IEC 60839-11-32

Edition1.0 2016-11

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside



## Alarm and electronic security systems – Part 11-32: Electronic access control systems – Access control monitoring based on Web services

## Systèmes d'alarme et de sécurité électroniques – Partie 11-32: Systèmes de contrôle d'accès électronique – Commande de contrôle d'accès en fonction des services Web

IEC 60839-11-32:2016-11(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

IEC 60839-11-32

Edition 1.0 2016-11

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside



## Alarm and electronic security systems – Part 11-32: Electronic access control systems – Access control monitoring based on Web services

## Systèmes d'alarme et de sécurité électroniques – Partie 11-32: Systèmes de contrôle d'accès électronique – Commande de contrôle d'accès en fonction des services Web

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 13.320

ISBN 978-2-8322-3779-3

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

# ALARM AND ELECTRONIC SECURITY SYSTEMS –

# Part 11-32: Electronic access control systems – Access control monitoring based on Web services

## FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.



International Standard IEC 60839-11-32 has been prepared by IEC technical committee 79: Alarm and electronic security systems.

The text of this standard is based on the following documents:

| CDV | Report on voting |
|---|---|
| 79/523/CDV | 79/547/RVC |

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 60839 series, published under the general title *Alarm and electronic security systems*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

**IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.**

# INTRODUCTION

This document makes it possible to build an alarm and electronic security system with clients, typically a monitoring console, and devices, typically an access control unit, from different manufacturers using common and well defined interfaces.

This document specifies only the data and control flow between a client and the services without reference to any physical device as the services required to implement a compliant electronic access control system (EACS) are not necessarily implemented on a single device, i.e. all services can be run on a control panel, event aggregator software on PC, etc.

This document does not define internal communication between an access control unit and its components if they are implemented on a single device.

This document is based upon work done by the ONVIF open industry forum. The ONVIF Access Control specification and ONVIF Door Control specification are compatible with this document.

This document is accompanied by a set of computer readable interface definitions:

- Access control service WSDL, see Clause A.1;
- Door control service WSDL, see Clause A.2;
- Common schema, see Clause A.3;

Due to the differences in terminology used in IEC 60839-11-1, IEC 60839-11-2 and the ONVIF specification that this part of IEC 60839 is based on, a reader should take special notice of the terms and definitions clause.

Additional services needed for configuration of an EACS such as definitions of schedules, handling of access rules, readers and credentials are outside the scope of this document. These services will be covered by other parts of the IEC 60839-11-3x family of standards.

# ALARM AND ELECTRONIC SECURITY SYSTEMS –

# Part 11-32: Electronic access control systems – Access control monitoring based on Web services

## 1 Scope

This part of IEC 60839 defines the Web services interface for electronic access control systems. This includes listing electronic access control system components, their logical composition, monitoring their states and controlling them. It also includes a mapping of mandatory and optional requirements as per IEC 60839-11-1.

This document applies to physical security only. Physical security prevents unauthorized personnel, attackers or accidental intruders from physically accessing a building, room, etc.

Web services usage and device management functionality are outside of the scope of this document. Refer to IEC 60839-11-31 for more information.

This document does not in any way limit a manufacturer to add other protocols or extend the protocol defined here. For rules on how to accomplish this refer to IEC 60839-11-31.

## 2 Normative references

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.

IEC 60839-11-1, *Alarm and electronic security systems – Part 11-1: Electronic access control systems – System and components requirements*

IEC 60839-11-2, *Alarm and electronic security systems – Part 11-2: Electronic access control systems – Application guidelines*

## 3 Terms, definitions and abbreviated terms

### 3.1 Terms and definitions

For the purposes of this document the terms and definitions given in IEC 60839-11-1 and IEC 60839-11-2, as well as the following, apply.

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

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

NOTE When the IEC term defined in IEC 60839-11-1 and IEC 60839-11-2 differs from the terms used in this document the IEC term will be given in parentheses in the section headers.

**3.1.1**
**access point**
**portal**
physical entrance/exit at which access can be controlled by a door, turnstile or other secure barrier

Note 1 to entry: For the purposes of this document the access point is considered to be a logical composition of a physical door and reader(s) controlling access in one direction.

