirement 2	“E00X”, “E00Y”, “E00Z”, “E00V” y “E00W” shall exist in the SUT.	
Prerequirement 3	The tester only have access to “E00X”, “E00Y” and “E00Z”.	
Test purpose	To verify that the SUT returns the identifiers of all irrigation entities to which the tester has authorized access.	
Test sequence	1. The communications protocol to be used in the SUT management interface is registered. 2. The tester sends the SUT a ReadEntityIDs request (1) with the necessary data for its authentication, according to the security method defined for the interface. 3. The SUT returns "0 - Action successfully executed" and the EntityIDs of the entities that have been granted access to the tester.	
Request 1 parameter 1	ActionID (string)	“Test_24.1”
Response 1 parameter 1	ActionID (string)	“Test_24.1”
Response 1 parameter 2	ResponseCode (string)	“0 – Action successfully executed”
Response 1 parameter 3	EntityIDs	“E00X”, “E00Y”, “E00Z”

D.7.3.2 Test of “Reading the static data of an irrigation entity known by the SUT (ReadEntityData method)”

D.7.3.2.1 Abstract conformance test

Table D.122 — Test description for ReadEntityData method

Test purpose	To verify that the SUT returns the static data of an irrigation entity that has been previously loaded with all its characteristic data.
System under test	Coordination broker
Test sequence	A tester acting as MIS application requests from the SUT the static data of an irrigation entity that it has loaded and to which this tester has access permission. The response verifies that the SUT correctly returns the data of the requested entity, as well as respecting the implementation of the management interface.

D.7.3.2.2 Conformance tests

A set of tests made up of the following are performed:

1) One (1) positive test; and

2) One (1) positive test.

Table D.123 — Test Case 26.1

Identifier	TC26.1
Applicable to	All entity types
Verification of	Management interface
Test type	Positive
System under test	Coordination broker
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT.
Prerequirement 2	“E00X”, “E00Y”, “E00Z”, “E00V” y “E00W” shall exist in the SUT.
Prerequirement 3	The tester only have access to “E00X”, “E00Y” and “E00Z”.
Prerequirement 4	The Test Case 24.1 has been successfully tested previously.

Test purpose	To verify that the SUT returns the static data of the irrigation entities to which the tester has authorized access.	
Test sequence	1. The entities registered in test case 24.1 shall be used. 2. The communications protocol to be used in the SUT management interface is registered. 3. The tester sends to the SUT a ReadEntityData request (1) with the necessary data for its authentication. 4. The SUT returns "0 - Action successfully executed" and the static data of the requested entity, including null values for all those parameters that have not been entered. The data returned by the SUT in the response corresponds to the data registered in its user interface.	
Request 1 parameter 1	ActionID (string)	"Test_25.1"
Request 1 parameter 2	EntityID (string)	"E00X"
Response 1 parameter 1	ActionID (string)	"Test_25.1"
Response 1 parameter 2	ResponseCode (string)	"0 – Action successfully executed"
Response 1 parameter 3 to n	Static data	The SUT returns all static data from the irrigation entity. Each of the following shall be parameters of the Response 1: EntityLevel, EntityType, ControlLevel, SpatialData, PrecursorEntities, SuccessorEntities, ChildEntities, ParentEntity, InternalFlowOut, PressureOut, DesignPressure, DesignFlowRate, Caliber, Temperature, RelativeHumidity, SolarRadiation, WindParameters, Rainfall, CumulativeVolumeIn, InternalFlowIn, PressureIn, SoilWaterContent, SoilWaterPotential, SoilTemperature, SoilConductivity, Behavior, DesignSimultaneity, DesignPower, DesignCapacity, DesignAutonomy, ActivityStatus, SystemStatus, OpeningDegree, Area, MeterReplacementEvent, PowerOnTimeCounterStart, EnergyMeterCounterStart

Table D.124 — Test Case 26.2

Identifier	TC26.2	
Applicable to	All entity types	
Verification of	Management interface	
Test type	Negative	
System under test	Coordination broker	
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT.	
Prerequirement 2	"E00X", "E00Y", "E00Z", "E00V" y "E00W" shall exist in the SUT.	
Prerequirement 3	The tester only have access to "E00X", "E00Y" and "E00Z".	
Prerequirement 4	The Test Case 24.1 has been successfully tested previously.	
Test purpose	To verify that the SUT returns an error message when the tester requests the static data of a non-existent or a non-access authorized irrigation entity.	
Test sequence	1. The entities registered in test case 24.1 shall be used. 2. The communications protocol to be used in the SUT management interface is registered. 3. The tester sends to the SUT a ReadEntityData request (1) with the necessary data for its authentication. 4. The SUT returns "1 – Execution error".	
Request 1 parameter 1	ActionID (string)	"Test_25.2"
Request 1 parameter 2	EntityID (string)	"E00X"
Response 1 parameter 1	ActionID (string)	"Test_25.2"
Response 1 parameter 2	ResponseCode (string)	"1 – Execution error"

D.7.4 Routing test

D.7.4.1.1 Abstract conformance test

Table D.125 — Test description for routing

Test purpose	To verify that the SUT correctly routes the requests to the subsystem controlling the destination irrigation entity.
System under test	Coordination broker
Test sequence	A tester acting as a MIS application send a request to the SUT for an irrigation entity it has registered. The SUT should send the requests to the subsystem controlling the destination irrigation entity.

D.7.4.1.2 Conformance tests

The conformance tests are composed by one (1) positive test case.

Table D.126 — Test Case 27.1

Identifier	TC27.1
Applicable to	All entity types
Verification of	Management interface
Test type	Negative
System under test	Coordination broker
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT.
Prerequirement 2	“E00X”, “E00Y”, “E00Z”, “E00V” y “E00W” shall exist in the SUT.
Prerequirement 3	"E00X" and "E00Y" are associated with testers that implement different communication protocols.
Test purpose	To verify that the SUT correctly routes requests to the subsystems controlling registered irrigation entities.
Test sequence	1. The communications protocol to be used in the SUT management interface is registered. 2. The tester sends two (2) Read requests (1) and (2) to the SUT. 3. The SUT routes the requests to the testers prepared to act as subsystems through the subsystem interface. For each request, the SUT uses the communications protocol assigned to each tester when registering the irrigation entities to which the requests are addressed. The requests send by the SUT to the testers conform to the different communication protocol implementation included in this document. 4. Both testers return to the SUT a "0 - Action successfully executed" and a correct value according to the requests (1) and (2). 5. The SUT returns the obtained responses to the first tester. It is verified that the responses are the expected. This sequence shall be performed with all methods common to both interfaces and implemented by the SUT.

D.7.5 Subsystem performing tests

D.7.5.1.1 Abstract conformance test

Table D.127 — Test description for subsystem performing

Test purpose	To verify that the SUT behaves as a subsystem when controlling virtual irrigation entities.
System under test	Coordination broker
Test sequence	A tester acting as a MIS application sends requests to the SUT referring to an irrigation

	entity that it has registered and with its control assigned. All the applicable tests defined for the subsystems shall be permormed.

D.7.5.1.2 Conformance tests

The conformance tests to be performed for an irrigation entity controlled by the SUT are the included in the following clauses of this document:

1) D.6.2.2 Test “Reading a property of an entity (Read method)”;
2) D.6.2.3 Test “Writing a property of an entity (Write method)”;
3) D.6.2.4 Test “Creating a procedural element (CreateRecipe method)”;
4) D6.2.5 Test “Read the procedural elements loaded in the SUT (ReadProceduralIDs method)”;
5) D6.2.6 Test “Stop a procedural element (StopRecipe method)”;
6) D.6.2.7 Test “Reading a procedural element report (ReadReport method)”;
7) D.6.2.8 Test “Reading the history of a property (ReadStandardHist method)”;
8) D.7.2.9 Test “Subscription to an entity events (SubscribeEvent method)”;
9) D.7.2.10 Test “Unsubscription to an entity events (UnSubscribeEvent method)”;
10) D.7.2.11 Test “Generation of an entity event (NewEvent method)”.

D.8 Tests over management information systems

D.8.1 Test application guide

An MIS application can implement a variable number of the methods defined in this document, according to the management needs it intends to cover. The battery of tests in any MIS application is limited to verify compliance with the management interface as well as the correct interpretation of the responses received to its requests.

D.8.2 Management interface verification

D.8.2.1 Abstract conformance test

Table D.128 — Test description for management interface verification

Test purpose	To verify that the SUT respects the management interface and correctly interprets the exchanged data.
System under test	MIS application
Test sequence	The MIS application, acting as SUT sends requests to a tester who acts as coordination broker. The requests received by the tester allows to verify if the SUT respects the management interface for the methods it implements. The answers provided by the tester are correctly interpreted by the SUT.

D.8.2.2 Conformance tests

The conformance tests are composed by three (3) positive test cases.

Table D.129 — Test Case 28.1

Identifier	TC28.1
Applicable to	All entity types
Verification of	Management interface
Test type	Positive
System under test	MIS application
Prerequirement 1	The entity “E00X” shall exist in the SUT.
Prerequirement 2	The SUT has access to “E00X” over the subsystem interface.
Prerequirement 3	The testers shall use one of the communication protocols supported by the SUT.
Prerequirement 4	The Test Case 24.1 has been successfully tested previously.
Test purpose	To verify the ability to process well-parametered requests in the management interface and send them correctly.
Test sequence	1. The communications protocol to be used in the SUT management interface in the test case is registered. 2. A request (1) is sent to the tester from the SUT user interface. 3. The content of the request (1) received by the tester is verified. 4. The tester returns a "0 - Action successfully executed" and the content of a well parameterized response. 5. The SUT shall register the correct reception of the response and the content shall be updated, verifying in the user interface the correct interpretation of the received response. This sequence shall be repeated for all methods that the SUT supports in the management interface.

Table D.130 — Test Case 28.2

Identifier	TC28.2
Applicable to	All entity types
Verification of	Management interface
Test type	Positive
System under test	MIS application
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT.
Test purpose	To verify the ability of the SUT to filter error responses obtained in the management interface.
Test sequence	1. The communications protocol to be used in the SUT management interface in the test case is registered. 2. A request (1) is sent to the tester from the SUT user interface. 3. The content of the request (1) received by the tester is verified. 4. The tester returns a "1- Execution error". 5. The SUT shall record the response and the error reported. This sequence shall be repeated for all error types that are defined in the document for the management interface.

Table D.131 — Test Case 28.3

Identifier	TC28.3
Applicable to	All entity types
Verification of	Management interface
Test type	Negative

System under test	MIS application
Prerequirement 1	The testers shall use one of the communication protocols supported by the SUT.
Test purpose	To verify the ability of the SUT to filter bad parameterized responses obtained in the management interface.
Test sequence	1. The communications protocol to be used in the SUT management interface in the test case is registered. 2. A request (1) is sent to the tester from the SUT user interface. 3. The content of the request (1) received by the tester is verified. 4. The tester returns a "0 - Action successfully executed" and a bad parameterized content. 5. The SUT shall record the response and to generate an error in its user interface.

D.9 Tests report

D.9.1 General

The tests report should include the following information.

D.9.2 SUT data

The first part of the report includes the characteristic data of the SUT, based on its Implementation Conformance Statement (ICS):

1) name of the owner of the product;

2) trade name;

3) stated irrigation entities controlled by the SUT;

4) expected functionalities;

5) extended functionalities supported;

6) the set of tests to be performed.

D.9.3 Test data

The test data that should be covered in the report including:

1) a reference to this document (i.e. ISO 21622-3) and the version implemented by the SUT;

2) series number of all the hardware elements tested as well as all the software version, including OS;

3) description of the type test module used (hydraulic or simulated), including the identification (model and specifications) of all the elements included in it, regardless if they are hydraulic or electronic;

4) if proceed, manometric pressure recorded during the tests, pilot setting pressure and volume of outflow.

D.9.4 Results

The results of the tests that shall be covered in the report include the following:

1. identification of all the test cases performed;

2. expected result and the criteria to accept every stage of each test;
3. the result obtained in each test case, indicating the non-accomplishments detected if exists;
4. a brief of passed, non-passed and non-performed test cases.

Annex E
(informative)

Coordination broker software requirement specifications

E.1 Overview

E.1.1 General

This annex is a functionalities requirement specification to facilitate the development of coordination brokers.

The objective is to establish the functional and non-functional requirements of a coordination broker in keeping with this document and its implementation annexes.

This annex specifies:

— minimal functions to guarantee the basic functional operation of a coordination broker;

— desirable functions to adapt to the needs of the end user.

The purpose of this annex is to orient the development of coordination brokers adapted to this document, describing all factors affecting the product and its requirements.

It addresses technical specialists in software application and communications system development.

Coordination brokers suited to the needs of each user may be developed in accordance with the specifications of this annex. In any case, all coordination brokers should support each of the irrigation entities defined in the physical model.

E.1.2 Definitions

E.1.3 System description

The coordination broker is a software element that provides interoperability between subsystems and management systems, performing the following basic functions:

— mapping irrigation entities and their associations with the subsystems controlling them;

— creation of an abstraction layer: management applications do not to require information about the subsystems on which they operate;

— coordination between subsystems, providing these with the information they require, regardless of origin.

The additional functions that a coordination broker can perform include:

— compilation of reports, records and events from the subsystems controlling the mapped irrigation entities;

— data consolidation and storage, acting as a data bank those compatible systems can connect to;

— control of virtual irrigation entities.

The coordination broker should provide three communication interfaces to stablish interoperability:

— management interface;

— subsystem interface; and

— event interface.

E.1.4 Technological environment

The coordination broker described is an application that attends to requests from the management interface, executing the functions associated to it and sending the required requests to the subsystems it coordinates. Hence, the coordination broker is:

— a server that attends to requests from the management interface;

— a server that receives events from the event interface; and

— a client that sends requests through the subsystem interface.

E.1.5 Standard and regulatory specifications

Development of the coordination broker should comply with the specifications defined in this document, as well as its implementation annexes.

This annex should observe IEEE 830-1998 IEEE Recommended Practice for Software Requirements Specifications, but only to define functionalities and without stablish any other particular software requirement or specification.

E.1.6 Architecture

The coordination broker should observe two different architectural patterns, depending on the view of system operation:

— For the communications view is used the client-server architecture as a distributed application model, where tasks are distributed among the resource or service providers (servers) and the resource users (clients). In this sense, the coordination broker should implement the three standard interfaces, acting with two roles:

  — client towards the subsystem level, using the data provided by them;

  — server towards the management level, providing data to any MIS;

  — server towards the subsystem level, designed to the subsystems event collection.

— For the functional view, the architectural pattern to use is the “model-view-controller”, separating data and business logic from the user interface.

E.1.7 User interface

The coordination broker also provide a user interface for enabling certain cases of use described in E.9.

E.1.8 Communication interfaces

The coordination broker should implement three interfaces:

— management interface to receive requests.

— subsystem interface to make requests.

— event interface to receive events.

E.1.9 Restrictions

The coordination broker should support the irrigation entities described in this document.

E.1.10 Memory requirements

No memory use restrictions exist for the coordination broker.

E.2 Data model

E.2.1 General

The coordination broker should assure data persistence for the static data for all the irrigation entities entered for administration, whatever their level in the physical model.

Additionally, the coordination broker should be able to store:

— reports from all procedural elements, in keeping with their standard definitions;

— standard history of the mandatory and extended properties reported from all the irrigation entities registered in the coordination broker, regardless of its level in the physical model;

— all procedural elements, fundamentally operations.

E.2.2 Data model: Report

Table E.1 describes the attributes and data types of reports.

Table E.1 — Report data model

Attribute	Type
Report	String
Status	Status

E.2.3 Data model: Standard history

Table E.2 describes the attributes and types of standard history.

Table E.2 — Standard history data model

Attribute	Type
EntityID	String
PropertyName	String
HistStandardValues	String

E.2.4 Data model: Procedural element

E.2.4.1 General

Table E.3 describes the attributes and types of all procedural elements (procedures, unit procedures and operations).

Table E.3 —Procedural element data model

Attribute	Type
ProceduralID	String
EntityID	String
RecipeType	RecipeType
Status	Status

E.2.4.2 Enumerated Status

Table E.4 describes the possible statuses of a procedural element.

Table E.4 —Status enumeration

List of values
Idle
Running
Stopped
Completed
Unknown

The procedural elements executed by virtual entities shall not show the Status value Unknown.

E.2.5 Data model: Irrigation entity

E.2.5.1 General

Table E.5 describes the attributes of an irrigation entity. This set of attributes may be extended if new types of irrigation entities containing other characteristic data are incorporated into the physical model.

Table E.5 —Irrigation entity data model

Field	Type	Applies to
Identification		
EntityID	string	All
EntityLevel	EntityLevel	All
EntityType	EntityType	All
ControlLevel	ControlLevel	All
Location		
SpatialData	string	All
PrecursorEntities	string	All
SuccessorEntities	string	All
ChildEntities	string	All
ParentEntity	string	All
Communication		
Path	string	All
CommProtocol	CommProtocol	All
Design		
InternalFlowOut	boolean	HYS, VIH, MIH, NBU, INT, SSS and NCP
PressureOut	boolean	HYS, VIH, MIH, NBU, INT, SSS and NCP
DesignPressure	integer	HYS, VIH, MIH, IHG, NBU, INT, PMS, PML and SSB
DesignFlowRate	decimal	HYS, VIH, MIH, IHG, NBU, INT, PMS, PML and SSB
Caliber	integer	HYS, VIH, MIH, NCP
Temperature	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
RelativeHumidity	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SolarRadiation	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
WindParameters	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
Rainfall	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
CumulativeVolumeIn	boolean	IHG, NBU, INT, PMS and NCP
InternalFlowIn	boolean	IHG, NBU, INT, PMS and NCP
PressureIn	boolean	IHG, NBU, INT, PMS and NCP
SoilWaterContent	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SoilWaterPotential	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SoilTemperature	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
SoilConductivity	boolean	HYS, VIH, MIH, IHG, PMS, RSV, SSS and NCP
Behavior	Behavior	IHG
DesignSimultaneity	integer	NBU, INT and SSS
DesignPower	integer	PMS and PML
DesignCapacity	integer	RSV

Field	Type	Applies to
DesignAutonomy	integer	RSV
ActivityStatus	boolean	NCP, INT and NBU
SystemStatus	boolean	NCP, INT and NBU
Mode	boolean	INT y NBU
OpeningDegree	boolean	NCP, INT and NBU
Area	integer	SSS and SSB
Historical		
MeterReplacementEvent	string	HYS, VIH, MIH, IHG and NCP
PowerOnTimeCounterStart	string	PMS and PML
EnergyMeterCounterStart	string	PMS

E.2.5.2 Enumerated EntityLevel

Enumerated EntityLevel may have the values in Table E.6.

Table E.6 —EntityLevel enumeration

List of values
Empty
Area
ProcessCell
Unit
EquipmentModule

E.2.5.3 Enumerated EntityType

Enumerated EntityType may have the values described in Table E.7. This list can be extended whenever a new type of irrigation entity is introduced into the physical model.

Table E.7 —EntityType enumeration

List of values	Name
HYS	Simple irrigation hydrant
IRA	Irrigation sector
INT	Irrigation network
PMS	Pumping station
PML	Pumping line
RSV	Reservoir
SSS	Solid set irrigation system
SSB	Solid set irrigation block
NCP	Network control point
VIH	Virtual irrigation hydrant
MIH	Manual irrigation hydrant

List of values	Name
HYG	Irrigation hydrants group
NBU	Network branch
Empty	No value is declared

E.2.5.4 Enumerated CommProtocol

Enumerated CommProtocol may have the values described in Table E.8. This list can be extended whenever a new communications protocol is introduced and defined with its respective implementation annex.

Table E.8 —CommProtocol enumeration

List of values
SOAP12
REST

E.2.5.5 Enumerated RecipeType

Enumerated RecipeType may have the values described in Table E.9. This list can be extended whenever a new type of irrigation entity is introduced into the physical model.

Table E.9 — RecipeType enumeration

List of values
IrrigationRecipe1
IrrigationRecipe2
IrrigationRecipe3
IrrigationRecipe4
IrrigationGroupRecipe1
NetworkBranchRecipe1
PumpingRecipe1
PumpingRecipe2
PumpingRecipe3
PumpingRecipe4
PumpingRecipe5
ControlPointRecipe1
IrrigationNetworkUnitProcedure1
SolidSetUnitProcedure1
PumpingUnitProcedure1
ReservoirUnitProcedure1

E.2.5.6 Enumerated ControlLevel

Enumerated ControlLevel may have the values described in Table E.10.

Table E.10 —ControlLevel enumeration

List of values
Empty
HigherControlLevel
LowerControlLevel

E.2.5.7 Enumerated PropertyName

Enumerated PropertyName may have the values described in Table E.11. This list can be extended whenever a new type of irrigation entity is introduced into the physical model.

Table E.11 —PropertyName enumeration

List of values
ActivityStatus
SystemStatus
Mode
CumulativeVolumeOut
InternalFlowOut
PressureOut
OpeningDegree
Temperature
RelativeHumidity
SolarRadiation
WindParameters
Rainfall
CumulativeVolumeIn
InternalFlowIn
PressureIn
InstantPower
EnergyConsumption
EntityPerformance
PowerOnTime
SoilWaterContent
SoilWaterPotential
SoilTemperature
SoilConductivity
Capacity

E.2.5.8 Enumerated Behavior

Enumerated Behavior may have the values described in Table E.12.

Table E.12 —Behavior enumeration

List of values	Description
Empty	No behavior is declared.
Behavior1	The irrigation entity presents simultaneity restrictions between the entities it contains.
Behavior2	The irrigation entity simultaneously operates the entities it contains.

E.3 Minimal functions — Coordination broker

E.3.1 Function 1: Irrigation entity identification

The applications connected by the management interface should access an irrigation entity using its EntityID. The coordination broker should transact connections linking it to the access Path of its controlling subsystem.

E.3.2 Function 2: Irrigation entity administration

The coordination broker should manage the irrigation entities to manage. The coordination broker should enable the creation, deletion and modification of irrigation entities.

The irrigation entities in the coordination broker domain are identified by an unique ID called EntityID, which serves as the specific identifier for the irrigation entity. This identifier should be unique among the irrigation entities to manage and control.

When creating a new irrigation entity or modifying one, the coordination broker should provide an environment for its configuration. Configuration requires the entry of all the data about the irrigation entity, including statement of its held extended properties.

During the creation or modification process, the coordination broker should:

— verify that the data entered comply with the format defined in this document;

— guarantee that the enumerated data only have the values described in the data model;

— check that no duplicate irrigation entity identifiers (EntityIDs) exist;

— check for compatibility errors in the EntityLevel and EntityType fields, which should respect the physical model established in this document;

— require the ParentEntity field for all irrigation entity not occupying the ProcessCell level at the pyshical model;

— modify the ChildEntities field of the entity exercising as ParentEntity for the irrigation entity being created. In both cases, the ParentEntity field shall be unique and shall contain a valid EntityID. The ChildEntities field shall be a set of valid EntityID identifiers;

— require the PrecusorEntities field for all irrigation entity not occupying ProcessCell level. Establish an override mechanism for this requirement for each other irrigation entity without any precursor entity, attending to the exceptions defined in this document;

— modify the SuccessorEntities field of the entity exercising as PrecursorEntities for the irrigation entity being created. The PrecursorEntities and the SuccessorEntities fields shall be a set of valid EntityID identifiers;

— verify, if possible, the connection with the subsystem controlling the irrigation entity;

— identify the communications protocol to use in communicating with the irrigation entity by selecting it from a list of possible values that may be extended. Should the irrigation entity be a virtual entity controlled by the coordination broker, this identification may be omitted;

— identify whether or not the irrigation entity being created is a virtual entity controlled by the coordination broker. If so, E.5.6 is also applicable.

When an irrigation entity is deleted, the coordination broker should delete it from the list of ChildEntities associated to the ParentEntity in question.

The coordination broker should check and validate the hydraulic connections established by comparing the values entered in the PrecursorEntities and SuccessorEntities fields (materials and diameters).

E.3.3 Function 3: Coordination broker authentication

The purpose of this function is to guarantee coordination broker authenticity. To achieve this, the coordination broker publishes a digital certificate guaranteeing its authenticity.

E.3.4 Function 4: Management of coordination broker access over the management interface

Any MIS wishing to access the coordination broker through the management interface authenticate itself. The authentication methods to be used for interface access are those described in the implementation annexes for the management interface.

Through its user interface, the coordination broker should enable the MIS able to access over the management interface, allowing:

— the creation of new permissions for a new MIS;

— the deletion of permissions for a MIS; and

— the modification of permissions for a MIS.

E.3.5 Function 5: Subsystem access management

The subsystems implements one of the authentication methods described in the implementation annexes for the subsystem interface. The coordination broker, for its part, implements all the possible authentication methods described to guarantee its connectivity to the subsystems.