Note 2 to entry: In this document, the term "door" has the same meaning as "access point" or "portal".

**3.1.2**
**access point actuator**
**portal actuator**
part of an access control system that interfaces to an access control unit releasing and securing a portal according to pre-set rules

Note 1 to entry: In this document, the term “door lock” is used.

**3.1.3**
**access point mode**
logical operating mode of the portal indicating whether the portal is locked, unlocked, blocked, locked down or locked open, etc.

Note 1 to entry: In this document, the term “door mode” is used.

**3.1.4**
**access point sensor**
**portal sensor**
electrical component used to monitor the open or closed status of an access point, or locked/unlocked status of a locking device, or the secure/unsecure status of an electromagnetic lock or armature plate

Note 1 to entry: In this document, the term “door monitor” is used.

**3.1.5**
**access point overriding**
**portal overriding**
action of issuing a manual command to bypass the pre-configured mode of operation (i.e. release/secure/block) of an access point

Note 1 to entry: In this document, the terms “momentary access” and “unlocked” are examples of access point overriding.

**3.1.6**
**alarm**
<access control system> condition requiring human assessment or intervention

Note 1 to entry: Often used in electronic access control system in the sense of alert.

Note 2 to entry: In this document, the term "door alarm" is used.

**3.1.7**
**client**
service requester

EXAMPLE System management, annunciation, monitoring console.

**3.1.8**
**device**
service provider

EXAMPLE: Access control unit.

**3.1.9**
**portal side**
logical composition of a physical door and reader (s) controlling access in one direction

### 3.2 Abbreviated terms

For the purposes of this document, the abbreviated terms given in IEC 60839-11-1 and IEC 60839-11-2, as well as the following apply.

ACMS Access Control Management System
BMS Building Management System
HTTP Hypertext Transfer Protocol
PSIM Physical Security Information Management
SOAP Simple Object Access Protocol
TLS Transport Layer Security
WSDL Web Services Description Language

## 4 Overview

### 4.1 Interoperability

This document provides new interoperability opportunities by separating configuration from control and monitoring. In traditional systems, the central management system pushes all configurations data to devices on startup and expects that this configuration data is not changed by other clients. Instead, a client shall expect that all information is stored on end-devices and can be changed by others.

An EACS system defined by this document relies on service-oriented architecture principles. This allows installations where different components can be replaced or updated independently.

### 4.2 Event handling

Event handling is a crucial part of access control operations. In addition to real-time event delivery IEC 68839-11-31 provides the means for accessing stored events on the edge to deliver them if connection is lost.

Events are divided into 3 groups depending on their origin and purpose:

1) Configuration change events. These events are provided to achieve interoperability between several clients that control a single device simultaneously.
2) Transaction events. The core functionality of EACS that provides daily monitoring of all access events, including access granted events designed to notify clients about all detailed information (who, when and probably where they passed) on every particular access granted event, access denial events (that may or may not contain reason information), etc.
3) Alarms and faults events. These events provide health status monitoring allowing operators to take action in case of hardware failure, intrusion or other suspicious activity.

Refer to IEC 68839-11-31 for details on event delivery mechanism and 5.7 for the list of events defined by this document.

### 4.3 Architecture

This document does not mandate any specific physical device layout. The scheme provided in Figure 1 is not intended to be taken as a pattern but to serve as a reference for better

understanding of the given specification. Based on the definitions below, different physical configurations of an access controlled door are possible.