The subsystem access management takes place from the user interface, from which the following may be performed:

— creating access permission to a subsystem;

— deleting access permission to a subsystem; and

— modifying access permission to a subsystem.

The coordination broker should store the permissions in a secure way.

E.3.6 Function 6: Subsystem entity access levels

The purpose of this function is to manage the access levels to a subsystem irrigation entity from a MIS. A MIS should not be able to execute just any method on the subsystems and over the coordination broker.

There are two access levels to an irrigation entity from any MIS client (pre-authorised according to E.3.5):

— Access to irrigation entities; and

— Invoking the methods of the management interface.

The coordination broker should implement an access management system where the following are defined for each management interface client:

— to what irrigation entities it has access; and

— what methods it can invoke in the subsystem interface.

Through its user interface, the coordination broker should enable the administration of:

— Management interface clients authorizations to the irrigation entities; and

— the authorized methods that each client may execute.

E.3.7 Function 7: Requests propagation to subsystem interface over the management interface

The coordination broker receives requests from the management interface intended to the irrigation entities controlled by the subsystems. When the data has not been requested previously, the coordination broker should propagate the request over the subsystem interface to obtain it.

Only those requests received that are common to both interfaces should be propagated from the management level to the subsystem level. The coordination broker reviews requests and check whether the data requested are already available.

The coordination broker may check the requests received in accordance with E.5.2. Should a subsystem return over the subsystem interface an error message, these should be propagated over the management interface.

Table E.13 describes the scope of propagation for each of the methods established by this document.

Table E.13 — Propagation of methods

Name	Propagation from Management to Subsystems
Write	Yes
Read	Yes
ReadStandardHist	Yes, provided that the data requested are verified as not previously existing in the coordination broker.
CreateRecipe	Yes
StopRecipe	Yes

Name	Propagation from Management to Subsystems
ReadProceduralIDs	Yes
ReadReport	Yes, provided that the data requested are verified as not previously existing in the coordination broker.

E.3.8 Function 8: Connection with subsystems

The coordination broker should act as a client and be capable of connecting to subsystems, respecting the subsystem interface and the methods defined in its implementation annexes.

The coordination broker should support all the protocols defined in the implementation annexes (Annex B for SOAP 1.2 and Annex G for REST). The coordination broker should access the static data of each irrigation entity to identify the communication protocol implemented for each one.

E.3.9 Function 9: Attending to system requests through the management interface

The coordination broker should act as a server and enable connections from MIS applications through requests that respect the management interface and the methods defined in its implementation annexes.

The coordination broker should support all the protocols defined in the implementation annexes (Annex A for SOAP 1.2 and Annex F for REST). Should the coordination broker receive corrupted data from the subsystems, it should reject this and return a communication error (Value1="4 – Communication error") to the client launching the request.

E.3.10 Function 10: Management of subscription of irrigation entity events

The coordination broker should be able to subscribe or unsuscribe to the events that occur in the irrigation entities. The subscription function should be executed related to the irrigation entities. Once the coordination broker subscribes to an entity, it should be subscribed to all its events.

The subscription/unsuscription process should be done through "SubscribeEvent" or "UnsubscribeEvent" sent from the coordination broker to the subsystem controlling the irrigation entity. The methods are defined in the implementation annexes related to the subsystem interface (Annex B for SOAP 1.2 and Annex G for REST). To use this method, the user should select the entities to subscribe over the coordination broker user interface. There is no subscription method at the management interface.

Once the coordination broker has subscribed to the events, the coordination broker is prepared to receive the events occurring in all the irrigation entities to which it has been previously subscribed. Events are received through the "NewEvent" method, defined in the implementation annex of the event interface.

E.3.11 Function 11: Request verification

The coordination broker should verify the integrity of all the requests received over the management interface to prevent the dispatch of poorly formulated requests through the subsystem interface. The coordination broker should check the structure of the received requests and does not propagate:

a) Poorly formulated requests: those differing from the requests defined in the implementation annexes or containing absent fields for which the default values have not been set in the document.

b) Requests addressed to non-existent irrigation entities: those requests made regarding irrigation entities not listed in the Broker (non-existent EntityID).

c) Requests addressed non-existent element procedures (unlisted ProceduralID).

d) Requests incompatible with the standard definitions: requests on properties not admitted or that present certain restrictions, such as WRITE method.

The coordination broker should check and validate the full content of requests for the methods defined, except for CREATERECIPE. Only in this case, the content of E.2.4 Data Model: Procedural element should be verified.

Any request not processed by the coordination broker returns a Value1=”2 - Lexical error” message. The coordination broker may expand the information on the error.

E.4 Desirable functions

E.4.1 Function 12: Standard history administration

The coordination broker should automatically send requests to obtain standard histories for the following purposes:

a) the coordination broker should act as a data bank for standard history on all the properties of all the irrigation entities it manages;

b) to reduce requests over the subsystem interface.

The coordination broker should obtain the standard history on all the mandatory and extended properties of the irrigation entities it manages, using the subsystem interface. This process should be done in a programmed manner on a daily basis, whereby the execution schedule may be configured over the coordination broker user interface.

The coordination broker may be asked for the standardized record of any property held by the irrigation entities it manages through the management interface.

The coordination broker should store the information obtained on a persistent data system, according to the model described in E.3.3. Standardized record administration should enable the configuration of scheduled collection through the “ReadStandardHist” method. The method fields that may be configured for automated requests are those reflected in Table E.14.

Table E.14 — Standardized record fields for configuration

NumHist	Integer	Number of samples to be contained in the daily standardized record requested.	The number of samples a day shall have one of the following values: 24, 48 or 96.
Statistics	String	Name of the statistical value for which the standardized record is to be generated.	Type of statistical value for the calculation of samples. This contains one of the following listed values: — last — average — minimum — maximum

If a request to obtain the standardized record of a property (PropertyName) is launched from the management interface, the coordination broker checks if the date entered in the request is already available by having collected it on schedule. If the coordination broker have previously requested that

standard history, it returns the data contained in its database and not launch the request over the subsystem interface. Should the coordination broker not have that standard history, it launches the request over the subsystem interface to obtain it.

The coordination broker should not save the data requested through discrete requests from the management interface in its data bank.

E.4.2 Function 13: Procedural elements management

The procedural elements are created with the “CreateRecipe” method from the management interface. In this function should be performed a set basic tests to ensure that the subsystem interface is not overloaded are conducted in this function. Following are checked:

1) That ProceduralID does not yet exist. Otherwise, a Value1=”2 - Lexical error” message is returned.

2) That the EntityType entered is compatible with the RecipeType. Otherwise, a Value1=”2 - Lexical error” message is returned.

The coordination broker should save the data regarding the creation of a procedural element in the data bank, in keeping with the model described in E.3.4. The Status field should start at “Idle” value.

The status field should be updated when a “ReadReport” action is executed, the subsystem returns a value for status, as described in E.3.4.2.

When the coordination broker receives a request to create a procedural element over the management interface, it sends this to the subsystem, which should execute it over the subsystem interface.

E.4.3 Function 14: Report administration

The coordination broker should store full reports on any procedural element. To obtain this, it generates a programmed request using the “ReadReport” method directed to the irrigation entity where the procedural element is going to be executed. This method should be executed when the coordination broker has identified the procedural element status as Stopped or Completed.

In the event of any report request, the coordination broker shall check if this has been previously requested. In affirmative case, the coordination broker should return the data contained in its database and not launch the request over the subsystem interface. In negative case, the coordination broker should launch the request over the subsystem interface.

E.4.4 Function 15: Data bank

The coordination broker should store its configuration and should maintain the following data in a single data bank:

— irrigation entities properly registered;

— records with errors, warnings and informative events related to the operation of the coordination broker.

The following should be included in the same database for each irrigation entity registered in the coordination broker:

— complete report of all procedural elements performed and to perform;

— daily standard history on mandatory and extended properties;

— events that have occurred in the irrigation entity and their procedural elements;

— procedural elements.

The coordination broker should avail of a mechanism to back up the data bank.

E.4.5 Function 16: Management of virtual irrigation entities

The coordination broker should manage virtual irrigation entities composed by multiple entities controlled by one or several subsystems. A virtual irrigation entity behaves like a non-virtual entity, being subject to requests as defined in E.4. In most cases, the coordination broker should be in charge of its management.

The following should be taken into account in managing virtual entities:

— the coordination broker should be able to assume the logic of any virtual irrigation entity defined in this document. A virtual entity should be composed of a set of irrigation entities that have been previously created in the coordination broker;

— the coordination broker should monitor each of the irrigation entities comprising the virtual entity, and thus launch programmed/automated methods addressed to the subsystem level in accordance with the requirements established by its control logic;

— the coordination broker should execute the procedural elements addressed to the virtual entities whose logic resides in it. As with the other entities, the procedural elements should originate at the management level;

— the requests addressed over the management interface to a virtual entity may require data requests from the irrigation entities comprising it. These requests should be made over the subsystem interface. The requests related to the logic of an entity managed by the coordination broker should be resolved without the use of the interfaces;

— if the coordination broker takes charge of administering a virtual irrigation entity, it should be the coordination broker which responds to the requests made by the management interface compiling the information to include in the response;

— the coordination broker should enable choice of the entity or entities to be used as reference entity/ies for every property of the virtual entity over the user interface, as defined in E.6.12 Function 18: Virtual irrigation entities logic execution.

E.4.6 Function 17: Management of coordination broker access over the event interface

The subsystems implements an authentication method to access to the events server of the coordination broker. The available methods are described in the event interface implementation annexes (Annex C for SOAP 1.2 or Annex H for REST).

The coordination broker should offer in its user interface the management of the subsystems access, enabling creation, deletion and modification of permissions.

E.4.7 Function 18: Virtual irrigation entities logic execution

E.4.7.1 General

The coordination broker should identify the architectural level where the logic proper to the virtual irrigation entity is to be executed being able to execute:

— in the coordination broker; and

— in a subsystem. In this case, Function 18 is not be applicable to this virtual irrigation entity.

When the coordination broker is responsible for the logic execution of any virtual entity defined in the document, it behaves as a subsystem. Upon creating a virtual irrigation entity, also should be defined the entities contained and its operational conditions, according to the standard physical model.

When registering, it shall be identified which of the available logics defined in the physical model is to be executed, the irrigation entities contained by and which of them are selected as reference to obtain its properties values.

The coordination broker should verify that the requests addressed to the irrigation entities contained within a virtual entity do not entail a violation of its logic, restricting and blocking all those that do.

E.4.7.2 Procedural elements execution

Where the logic of an irrigation virtual entity resides in the coordination broker, this shall be responsible for executing the procedural elements associated to the entities given in Table E.15.

Table E.15 — Enumerated RecipeTypes for virtual entities

List of values	Executing entity	Procedural element Type
IrrigationGroupRecipe1	IHG	Operation
NetworkBranchRecipe1	NBU	Operation
IrrigationNetworkRecipe1	INT	Unit procedure

The logic to apply for each procedure is defined in the physical model.

E.4.7.3 Procedural elements status

The status of the procedural elements should be generated in the coordination broker through the aggregate operational statuses contained, updated through the events being recorded in the entities contained.

The possible statuses of procedural elements being executed in a virtual entity residing in the coordination broker are:

— Idle, in which all the procedural elements contained are in this status. This may contain procedural elements in Unknown status;

— Running, in which at least one of the procedural elements contained, is in this status. This may contain procedural elements in Unknown status; and

— Completed, in which all the procedural elements contained are in Stopped or Completed status (without differentiation). This cannot contain procedural elements in Unknown status.

E.4.7.4 Property management

E.4.7.4.1 General

The coordination broker should calculate the properties associated to the virtual irrigation entity, while respecting the definition and observing the following criteria.

An admissible error policy should be established, according to the number of reference entities to be used for the calculation of a property. It may be established different percentages admissibility for errors. If the percentage set as maximum error for a certain number of reference entities is not reached, the property should be calculated with all the values obtained. Otherwise, "3 - Execution error" should be returned.

It is a requirement the prior selection of the irrigation entities serving as reference for the property calculation.

E.4.7.4.2 ActivityStatus reading

When a reading request is received, the possible values for ActivityStatus should be obtained based on the ActivityStatus of the entities selected as reference entities. When a reading request is received, the coordination broker should obtain the property value using the values of this property for all its reference entities. During creation of the virtual irrigation entity, the user should identify the reference entities to be taken into account for calculating this property.

The following criteria are established for determining field 1 of the property:

— Active: When at least one of the reference entities is in this ActivityStatus. This may contain entities in Unknown status.