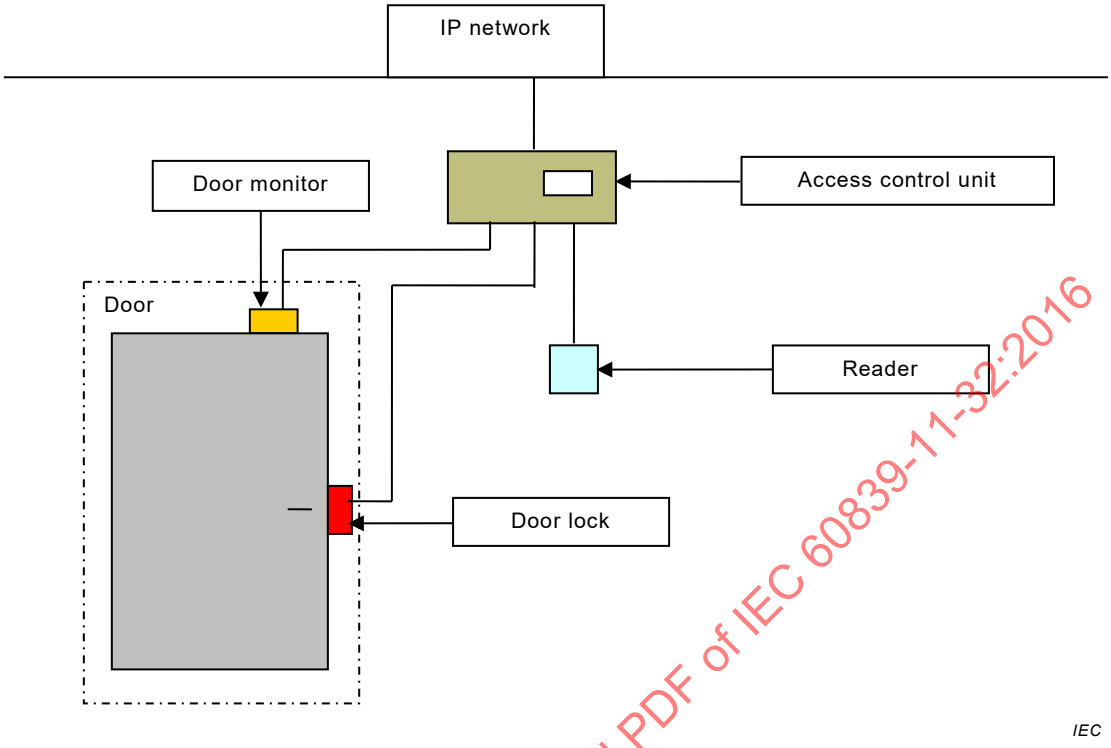


**Figure 1 – Schematic overview of an access controlled door**

A door that is controlled by an electronic access control system is equipped with the following devices:

- An access control unit that provides connections for reader, door sensor, door lock and additional digital inputs and outputs. This panel enables the software to interact with the physical devices. Sometimes these panels also contain storage and local intelligence to provide an offline functionality, so that the door will work as expected, even if there is no management system above available.
- A reader that is able to read a credential. In most cases a reader is only mounted at the outer (unsecure) side of the door. If the system monitors when somebody is leaving an area, a card reader will be mounted on both sides of the door.
- A door monitor that signals the control panel, that the door is open or closed.
- A door lock that can be engaged by the access control unit to release the door, for example in case an authorized credential is recognized.

The access control unit will, through the IP network, be connected to a system, typically a monitoring console, for monitoring and configuration.

### 4.4 External authorization (Overriding)

External authorization is a feature used to take access decisions for an access point outside the access control unit. External authorization entails, but is not limited to, a policy within the access control unit where the access control unit delegates the access decisions to an outside entity such as a guard or ACMS.

### 4.5 Security considerations

This document assumes possibility of building EACS systems interacting on the device level. This implies more security consideration than regular client-server interaction.

IEC 68839-11-31 defines several mechanisms to achieve this. They include, but are not limited to

- TLS for transport encryption;
- HTTP digest for client authentication;
- user management and access policies for client authorization;
- IEEE 802.1X certificate management for server authentication and spoofing protection.

Refer to the respective whitepapers and specifications for more information.

### 4.6 Door (access point) control

The door control service provides mechanisms for controlling physical door instances and monitoring their status.

The Door in this document can refer to such physical objects as an automatic barrier or a door equipped with an electric lock. Turnstiles which can restrict access in either direction can be represented with a pair of doors.

### 4.7 Design considerations

#### 4.7.1 Instance-level capabilities

A single EACS device may have diverse components of the same type. For example, a controller may operate two doors: one at the entrance to the building which has secure locking, monitoring and alarm abilities, and the other one is internal which can be only locked and unlocked.

Therefore, capabilities can be divided into two groups:

- overall service capabilities;
- capabilities for a particular entity in the service. It can also work in conjunction with the GetEventProperties function to provide finer control over the system.

Refer to 5.2 and 6.2 for more information.

#### 4.7.2 Retrieving status

This document defines two parallel mechanisms for retrieving status information for most entities:

- Get<*Entity*>State functions return a cumulative snapshot of the current state, operating mode and other run-time information.
- The Event service returns up-to-date and consistent states of entities. Each entity provides a set of events (usually one per each field in the State type) to notify a client about status changes. As far as these events are property events, a client receives the current state whenever a new subscription is initialized.

#### 4.7.3 Retrieving system configuration

This document defines several Get-functions that can return data incrementally. These functions allow the processing of a large number of entities even though resources are highly constrained.

To return data incrementally, these functions make use of a parameter called StartReference. StartReference is a device internal identifier used to continue fetching data from the last position, and allows a client to iterate over a large dataset in smaller chunks. The device handles a reasonable number of different StartReferences at the same time and they live for a reasonable time so that clients are able to fetch complete datasets.

A client always passes the value returned from a previous request to continue fetching data. Clients do not use the same reference more than once.

For example, the StartReference can be the incrementing start position number or the underlying database transaction identifier.

The returned NextStartReference is used as the StartReference parameter in successive calls, and may be changed by device in each call.

The following pseudo-code demonstrates how information about all access points can be obtained from a device:

```
StartRef = null
do {
     Response = GetAccessPointInfoList(StartReference = StartRef)
     if (Response.AccessPointInfo != null) {
          AllAccessPoints.Append(Response.AccessPointInfo)
     }
     StartRef = Response.NextStartReference
} while (StartRef != null)
```

# 5 Access control

## 5.1 General

This service offers commands to retrieve status information and to control access point instances.

## 5.2 Service capabilities

### 5.2.1 General

A device shall provide service capabilities in two ways:

1) With the GetServices method of Device service when IncludeCapability is true. Refer to IEC 68839-11-31 for more details.
2) With the GetServiceCapabilities method.

### 5.2.2 Data structures: ServiceCapabilities

The service capabilities reflect optional functionality of a service. The information is static and does not change during device operation. The following capabilities are available:

- **MaxLimit**

  The maximum number of entries returned by a single Get<Entity>List or Get<Entity>request. The device shall never return more than this number of entities in a single response.

### 5.2.3 GetServiceCapabilities command

This operation returns the capabilities of the access control service. A device shall support this command as described in Table 1.

**Table 1 – GetServiceCapabilities command**

| **GetServiceCapabilities** | Access class: PRE_AUTH |
|---|---|
| **Message name** | **Description** |
| GetServiceCapabilitiesRequest | *This message shall be empty* |
| GetServiceCapabilitiesResponse | *This message contains:*<br>• *"Capabilities": The capability response message contains the requested Access Control service capabilities using a hierarchical XML capability structure.*<br>tac:ServiceCapabilities **Capabilities [1][1]** |
| **Fault codes** | **Description** |
| | *No command specific faults* |

## 5.3 Access point (portal side) information

### 5.3.1 Data structures

#### 5.3.1.1 AccessPointInfo

The AccessPointInfo structure contains basic information about an access point instance. An access point defines an entity a credential can be granted or denied access to. The AccessPointInfo provides basic information on how access is controlled in one direction for a door (from which area to which area).

Multiple access points may cover the same door. A typical case is one access point for entry and another for exit, both referencing the same door.

A device shall provide the following fields for each access point instance:

- **Token**
  A service-unique identifier of the access point.
- **Name**
  A user readable name. It shall be up to 64 characters.
- **Entity**
  Reference to the entity used to control access; the entity type may be specified by the optional EntityType field explained below but is typically a door.
- **Capabilities**
  The capabilities for the access point.