— Inactive: When all the reference entities are in this ActivityStatus. This may contain entities in Unknown status.

— Unknown: When all the reference entities are in this ActivityStatus.

The following criteria are established for determining field 2 of the property:

— OutOfOrder: When all of the reference entities are in this ActivityStatus substate. This may contain entities in Unknown status.

— Ready: When at least one of the reference entities is in this ActivityStatus substate. This may contain entities in Unknown status.

— Unknown: When the field 1 takes the value Unknown.

E.4.7.4.3 ActivityStatus writing

The ActivityStatus property may be modified through the Write method. When a writing request is received, the coordination broker follows the next criteria.

To write field 1:

— Active. Determination of this value for field 1 has no effect on the status of the reference entities.

— Inactive. Determination of this value for field 1 changes the status of all the reference entities to Inactive.

The write field 2:

— OutOfOrder. Determination of this value for field 2 changes the status of all the reference entities to OutOfOrder.

— Ready. Determination of this value for field 2 does not change the status of the reference entities.

E.4.7.4.4 SystemStatus

This property should only be available for those virtual irrigation entities with entity logic residing in the subsystem level. For any others, the coordination broker should respond with the set value “Not supported”.

E.4.7.4.5 Mode

This property should be available in some virtual irrigation entities, as established in this document. When a reading request is received, the coordination broker should obtain the property value using the values of this property for all its reference entities. During creation of the virtual irrigation entity, the user shall identify the reference entities to be taken into account for calculating this property.

In those cases where the property exists, its value should be calculated as follows:

— Programmed. When at least one reference entity is in this Mode.

— OnDemand. When at least one reference entity is in this Mode and none at Programmed.

— Monitoring. When all of the reference entities contained are in this Mode.

The coordination broker may establish acceptability thresholds to calculate the property value, when some of the contained entities respond with errors defined in this document. Acceptability thresholds can be different depending on the number of entities contained.

E.4.7.4.6 CumulativeVolumeOut

When a reading request is received, the coordination broker should add the values of the CumulativeVolumeOut properties for all the reference entities for this property. During creation of the virtual irrigation entity, the user should identify the reference entities to be taken into account for calculating this property.

The timestamp of the response to generate corresponds to the latest timestamp obtained from the request to the reference entities.

The writing of this property in the virtual entities is not supported.

The coordination broker may establish acceptability thresholds to calculate the property value, when some of the contained entities respond with errors defined in this document. Acceptability thresholds can be different depending on the number of entities contained.

E.4.7.4.7 PressureOut/PressureIn

This property should be available when its existence is reported in the static data for the virtual entity. The coordination broker should reflect PressureOut/PressureIn as the value recorded in the entity identified as reference for this property. The coordination broker should only allow the selection of one reference entity for each of these properties.

When a reading request is received, the coordination broker returns the property value from its reference entity.

In an irrigation hydrant grouping (IHGs), the PressureIn property may only be requested if the logic resides in the subsystem. In these circumstances, if a request is received and the logic resides in the coordination broker, it returns “3 – Not supported”.

E.4.7.4.8 InternalFlowOut

This property should be available when its existence is reported in the static data for the virtual entity. The coordination broker should sum up the values for the InternalFlowOut property on all the entities selected as reference; hence, in this case, multiple selections should be possible.

When a reading request is received, the coordination broker should add the values of the InternalFlowOut properties for all the reference entities for this property. The sum of such values should be admitted as valid provided that their time stamp falls within a window of 900 seconds. Otherwise, the Broker returns "1 – Execution error".

E.4.7.4.9 InternalFlowIn

This property shall be available when its existence is reported in the static data for the virtual entity. The coordination broker shall obtain InternalFlowIn from the entity that has been chosen as reference. Multiple reference entities may be selected, in which case the values of their InternalFlowIn properties shall be added to generate that of the virtual entity.

When a reading request is received, the coordination broker shall add the values of the InternalFlowIn properties for all the reference entities for this property. The sum of such values should be admitted as valid provided that their time stamp falls within a window of 900 seconds. Otherwise, the Broker returns "1 – Execution error".

In irrigation branches (NBUs) and networks (INTs), the value of this property may be obtained from reference entities not contained by the virtual entity. This linkage process is taken into account in E.5.6.

In IHGs entities, the property may only be requested if the logic resides in a subsystem. In these circumstances, if a request is received and the logic resides in the coordination broker, it returns "3 – Not supported".

E.4.7.4.10 CumulativeVolumeIn

This property should be available when its existence is reported in the static data for the virtual entity. When a reading request is received, the coordination broker should add the values of the CumulativeVolumeIn properties for all the reference entities for this property. Multiple reference entities may be selected in which case the values of their CumulativeVolumeIn properties should be added to generate that of the virtual entity. When one or more entities that are not part of the virtual entity are set as reference entities, the sum of their CumulativeVolumeOut values should be used to obtain CumulativeVolumeIn value.

In irrigation branches (NBUs) and networks (INTs), the value of this property may be obtained from reference entities not contained by the virtual entity. This linkage process is defined in E.5.6.

In IHGs entities, the CumulativeVolumeIn property may only be requested if the logic resides in the subsystem. In these circumstances, if a request is received and the logic resides in the coordination broker, it returns "3 – Not supported".

The writing of this property in the virtual entities is not supported.

The coordination broker may establish acceptability thresholds to calculate the property value, when some of the contained entities respond with errors defined in this document. Acceptability thresholds can be different depending on the number of entities contained.

E.4.7.4.11 OpeningDegree

The OpeningDegree property value should be the mean of the OpeningDegree property for all the entities contained.

The coordination broker may establish acceptability thresholds to calculate the property value, when some of the contained entities respond with errors defined in this document. Acceptability thresholds can be different depending on the number of entities contained.

E.4.7.5 Mandatory verifications

The verifications for the coordination broker includes:

— verifying compliance with the restrictions established in the logic attributed to the entity in the physical model;

— in the case of IHGs, the relationship between the IHG and its contained VIHs. Mandatory condition of existence for IHG is established, when exist one or more VIHs;

— in the case of NBUs, the mandatory identification of all the irrigation entities forming part thereof, respecting the foregoing restriction where contain IHGs. A mandatory condition of existence for NBUs is not established; and

— in the case of INTs, the mandatory identification of all the irrigation hydrants forming part of these, respecting the restrictions established with regard to NBUs and IHGs.

E.4.7.6 Desirable verifications

E.4.7.6.1 CumulativeVolumeIn=CumulativeVolumeOut

The coordination broker should be able to verify equality between the properties of CumulativeVolumeIn and ComulativeVolumeOut, provided that both exist in a virtual entity. In the event of disparity, the coordination broker should generate a "CumulativeVolumeIn≠CumulativeVolumeOut" event. The coordination broker enables definition of a threshold of acceptable differences between CumulativeVolumeIn and CumulativeVolumeOut in order to determine when the event should be generated.

E.4.7.7 Report generation

The coordination broker should generate reports of the procedural elements executed in a virtual irrigation entity. The report structure should fully respect the specifications established in this document. The coordination broker should have the capacity to:

— respond to ReadReport requests with both complete and partial reports;

— automatically generate complete reports once the procedural elements have finished;

— store the complete reports on all finished procedural elements.

E.4.7.8 Standard history generation

The coordination broker should generate the standard history for all properties attributed to a virtual irrigation entity. The standard history structure should fully respect the specifications established in this document. The coordination broker should have the capacity to:

— Respond to ReadStandardHist requests only when the request are for a day different from the current one;

— Generate an error "1 - Execution error" when the ReadStandardHist method is requested for the current day;

— Automatically generate standard histories for all irrigation entity properties, respecting the criteria established in this document; and

— Store the standard history of mandatory and extended properties of every virtual entity controlled by the Broker.

E.4.7.9 Event generation

The coordination broker should generate the events associated to the virtual irrigation entity. The events to generate are those defined for the virtual irrigation entity in the physical and procedural models.

E.5 Specific requirements

E.5.1 General

The detailed description of the functional and non-functional system requirements to allow the development of a coordination broker should be identified unequivocally according to the following criteria.

Firstly, they are classified by nature:

— RF – To indicate functional requirements;

— RNF – To indicate non-functional requirements; and

— RES – To indicate restrictions.

Secondly, the requirement identifier should be followed by a number indicating the function to which it relates from among those included in E.3 and E.4.

Finally, a period should be added, followed by a requirement number.

E.5.2 User interface requirements

The coordination broker should avail of a user interface for configuration and for the execution of the functions described that are required from this interface.

E.5.3 Communication interfaces requirements

The coordination broker should avail of as many communication modules as there are communication protocols implemented for each of the interfaces defined, following the specifications in the implementation annexes of each protocol used in:

— the management interface;

— the subsystem interface; and

— the event interface.

The coordination broker should implement, at least, one of the communication protocols defined in this document for each interface.

E.5.4 Minimal functions

E.5.4.1 Function 1: Unequivocal identification of irrigation entities

Requirement number	RF1.1		
Requirement name	Irrigation entities should be identified on all architecture levels using a unique EntityID.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should use EntityID as an unequivocal identifier for the irrigation entities it manages. The coordination broker should respond to requests over the management interface, regarding the irrigation entities managed by the coordination broker, provided the unique EntityID is respected. The coordination broker should send requests to the subsystems over the subsystem interface, using EntityID and the access path to each subsystem. No multiple irrigation entities with the same EntityIDs should exist at the same process cell. The coordination broker should generate error messages over the user interface when there are inconsistencies in the data of an irrigation entity during the creation process. The system should not permit the creation of entities with data considered erroneous or inconsistent.		

E.5.4.2 Function 2: Irrigation entity administration

Requirement number	RF2.1		
Requirement name	Irrigation entity creation.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should enable the creation of an irrigation entity over its user interface, respecting the data model described in E.2. The coordination broker should ensure that the data entered for an irrigation entity are correct. If not, it should report the detected errors. Such checks are described in RF2.4. The coordination broker should add the identifier for the irrigation entity being created to the ChildEntities field of the entity exercising as ParentEntity. The coordination broker should add the identifier for the irrigation entity being created to the SuccessorEntities field of the entity exercising as PrecursorEntity.		

Requirement number	RF2.2		
Requirement name	Irrigation entity deletion.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional

Description	The coordination broker should enable deletion of existing irrigation entities over its user interface. The entity deleted should cease to be available over the communication interfaces, but should remain accessible for administration from the user interface. The coordination broker should warn the user of all the dependencies to be affected by way of parent/child and precursor/successor entities. Deletion of an irrigation entity should not be allowed as long as it continues to have child and successor entities. The coordination broker should delete the identifier of the irrigation entity to be deleted from the ChildEntities field of the entity exercising as ParentEntity as well as from de SuccessorEntities field of the entity exercising as PrecursorEntity. Any entity deleted may once more be created if the verification conditions are met RF2.4.

Requirement number	RNF2.3		
Requirement name	Irrigation entity modification.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should enable modification of existing irrigation entities over its user interface. All the values except EntityID may be changed. The coordination broker should ensure that the data entered for an irrigation entity are correct. Such checks are described in RF2.4. The coordination broker should warn the user of all the dependencies to be affected by way of parent/child and precursor/successor entities. No changes in EntityLevel and EntityType fields that generate dependency inconsistencies by altering the relationships should be allowed.		

Requirement number	RNF 2.4		
Requirement name	Data verification for irrigation entities.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should run a check on the irrigation entities data entered. Such verification should be conducted in two situations: on creating a new irrigation entity and on making a modification to the data of an existing one. Following items should be checked that: — the coordination broker should not allow the existence of two identical EntityIDs and should report the error over the user interface, disallowing completion of such new entity creation; — the data entered comply with the format specified in this document; — the enumerated can only take on the values described in E.2; — the “EntityLevel” and “EntityType” fields respect the Physical Model; — any irrigation entity occupying process cell level has a “ParentEntity”; and — any irrigation entity, unless otherwise specified, has a “PrecursorEntity”.		

Requirement number	RNF2.5		
Requirement name	Connection checks between irrigation entities.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should check the hydraulic connections formed upon the creation of irrigation entities to ensure that the connection between the entities is correct in accordance with the PrecursorEntities-SuccesorEntities relationships, diameters, materials and sections identified in each connection.		

Requirement number	RNF2.6		
Requirement name	Path validation.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☐ Medium/Desirable	☒ Low/Optional
Description	To finalize the creation process, the coordination broker should run a communication test with the reporting subsystem in the Path field to check whether the connection between applications has been established correctly.		

E.5.4.3 Function 3: Requests propagation over the management and subsystem interfaces

Requirement number	RF3.1		
Requirement name	Requests propagation over interfaces.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should accept requests over the management interface. Should the data be available in the coordination broker, it should return the response and not propagate the request over the subsystem interface. Requests for data should be propagated when the coordination broker does not have it, launching the request over the subsystem interface. Requests for which the addressee is the coordination broker itself should not be propagated over the subsystem interface (methods defined in this document not common to both interfaces and virtual irrigation entities controlled by the coordination broker). For the Management and Subsystem interfaces, Table E.13 describes the requests to be propagated directly to the subsystems upon receipt by the management interface, and those to be propagated if the data is not available in the coordination broker. The requests coming from the subsystems through the event interface is received by the coordination broker but not propagated through the management interface.		

Requirement number	RF3.2		
Requirement name	Available methods for requests to the coordination broker over the management interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional

Description	The methods available at the management interface are those defined in this document and in its implementation annexes (Annex A or Annex F): CreateRecipe, ReadProceduralIDs, Write, Read, ReadStandardHist, ReadEvent, StopRecipe, ReadReport, ReadEntityIDs, ReadEntityData and GetProceduralIDs.

Requirement number	RF3.3		
Requirement name	Available methods for requests to the subsystems from the coordination broker over the subsystem interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The methods available at the subsystem interface are those defined in this document and in its implementation annexes (Annex B or annex G): CreateRecipe, Write, Read, ReadstandardHist, StopRecipe, ReadReport, SubscribeEvent, UnSubscribeEvent and ReadProceduralIDs.		

Requirement number	RF3.4		
Requirement name	Available methods for requests from the subsystems over the event interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The methods available at the event interface are those defined in this document and in its implementation annexes (Annex C or Annex H): NewEvent.		

E.5.4.4 Function 4: Coordination broker authentication

Requirement number	RF44.1		
Requirement name	Security certificate of the coordination broker.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should guarantee its authenticity, using a security certificate that enables the creation of a secure encrypted communications channel.		

E.5.4.5 Function 5: Management of coordination broker access over the management interface

Requirement number	RF5.1		
Requirement name	Management of coordination broker access over the management interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	Any MIS requesting access to the coordination broker over the management interface should be registered. Otherwise, access should be denied. The possible methods used for access control are described in the implementation annexes for the management interface (Annex A or Annex F).		

Requirement number	RF4.2		
Requirement name	User interface for access control.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker user interface should offer the possibility of creating, deleting and modifying the users allowed to access over its management interface. During the permission creation for a MIS, the coordination broker should: — Provide the access data; — Select the entities to which the user has access in accordance with E.3.6; — Select the methods that the user may execute on the entities to which s/he has access in accordance with E.3.6. For every instance of access, the authentication should be completed.		

E.5.4.6 Function 6: Management of events subscription and unsuscription

Requirement number	RF6.1		
Requirement name	Management of coordination broker access over the event interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	Any subsystem requiring access to the coordination broker over the event interface should be registered. The possible methods used for access control are described in the implementation annexes for the event interface. If the subsystem does not have access permit, it should be denied and the "1 - Execution error" message returned.		

Requirement number	RF6.2		
Requirement name	User interface for access through event interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker user interface should offer the possibility of creating, deleting and modifying the users allowed to access to the coordination broker over the event interface. During the permission creation for a subsystem, the coordination broker should provide the access data. For every instance of access, the authentication should be completed.		

E.5.4.7 Function 7: Attending to requests through the management interface

Requirement number	RF7.1		
Requirement name	Attending to requests through the management interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional

Description	The coordination broker should avail servers with the capacity to manage concurrent connections for multiple clients. It implements servers for any communication protocol with an available implementation annex for the management interface (Annex A or Annex F).

Requirement number	RNF7.2		
Requirement name	Connections with MIS applications.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should administer the requests and connections of different MIS, generating the access data required to enable MIS authentication, in accordance with the authentication methods defined in the management interface implementation annexes (Annex A or Annex F).		

Requirement number	RF7.3		
Requirement name	Use of the management interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The requests received by the coordination broker should respect the management interface, as well as the methods defined in the document and the management interface implementation annexes (Annex A or Annex F).		

E.5.4.8 Function 8: Connection with subsystems

Requirement number	RF8.1		
Requirement name	Connection with subsystems 1.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should avail of clients with a capacity for concurrent connections to multiple subsystems. It implements clients for any communication protocol with an available implementation annex for the subsystem interface.		

Requirement number	RF8.2		
Requirement name	Connection with subsystems 2.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should administer the requests and connections with different subsystems, storing the required data for its authentication in accordance with the authentication methods included in the subsystem interface implementation annexes. It should use the communications protocol reported for each irrigation entity in E.2.5.4.		

Requirement number	RF8.3		
Requirement name	Use of the subsystem interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should connect with the subsystems using the subsystem interface, as well as the methods defined in this document and the subsystem interface implementation annexes (Annex B or Annex G).		

E.5.4.9 Function 9: Subsystem access management

Requirement number	RF9.1		
Requirement name	Secure subsystem access.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should access subsystems securely over the subsystem interface. The data to be registered in the coordination broker to access a subsystem is defined by the authentication method implemented, among all those included in the implementation annexes for the subsystem interface (Annex B or Annex G).		

Requirement number	RF9.2		
Requirement name	Subsystem access management.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker user interface should offer the possibility of creating, deleting and modifying the users required to access a subsystem over the subsystem interface. For each access, the coordination broker provides the security data required by the subsystem, according to the method that has been implemented from among those defined in the implementation annexes for the subsystem interface (Annex B or Annex G).		

E.5.4.10 Function 10: Subsystem entity access levels

Requirement number	RF10.1		
Requirement name	Subsystem entity access level management.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should offer a client subsystem access setup. Configuration of the access level would require: — The MIS to have an access permit by registering this in the coordination broker in accordance with E.5.7. — The subsystem to which access is permitted to have been previously registered in the coordination broker in accordance with E.5.9. — The prior selection of its accessible entities and the methods admitted for		

	each MIS. The methods should be those described in the document for the subsystem interface. The coordination broker should accept requests if the MIS has access to the irrigation entity and to the method to be used. Otherwise, access should be denied and the response over the subsystem interface is the "2- Lexical error".

E.5.4.11 Function 11: Management of events subscription and unsuscription

Requirement number	RF11.1		
Requirement name	Subscribing/unsubscribing to the events of an irrigation entity.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	Over its user interface the coordination broker should offer the subscription and unsubscription to the events occurring in an irrigation entity. Subscribing/unsubscribing applies to all the events of the irrigation entity, there being no mechanism to subscribe to specific events. The coordination broker uses two standard methods over the subsystem interface (Annex B or Annex G): — the SubscribeEvent method to subscribe; and — the UnsubscribeEvent method to unsubscribe.		

Requirement number	RF11.2		
Requirement name	Attending to requests through the event interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should avail of a server with the capacity to manage connections for multiple clients in order to attend the report of events. It implements servers for any communication protocol with an available implementation annex for the event interface (Annex C or Annex H).		

Requirement number	RF11.3		
Requirement name	Subsystem connections to report events.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should administer the requests and connections with the subsystems wanting to report events, generating the access data required to enable the subsystem authentication, in accordance with the authentication methods defined in the event interface implementation annexes (Annex C or Annex H).		

Requirement number	RF11.4		
Requirement name	Use of the event interface.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☒ High/Essential	☐ Medium/Desirable	☐ Low/Optional
Description	The requests received by the coordination broker should respect the event		

	interface, as well as the methods defined in the document and the management interface implementation annexes (Annex C or Annex H).

E.5.4.12 Function 12: Request verification

Requirement number	RF12.1		
Requirement name	Verification of the parameters contained in requests for all the methods (except for CreateRecipe).		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	For each possible protocol, the coordination broker should check that all requests adjust strictly to the definitions of its implementation annex for the management interface. The coordination broker should not propagate requests if: — Are badly formulated. — Are addressed to irrigation entities with non-existent EntityIDs. — Are addressed to procedural elements with non-existent ProceduralIDs. — The requests are incompatible with the definitions of the standard. It should return a "2 - Lexical error" whenever the content of a request does not adjust to the format defined, to the expected sequence or when includes incompatible parameters with the standard definitions.		

Requirement number	RF12.2		
Requirement name	Verification of the fields contained in requests for the CreateRecipe method.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should run the same checks defined in RF11.1, but only on the parameters defined in E.3.4.		

E.5.5 Desirable functions

E.5.5.1 Function 13: Standard history administration

Requirement number	RF13.1		
Requirement name	Configuration of standard history for programmed collection 1.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker user interface should enable configuration for the programmed collection of standard histories defined at the physical model of this document. The programmed collection should be set up using two ReadStandardHist method parameters: NumHist and Statistics. The admissible values for these two parameters are identified in the standard. Given any value that does not adjust to specifications, the coordination broker should return an error message without completing the programmed collection configuration.		

Requirement number	RF13.2		
Requirement name	Configuration of standard history for programmed collection 2.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The programmed collection can be configured for each irrigation entity.		

Requirement number	RF13.3		
Requirement name	Obtaining standard history.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should obtain the standard history of all the mandatory and extended properties for each irrigation entity in keeping with RF3.1 and RF3.2. A periodic daily request should be launched to obtain all the properties of each entity, using the subsystem interface. The data obtained should be saved in a persistent data bank. Should a request received from the management interface is adjusted to the programmed collection configuration, the coordination broker should respond with data from its data bank. Otherwise, a request over the subsystem interface is performed to obtain the information and to respond to the original request. The ReadStandardHist method is used to obtain standard histories.		

Requirement number	RF13.4		
Requirement name	Storing standard history.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should store the standard histories obtained from the subsystems on a persistent data system using the model described in E.3.3. Storable volume should adjust to the criteria of the developer/user and their requirements.		

Requirement number	RF13.5		
Requirement name	Frequency in obtaining standard history.		
Type	☐ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☐ Medium/Desirable	☒ Low/Optional
Description	Programmed reading frequency for the standard history should be once a day and should always be conducted at the end of a calendar day (this may only be requested once the calendar day has ended).		

Requirement number	RF13.6		
Requirement name	Properties for which standard history are to be requested.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should request the standard history on all the mandatory properties of an irrigation entity. In addition, it should request the standard history on all the extended properties reported upon the creation of the entity.		

E.5.5.2 Function 14: Procedural elements administration

Requirement number	RF14.1		
Requirement name	Procedural elements creation.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The procedural elements creation should be performed using the CreateRecipe method. When the coordination broker receives a request over the management interface, it should: a) verify that the ProceduralID does not exist; b) register its database the data on the request reflected in E.2.4; c) initialize Status with the value "Idle" from among the options reflected in E.2.4.2; d) propagate the request to the subsystem responsible for the irrigation entity for which is intended.		

Requirement number	RF14.2		
Requirement name	Updating the Status.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The Status should be updated using ReadReport and saving its value in the coordination broker data bank. The value should not be changed until a new ReadReport request is made.		

Requirement number	RF14.3		
Requirement name	Verifying that the irrigation entity supports the RecipeType.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should check whether the procedural element is supported by the irrigation entity obliged to execute it, as defined in the procedural model of the standard. Hence, if a given RecipeType cannot be executed for a certain type of irrigation entity (EntityType) as defined in this document, the coordination broker should return a "2 – Lexical error".		

E.5.5.3 Function 15: Report administration

Requirement number	RF15.1		
Requirement name	Obtaining complete reports.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should obtain the complete reports from all the procedural elements processed. To identify when a procedural element has its complete report, the status is Stopped or Completed has to be verified. This verification is performed using the events indicating the end of the procedural element (StoppedOperation or CompletedOperation). When the completion has been verified, the coordination broker should use the ReadReport method to retrieve the complete report. The coordination broker should store all obtained complete reports in a data bank. When a request for a report is received from the management interface, the coordination broker should verify if it is stored in its data bank. If not, the report has not been completed yet, but a partial report can be obtained by request to the subsystem. Should the report requested correspond to a procedural element in Idle, Running or Unknown status, the coordination broker should forward the request to the subsystem. In this case, the response does not be saved in the data bank, this being ad hoc information.		

Requirement number	RF15.2		
Requirement name	Storing complete reports.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker shall store the information obtained from the subsystems on a persistent data system using the model described in E.3.2. The coordination broker shall have the capacity to store complete reports. Storable volume shall adjust to the criteria of the developer/user and their requirements.		