To provide more information the device may include the following optional fields:

- **Description**
  Optional user readable description for the access point. It shall be up to 1 024 characters.
- **AreaFrom**
  Optional reference to the area from which access is requested.
- **AreaTo**
  Optional reference to the area to which access is requested.
- **EntityType**
  Optional entity type; if missing, a Door type as defined by the door control service should be assumed. This can also be represented by the QName value “tdc:Door” – where tdc is the namespace of the Door Control service: http://www.onvif.org/ver10/doorcontrol/wsdl. This field is provided for future extensions; it will allow an access point being extended to cover entity types other than doors as well.

#### 5.3.1.2 AccessPointCapabilities

The access point capabilities reflect optional functionality of a particular physical entity. Different access point instances may have different set of capabilities. This information may change during device operation, for example if hardware settings are changed. The following capabilities are available:

- **DisableAccessPoint**
  Indicates whether or not this access point instance supports EnableAccessPoint and DisableAccessPoint commands.
- **Duress**
  Indicates whether or not this access point instance supports the generation of duress events.
- **AnonymousAccess**
  Indicates whether or not this access point has a REX or other input that allows anonymous access.
- **AccessTaken**
  Indicates whether or not this access point instance supports the generation of AccessTaken and AccessNotTaken events.

  If AnonymousAccess and AccessTaken are both true, it indicates that the Anonymous versions of AccessTaken and AccessNotTaken are supported as well.
- **ExternalAuthorization**
  Indicates whether or not this AccessPoint instance supports the ExternalAuthorization operation and the generation of Request events.

  If AnonymousAccess and ExternalAuthorization are both true, it indicates that the Anonymous version is supported as well.

### 5.3.2 GetAccessPointInfoList command

This operation requests a list of all of AccessPointInfo items provided by the device. A device shall support this command as described in Table 2.

A call to this method shall return a StartReference when not all data is returned and more data is available. The reference shall be valid for retrieving the next set of data. Refer to 4.7.3 for more details.

The number of items returned shall not be greater than the Limit parameter.

**Table 2 – GetAccessPointInfoList command**

| **GetAccessPointInfoList** | Access class: READ_SYSTEM |
|---|---|
| **Message name** | **Description** |
| GetAccessPointInfoListRequest | *This message contains:*<br>• *"Limit": Maximum number of entries to return. If Limit is omitted or if the value of Limit is higher than what the device supports, then the device shall return its maximum amount of entries.*<br>• *"StartReference": Start returning entries from this start reference. If not specified, entries shall start from the beginning of the dataset.*<br>xs:int **Limit [0][1]**<br>xs:string**StartReference [0][1]** |
| GetAccessPointInfoListResponse | *This message contains:*<br>• *"NextStartReference": StartReference to use in next call to get the following items. If absent, no more items to get.*<br>• *"AccessPointInfo": List of AccessPointInfo items.*<br>xs:string **NextStartReference [0][1]**<br>tac:AccessPointInfo**AccessPointInfo [0][unbounded]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:InvalidStartReference | *StartReference is invalid or has timed out. Client needs to start fetching from the beginning.* |

### 5.3.3 GetAccessPointInfo command

This operation requests a list of AccessPointInfo items matching the given tokens. A device shall support this command as described in Table 3.

The device shall ignore tokens it cannot resolve and shall return an empty list if there are no items matching specified tokens. The device shall not return a fault in this case.

If the number of requested items is greater than MaxLimit, a TooManyItems fault shall be returned.

**Table 3 – GetAccessPointInfo command**

| **GetAccessPointInfo** | Access class: READ_SYSTEM |
|---|---|
| **Message name** | **Description** |
| GetAccessPointInfoRequest | *This message contains:*<br>• *"Token": Tokens of AccessPointInfo items to get.*<br>pt:ReferenceToken**Token [1][unbounded]** |
| GetAccessPointInfoResponse | *This message contains:*<br>• *"AccessPointInfo": List of AccessPointInfo items.*<br>tac:AccessPointInfo**AccessPointInfo [0][unbounded]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgs<br>ter:TooManyItems | *Too many items were requested, see MaxLimit capability.* |

## 5.4 Area information

### 5.4.1 Data structures: AreaInfo

The AreaInfo structure contains basic information about an area. A device shall provide the following fields for each area:

- **Token**
  A service-unique identifier of the area.
- **Name**
  User readable name. It shall be up to 64 characters.
  To provide more information, the device may include the following optional fields:
- **Description**
  User readable description for the area. It shall be up to 1 024 characters.