E.5.5.4 Function 16: Data bank

Requirement number	RF16.1		
Requirement name	Coordination broker configuration data storage.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should keep in a single database the user configuration data on: — programmed collections (standard history and reports); — configurations of authorizations for different users; — authorized MIS to access the management interface; — subsystems authorized to access the event interface; — users authorized for subsystem to access its subsystem interface; — other basic coordination broker configurations that depend on user criteria.		

Requirement number	RF16.2		
Requirement name	Coordination broker Events log.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/ Optional
Description	The coordination broker should avail of an event log with identification (date, time and user) for: — changes in coordination broker configuration; — requests received over the management interface, identifying the sender; — requests received over the event interface, identifying the sender; — requests send over the subsystem interface, identifying the receiver; — responses obtained from the subsystem interface, identifying the sender; — requests given over the management interface, identifying the receiver; — failures and errors in the coordination broker; — records of backups made; — records of creation/deletion for each irrigation entity; — records of actions executed on virtual entities controlled by the coordination broker upon prior request over the management interface; and — records of actions and changes in configuration over the user interface, executed on virtual entities controlled by the coordination broker.		

Requirement number	RF16.3		
Requirement name	Static data bank for irrigation entities.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should keep all the static data defined in this document for each irrigation entity created in the coordination broker, even though they are subsequently deleted.		

Requirement number	RF16.4		
Requirement name	Standard histories maintenance.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should maintain a data bank for the standard histories obtained through E.4.1 in keeping with E.2.3.		

Requirement number	RF16.5		
Requirement name	Procedural elements records.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should maintain a data bank with the data on the		

	procedural elements received and managed, in keeping with E.4.3 and E.2.4.

Requirement number	RF16.6		
Requirement name	Maintenance of reports in the data bank.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should keep a data bank of complete reports (those in which the Status of the procedural element is Completed or Stopped) in keeping E.4.3 and E.2.2.		

Requirement number	RF16.7		
Requirement name	Data bank backup configuration.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should avail of a mechanism to configure backups. Configuration should be based on a maximum time between backups and a maximum size set as admissible for the data bank.		

Requirement number	RF16.8		
Requirement name	Irrigation entity event log.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should maintain a data bank with the events that have occurred in all the irrigation entities to which it has subscribed.		

Requirement number	RF16.9		
Requirement name	Access to and review of data kept in the data bank.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker user interface should enable access to the data stored in the data bank. The access should avail of a search and filter function for reviewing standard histories, entity events, registered procedural elements, entities static data and reports.		

Requirement number	RF16.10	
Requirement name	Coordination broker event log access and review.	
Type	☒ Requirement	☐ Restriction

Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker user interface should enable access to its event log. The access should avail of a search and filter function to review the contents of the said log.		

E.5.5.5 Function 17: Management of virtual irrigation entities

Requirement number	RF17.1		
Requirement name	Steps in the creation of virtual irrigation entities.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	On creating a virtual entity in the coordination broker, the following should be selected: — the entities comprising it; and — among the entities comprising it, the reference entities for generating its properties. These irrigation entities should have been previously registered in the coordination broker. The coordination broker should be responsible for executing its logic, provided HigherControlLevel as the value assigned during the registration for E.2.5.6.		

Requirement number	RF17.2		
Requirement name	Procedural elements validation and execution.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should validate the procedural elements intended to the virtual entity that controls and that has been received from the management interface. Once a procedural element has been validated, the coordination broker should execute it launching over the subsystem interface the requests required to perform the irrigation entity logic.		

Requirement number	RF17.3		
Requirement name	Coordination broker behavior as subsystem.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should act as a subsystem for controlling the virtual entities to which its control has been assigned. The subsystem interface should not be used, since the coordination broker controls the virtual irrigation entity.		

E.5.5.6 Function 18: Virtual irrigation entity logic execution

Requirement number	RF18.1		
Requirement name	Virtual entity logic compilation.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should execute any logic attributed to a virtual entity that it controls. The logic to be executed should be one of the possible for the entity, attending to the standard. The possible behaviors are defined in E.2.5.8. If the virtual entity cannot execute differentiated logics, this parameter does not be taked into account by the coordination broker.		

Requirement number	RF18.2		
Requirement name	Validation of requests regarding a virtual irrigation entity.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should run checks in addition to those defined in E.3.11 to validate the contents of the requests. The coordination broker should check to see that CreateRecipe requests do not include any incompatible parameter: — with the entity modelling entered through its static data; and — with the logic defined for the entity. In both cases it should return a “3”- Not supported message.		

Requirement number	RF18.3		
Requirement name	Virtual irrigation entity logic execution.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should execute the logic associated to the virtual entity in accordance with the definition in this document. For that, the coordination broker performs all requests over the subsystem interface necessary to execute the procedural elements and guarantee its proper performing.		

Requirement number	RF18.4		
Requirement name	Virtual irrigation entity ActivityStatus generation.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should generate status readings for the virtual entities it controls. The ActivityStatus should be obtained based on the ActivityStatus of the		

	entities that it contains, in accordance with the criteria established in E.4.7.4.

Requirement number	RF18.5		
Requirement name	Virtual irrigation entity procedural elements status generation.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should generate the status for the procedural elements executed by the virtual entity under its control. The status of a procedural element should be generated from the status of the procedural elements of the irrigation entities that it contains, as defined in E.4.7.3.		

Requirement number	RF18.6		
Requirement name	Generating properties.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should generate the values of the properties assigned to a virtual entity under its control. To generate these, it should make the necessary requests through the subsystem interface. Properties should be generated as defined in E.4.7.4.		

Requirement number	RF18.7		
Requirement name	Generating reports, standard histories and events.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should generate the reports of the procedural elements executed by virtual entities under its control, attending the requests over the management interface. Likewise, it should generate the standard histories of all the mandatory and extended properties reported for each virtual entity. It should also generate the events associated to their properties and to their procedural elements. Administration of all these should observe the definitions contained in E.4.7.7, E.4.7.8 and E.4.7.9.		

Requirement number	RF18.8		
Requirement name	Generating responses to requests addressed to virtual irrigation entities.		
Type	☒ Requirement	☐ Restriction	
Requirement priority	☐ High/Essential	☒ Medium/Desirable	☐ Low/Optional
Description	The coordination broker should generate the responses to the requests addressed to the virtual irrigation entities under its control. Whenever the request refers to an existing property, standard history or procedural element, it should respond with the data requested and “0- Action successfully executed message”.		

	Whenever the request refers to a non-existent property, standard history or procedural element, it should respond with a “2 - Lexical error” message.

E.6 Request management

The criteria to manage the requests by the coordination broker are presented in the following diagrams:

— Diagram 1. Requests from the management interface not propagated to a subsystem. (ReadEntityIDs, ReadEntityData, GetProceduralIDs and ReadEvent). See Figure E.1.

— Diagram 2. Requests from the management interface propagated to a subsystem. (ReadProceduralIDs, Write, Read and StopRecipe). See Figure E.2.

— Diagram 3. Requests from the management interface propagated to the subsystem for which the coordination broker should save data in its database. (CreateRecipe, ReadStandardHist and ReadReport). See Figure E.3.

— Diagram 4. Requests made by the coordination broker because a StoppedOperation or CompletedOperation procedural event has been received requesting a report on an ended procedural element (ReadReport). See Figure E.4.

— Diagram 5. Programmed collections made by the coordination broker to obtain the standard history of an entity. See Figure E.5.

— Diagrams 6 and 7. Requests made by the coordination broker to subscribe or unsubscribe to the events of an irrigation entity (SubscribeEvent and UnsubscribeEvent). See Figures E.6 and E.7.

— Diagram 8. Request to the coordination broker made by subsystems where events have occurred (NewEvent). See Figure E.8.

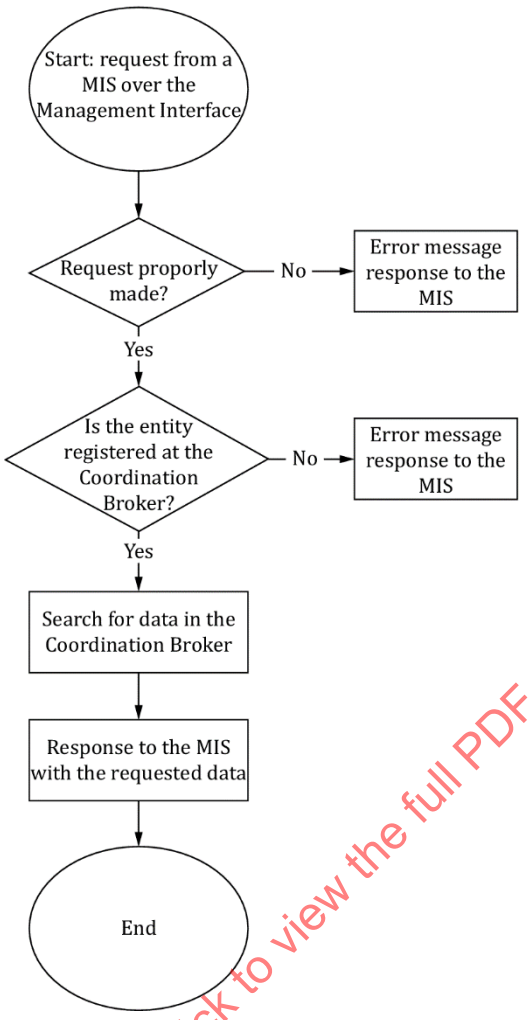


Figure E.1 — Requests from the management interface that are not propagated

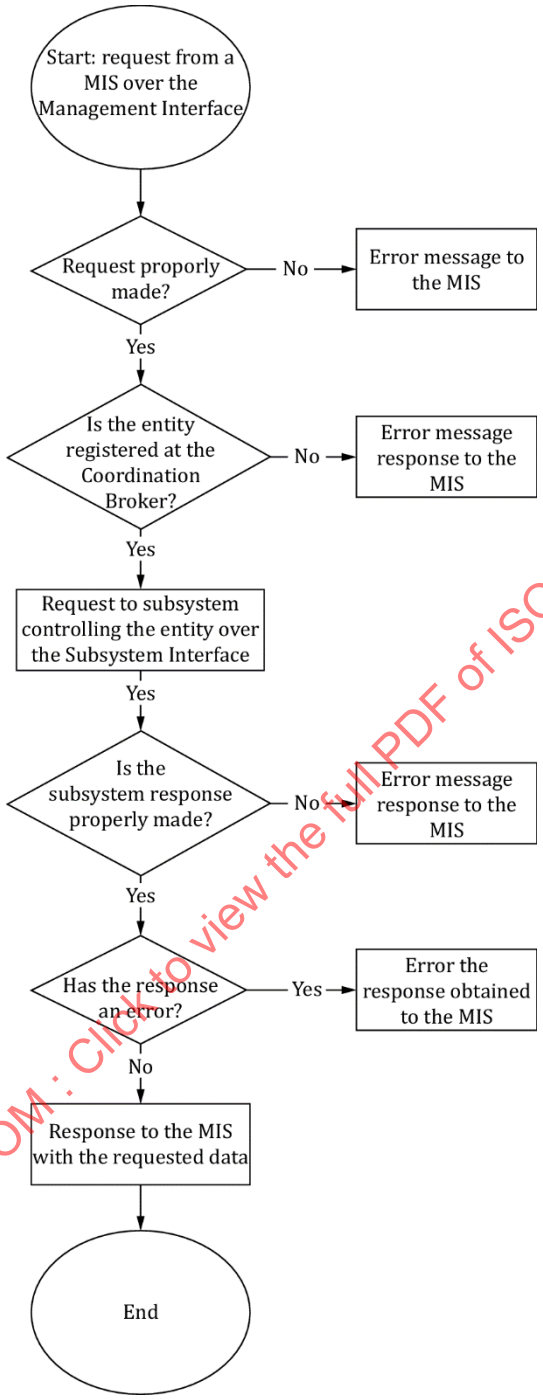


Figure E.2 — Requests from the management interface that are propagated

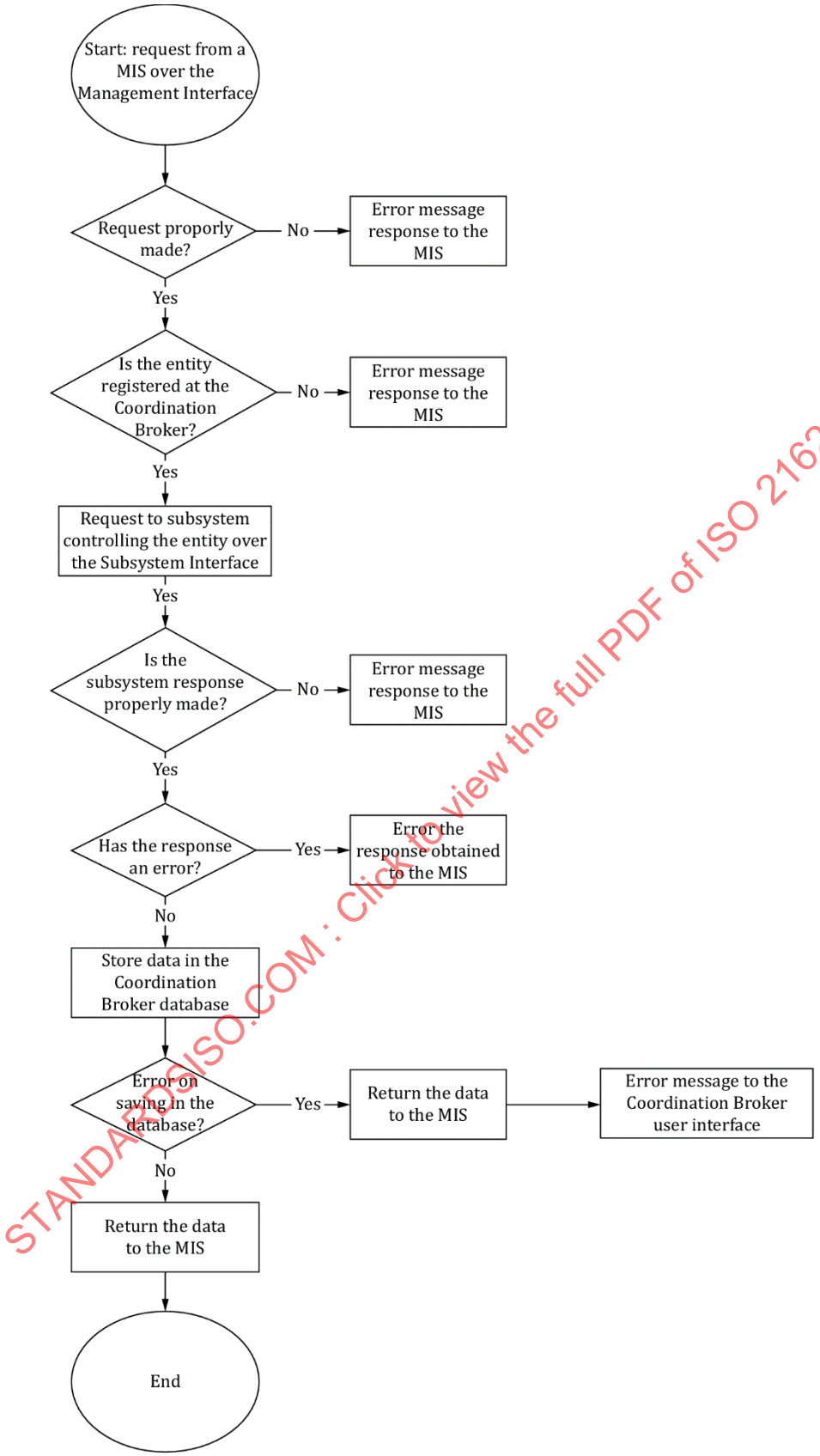


Figure E.3 — Requests from the management interface propagated to the subsystem

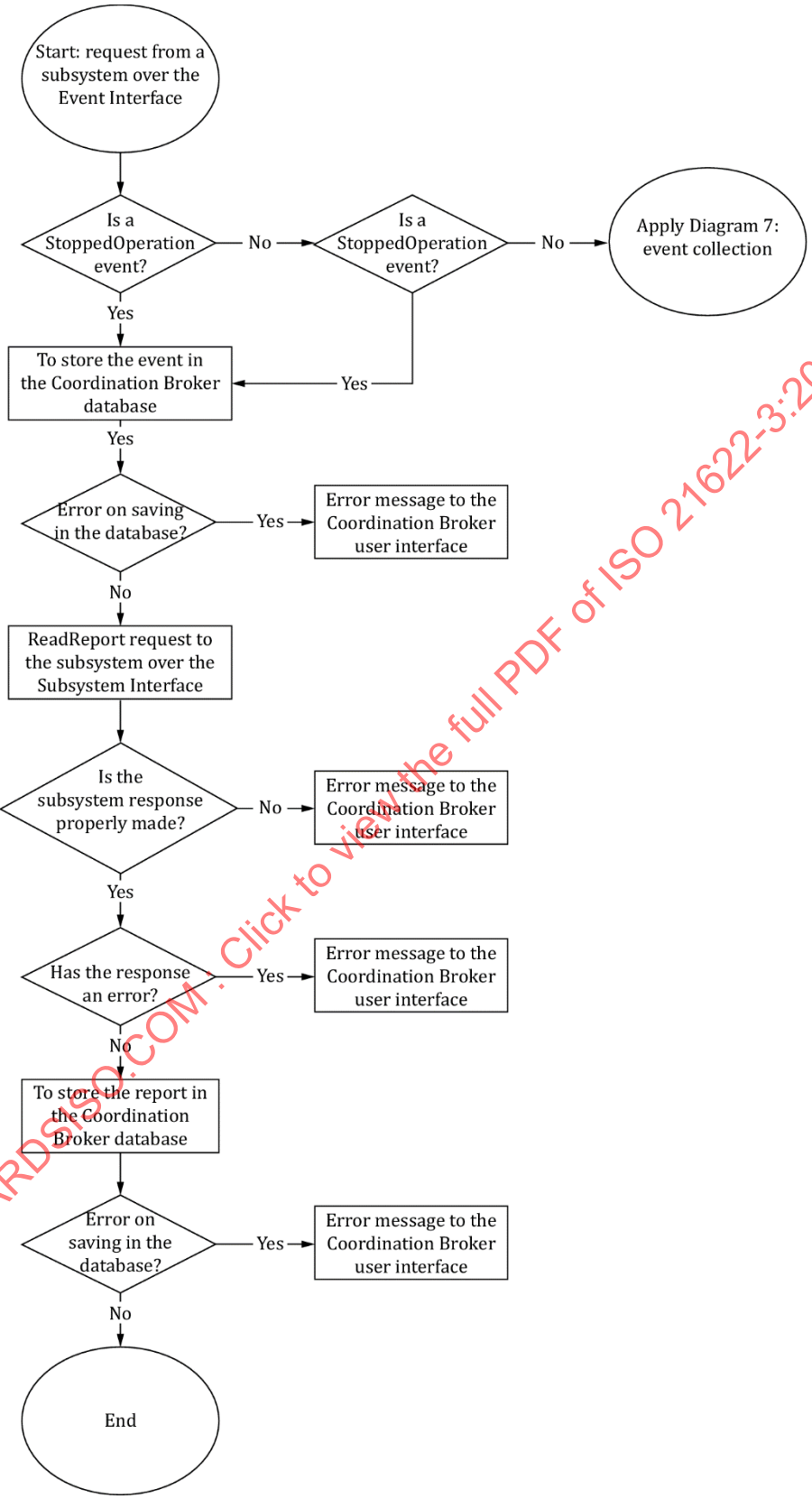


Figure E.4 — Automatic report collection

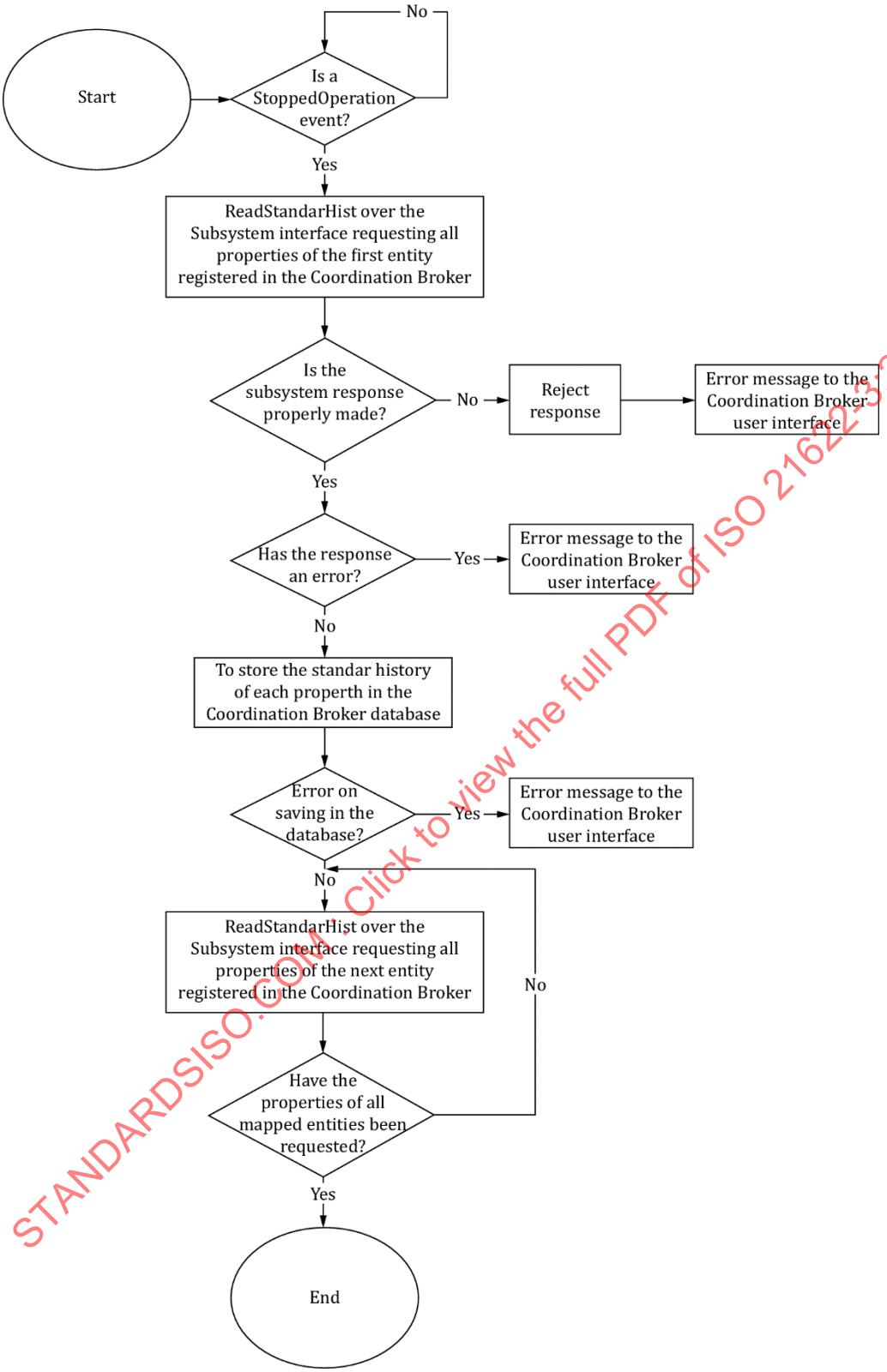


Figure E.5 — Standard history collection

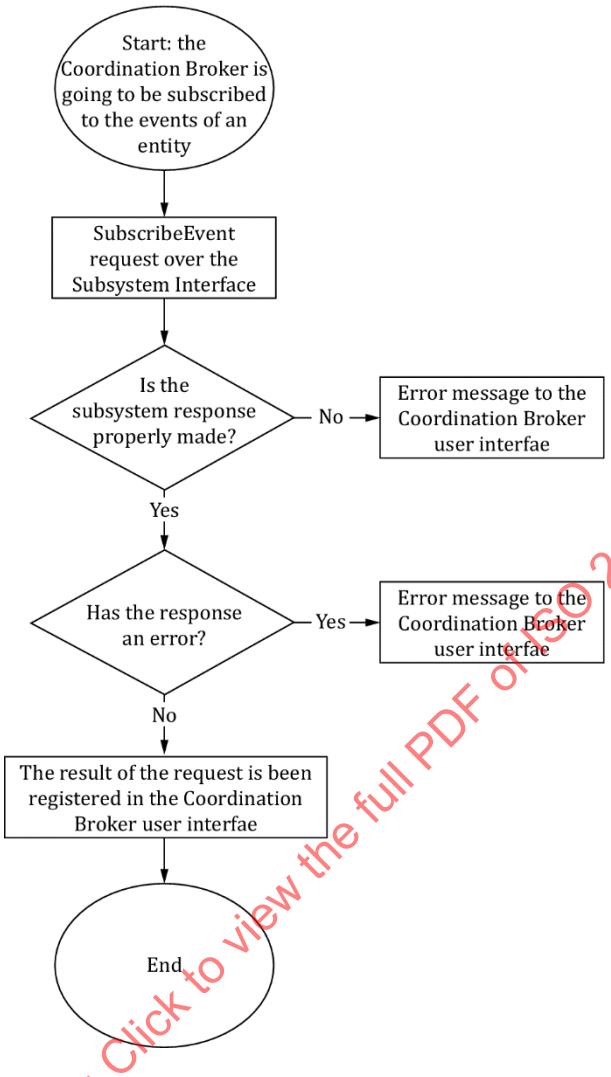


Figure E.6 — Coordination broker request to subscribe to the events of an irrigation entity