### 5.4.2 GetAreaInfoList command

This operation requests a list of all AreaInfo items provided by the device. A device shall support this command as described in Table 4.

A call to this method shall return a StartReference when not all data is returned and more data is available. The reference shall be valid for retrieving the next set of data. Refer to 4.7.3 for more details.

The number of items returned shall not be greater than the Limit parameter.

**Table 4 – GetAreaInfoList command**

| **GetAreaInfoList** | Access class: READ_SYSTEM |
|---|---|
| **Message name** | **Description** |
| GetAreaInfoListRequest | *This message contains:*<br>• *"Limit": Maximum number of entries to return. If Limit is omitted or if the value of Limit is higher than what the device supports, then the device shall return its maximum amount of entries.by the device.*<br>• *"StartReference": Start returning entries from this start reference. If not specified, entries shall start from the beginning of the dataset.*<br>xs:int **Limit [0][1]**<br>xs:string **StartReference [0][1]** |
| GetAreaInfoListResponse | *This message contains:*<br>• *"NextStartReference": StartReference to use in next call to get the following items. If absent, no more items to get.*<br>• *"AreaInfo": List of AreaInfo items.*<br>xs:string **NextStartReference [0][1]**<br>tac:AreaInfo**AreaInfo [0][unbounded]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:InvalidStartReference | *StartReference is invalid or has timed out.Client needs to start fetching from the beginning.* |

### 5.4.3 GetAreaInfo command

This operation requests a list of AreaInfo items matching the given tokens. A device shall support this command as described in Table 5.

The device shall ignore tokens it cannot resolve and may return an empty list if there are no items matching specified tokens.

If the number of requested items is greater than MaxLimit, a TooManyItems fault shall be returned.

**Table 5 – GetAreaInfo command**

| **GetAreaInfo** | Access class: READ_SYSTEM |
|---|---|
| **Message name** | **Description** |
| GetAreaInfoRequest | *This message contains:*<br>• *"Token": Tokens of AreaInfo items to get.*<br>pt:ReferenceToken**Token [1][unbounded]** |
| GetAreaInfoResponse | *This message contains:*<br>• *"AreaInfo": List of AreaInfo items.*<br>tac:AreaInfo**AreaInfo [0][unbounded]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgs<br>ter:TooManyItems | *Too many items were requested, see MaxLimit capability.* |

## 5.5 Access point (portal side) status

### 5.5.1 General

The state of the access point is determined by a number of operations that can be performed on it depending on its capabilities; refer to access point capabilities in 5.3.1.2.

### 5.5.2 Data structures: AccessPointState

The AccessPointState contains state information for an access point. A device shall provide the following fields for each access point instance:

- **Enabled**
  Indicates that the access point is enabled. By default this field value shall be true, if the DisableAccessPoint capability is not supported.

### 5.5.3 GetAccessPointState command

This operation requests the AccessPointState for the access point instance specified by the Token parameter. A device shall support this command as described in Table 6.

**Table 6 – GetAccessPointState command**

| **GetAccessPointState** | Access class: READ_SYSTEM_SENSITIVE |
|---|---|
| **Message name** | **Description** |
| GetAccessPointStateRequest | *This message contains:*<br>• *"Token": Token of the access point instance to get AccessPointState for.*<br>pt:ReferenceToken **Token [1][1]** |
| GetAccessPointStateResponse | *This message contains:*<br>• *"AccessPointState": AccessPointState item.*<br>tac:AccessPointState**AccessPointState [1][1]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *AccessPoint is not found* |

## 5.6 Access control commands

### 5.6.1 General

The service control commands contain operations that allow modifying access point states and controlling access points.

### 5.6.2 Data structures: Decision enumeration

The Decision enumeration represents a choice of two available options for an access request:

- **Granted**
  The decision is to grant access.
- **Denied**
  The decision is to deny access.

### 5.6.3 EnableAccessPoint command

This operation allows enabling an access point.