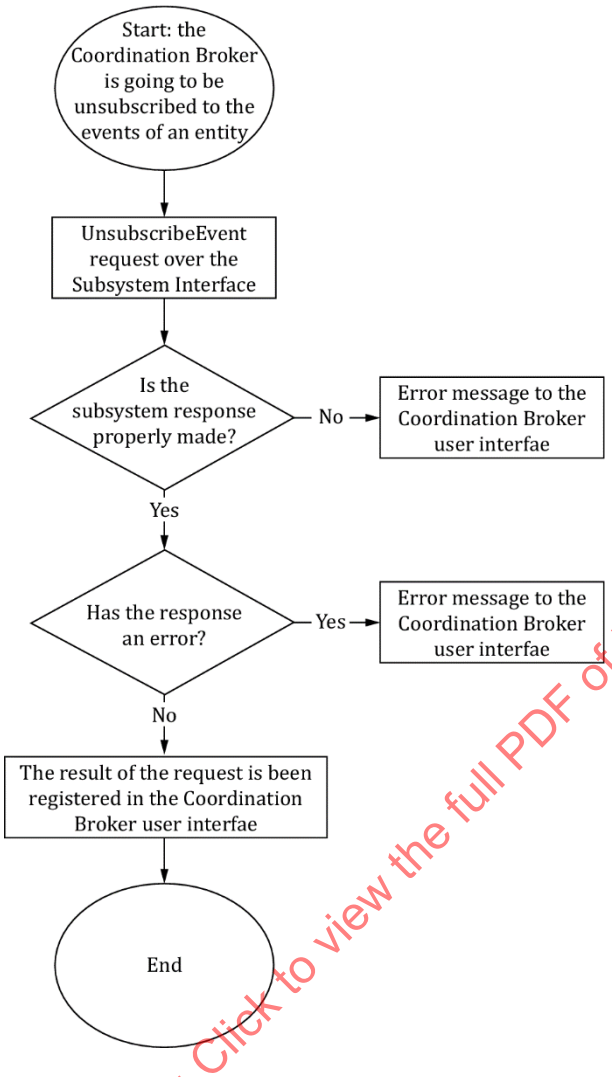


Figure E.7 — Coordination broker request to unsubscribe to the events of an irrigation entity

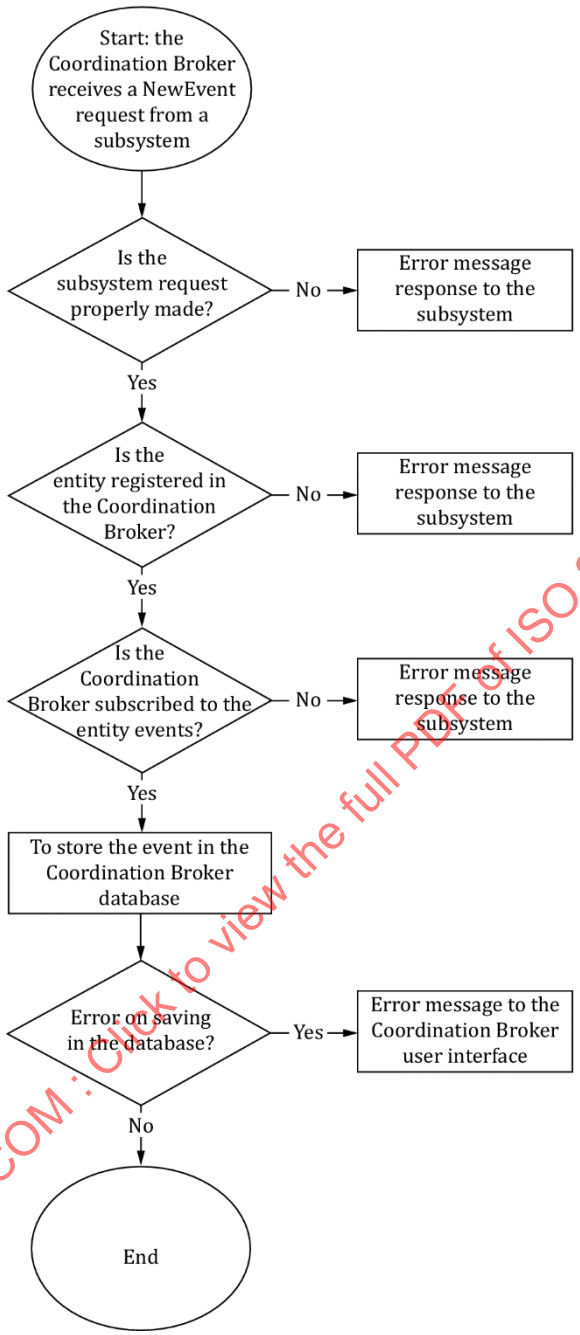


Figure E.8 — Event collection

E.7 Cases of use

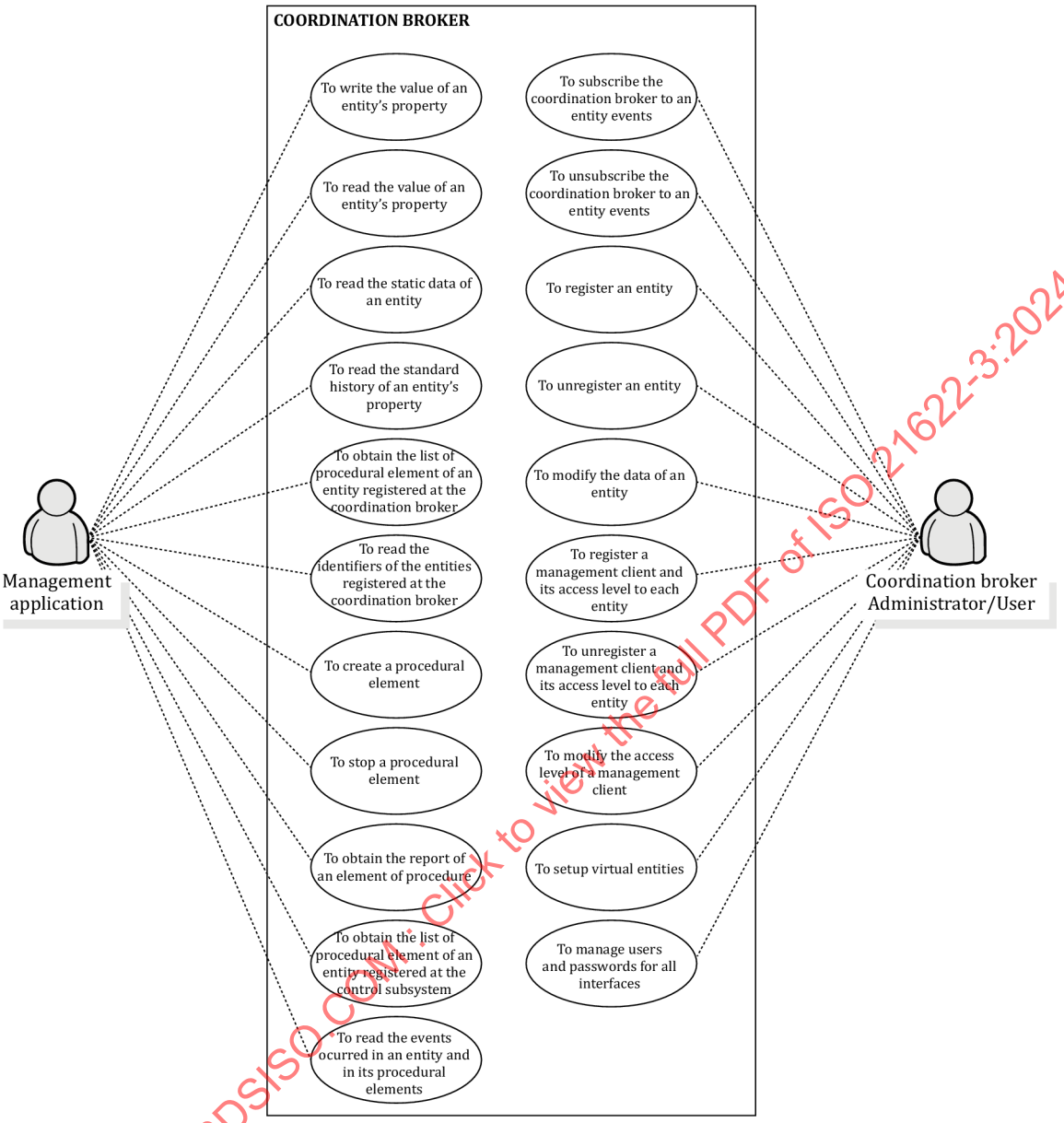


Figure E.9 — Cases of use

Annex F
(normative)

Management interface with REST

F.1 Overview

This annex outlines the information needed to implement the management interface through Web Services and REST technology. This implementation permits MIS applications to retrieve data from the upper control level: coordination. Information can be generated at this control level or requested to the lower control level (subsystems). MIS applications only need to implement the methods required to perform their management tasks.

This annex concerns MIS and coordination brokers adapted to this document. The interface between these architecture elements is addressed by a Web Service:

— Upper control level: the coordination broker publishes a Web Service as a server; and

— Any application located at management level acts as a Web Services client.

This annex defines a set of methods available for any MIS applications by its publication by the upper control level, where the coordination brokers are located.

F.2 Requirements

F.2.1 General

The names of the classes and the methods presented in this annex shall be used for the implementation of the management interface when using REST. Likewise, shall be used the following protocols defined by the Web Services Protocol Stack:

— HTTP 1.1 (Hypertext Transfer Protocol) RFC 2616;

— TLS within HTTP/1.1 RFC 2817;

— TLS 1.2 RFC 5246;

— OAuth 2.0 Framework RFC6750 and RFC6749;

— REST (Representational State Transfer). The specifications of this architecture style is available at https://restfulapi.net/;

— WADL 2.0 (Web Application Description Language). Language of web service's definition. The specifications of this language are available at https://www.w3.org/Submission/wadl/; and

— JSON (JavaScript Object Notation). The language of exchanging data selected is available at https://tools.ietf.org/html/rfc7159

F.2.2 Data protection

Data protection regulations in the jurisdiction of use can apply.

F.2.3 Security

The REST interface described in this document shall use Oauth2. Therefore, a request shall first be made to the authorization server. This server provides a token to access the services described in this document. The servers shall have a digital certificate for ensure the security through the communication channel defined, using HTTPS over TLS 1.2.

The security criteria defined in this annex shall to be understood as minimal, being possible to establish extra criteria to access data.

F.2.4 Header

The header of all requests and responses, except those intended for the authorization server, shall be encapsulated in an HTTP 1.1 transaction; the HTTP 1.1 header shall include the following attribute:

— Content-Type: application / json; Charset: utf-8; Date: YYYYMMDDhhmmss±hhmm; Authentication: hmac user: Bearer token

The header of requests for the authorization server shall include:

— Content-Type: application/x-www-form-urlencoded

F.2.5 Servers requirements

Any application located at the upper control level, shall provide the Uniform Resource Identifier (URI) for each Web Service. The main structure of the Web Services is the following URI:

— http://<domain_name>:<port>/rest/ManagementServices

The structure of the authorization server is the following URI:

— http://<domain_name>:<port>/rest/Security

F.2.6 Web Application Description Language (WADL)

The WADL is a description of the web service published by the server that establishes a contract between the server and the clients. It specifies the interface through which a client can gain access to the service and details about how it shall be used.

The server and the clients shall fully respect the WADL described in this annex (even the sequence of the parameters) to ensure communication through the interface. The WADL is composed by the implementation classes defined below.

F.2.7 Classes and enumerations

The published Web Services can be case sensitive for enumerated values. The UpperCamelCase convention shall be used in all classes and enumerations.