A device that signals support for DisableAccessPoint capability for a particular access point instance shall support this command as described in Table 7.

**Table 7 – EnableAccessPoint command**

| **EnableAccessPoint** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| EnableAccessPointRequest | *This message contains:*<br>• *"Token": Token of the access point instance to enable.*<br>pt:ReferenceToken **Token [1][1]** |
| EnableAccessPointResponse | *This message shall be empty* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:ActionNotSupported<br>ter:NotSupported | *The operation is not supported.* |

### 5.6.4 DisableAccessPoint command

This operation allows disabling an access point.

A device that signals support for DisableAccessPoint capability for a particular access point instance shall support this command as described in Table 8.

**Table 8 – DisableAccessPoint command**

| **DisableAccessPoint** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| DisableAccessPointRequest | *This message contains:*<br>• *"Token": Token of the access point instance to disable.*<br>pt:ReferenceToken **Token [1][1]** |
| DisableAccessPointResponse | *This message shall be empty* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:ActionNotSupported<br>ter:NotSupported | *The operation is not supported.* |

### 5.6.5 ExternalAuthorization command

This operation allows a Denied or Granted decision at an access point instance.

A device that signals support for ExternalAuthorization capability for a particular access point instance shall support this method as described in Table 9.

**Table 9 – ExternalAuthorization command**

| **ExternalAuthorization** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| ExternalAuthorizationRequest | *This message contains:*<br>• *"AccessPointToken": Token of the access point instance.*<br>• *"CredentialToken": Optional token of the Credential item involved.*<br>• *"Reason": Optional reason for decision.*<br>• *"Decision": Decision – Granted or Denied.*<br>pt:ReferenceToken **AccessPointToken [1][1]**<br>pt:ReferenceToken **CredentialToken [0][1]**<br>xs:string **Reason [0][1]**<br>tac:Decision **Decision [1][1]** |
| ExternalAuthorizationResponse | *This message shall be empty* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Access point is not found.* |
| env:Receiver<br>ter:ActionNotSupported<br>ter:NotSupported | *The operation is not supported.* |

## 5.7 Notification topics

### 5.7.1 Event overview

The AccessControl service specifies events to be used for access transactions, for example an access request is made and an access is granted or denied, when duress is detected, and when an important configuration has been changed.

The main topics for access transaction events are:

- tns1:AccessControl/AccessGranted/ – when access is granted.
- tns1:AccessControl/AccessTaken/ – when access is taken after being granted.
- tns1:AccessControl/AccessNotTaken/ – when access is not taken after being granted.
- tns1:AccessControl/Denied/ – when access is denied.
- tns1:AccessControl/Duress – when duress is detected.
- tns1:AccessControl/Request/ – when external authorization is requested or has timed out.

The main topic for status updates is:

- tns1:AccessPoint/State/ – for status updates.

The main topics for configuration change notifications are:

- tns1:Configuration/AccessPoint – when AccessPointConfiguration has been changed.
- tns1:Configuration/Area – when Area configuration has been changed.

NOTE The term “main topic” here means that a client can subscribe to, for example, tns1:AccessControl/AccessGranted, but the actual event sent (and thus received) can be the main topic itself or any subtopic of the main topic (such as tns1:AccessControl/AccessGranted/Credential). New subtopics may be defined in the future.

### 5.7.2 General transaction event layout

All transaction events use the following message scheme:

```
<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
      Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

where AccessPointToken identifies the AccessPoint where the decision was made.

The Data parameters are optional. See the definitions in 5.7.3 to 5.7.10 for their requirements. The parameter CredentialToken shall be present and CredentialHolderName may be present for credential based transactions. The optional parameter Reason holds error reason information and the optional parameter External signals whether the transaction was triggered as a result of the ExternalAuthorization command.

### 5.7.3 Access granted

#### 5.7.3.1 General

Whenever a positive access decision is made, i.e. access is granted, a device shall provide a corresponding event message as per 5.7.3.2 and 5.7.3.3.

The event shall be sent immediately once the decision is made regardless of whether access was taken or not.

#### 5.7.3.2 Anonymous

Whenever access is granted for an anonymous user at an access point, a device that signals AnonymousAccess capability for a particular access point instance shall provide the following event:

```
Topic: tns1:AccessControl/AccessGranted/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
</tt:Data>
</tt:MessageDescription>
```

If the command was triggered as a result of the ExternalAuthorization command, the data element External shall be set to true, otherwise the element is optional.

#### 5.7.3.3 Credential

Whenever a valid credential has passed all the necessary checks and a user or card holder is granted access to the access point, but not yet accessed it, entered or exited, the device shall provide the following event data:

```
Topic: tns1:AccessControl/AccessGranted/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

If the command was triggered as a result of the ExternalAuthorization command, the data element External shall be set to true, otherwise the element is optional.

The data element CredentialHolderName is optional.

### 5.7.4 Access taken

#### 5.7.4.1 General

A device that signals support for AccessTaken capability for a particular access point instance shall provide a corresponding event to notify client whenever an authorized person takes access.

#### 5.7.4.2 Anonymous

The device that signals support for AnonymousAccess capability for a particular access point instance shall provide the following event:

```
Topic: tns1:AccessControl/AccessTaken/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.4.3 Credential

When the device detects that access is taken and the credential can be identified, it shall provide the following event:

```
Topic: tns1:AccessControl/AccessTaken/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

The data element CredentialHolderName is optional.

### 5.7.5 Access not taken

#### 5.7.5.1 General

A device that signals support for AccessTaken capability for a particular AccessPoint instance shall provide a corresponding event to notify the client whenever a person was authorized but did not take access in time.

#### 5.7.5.2 Anonymous

The device that signals support for AnonymousAccess capability for a particular access point instance shall provide the following event:

```
Topic: tns1:AccessControl/AccessNotTaken/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
```

```
                                Type="pt:ReferenceToken"/>
</tt:Source>

</tt:MessageDescription>
```

#### 5.7.5.3 Credential

When the device detects that access is not taken and the credential can be identified, it shall provide the following event:

```
Topic: tns1:AccessControl/AccessNotTaken/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                                Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                                Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                                Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

The data element CredentialHolderName is optional.

### 5.7.6 Access denied

#### 5.7.6.1 General

A device shall provide one of the events as per 5.7.6.2, 5.7.6.3 and 5.7.6.4 whenever a person is denied to access.

The denial reason shall be present in the parameter Reason. This document defines the following strings that should be used by a device:

- **CredentialNotEnabled**
  The device shall provide the above reason whenever a valid credential is not enabled or has been disabled, for example due to credential being lost, to prevent unauthorized entry.
- **CredentialNotActive**
  The device shall provide the above reason, whenever a valid credential is presented though it is not active yet, for example the credential was presented before the start date.
- **CredentialExpired**
  The device shall provide the above reason whenever a valid credential was presented after its expiry date.
- **InvalidPIN**
  The device shall provide the above reason whenever an entered PIN code does not match the credential.
- **NotPermittedAtThisTime**
  The device shall provide the above reason whenever a valid credential is denied access to the requested AccessPoint because the credential is not permitted at the moment.
- **Unauthorized**
  The device shall provide the above reason whenever the presented credential is not authorized.
- **Other**
  The device shall provide the above reason whenever the request is denied and no other specific event matches it or is supported by the service.

More values may be defined by either future revisions of this document or as vendor specific extensions. To allow for this, a client shall treat unknown strings as “Other”.

While the Reason strings originally defined by this document do not use a QName style syntax, extensions to the Reason values will always use a QName style syntax. The prefix “pt” is reserved for use by ONVIF, i.e. "pt:<reason>". Vendor specific extensions should choose a suitable prefix.

Even if there are multiple reasons for denial, a device shall only send one reason and how the device chooses the reason in this case is outside the scope of this document.

#### 5.7.6.2 Anonymous

The device that signals support for AnonymousAccess capability for a particular access point instance should provide the following event when access is denied and credential information is not provided:

```
Topic: tns1:AccessControl/Denied/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

If the command was triggered as a result of the ExternalAuthorization command, the data element External shall be set to true, otherwise the element is optional.

#### 5.7.6.3 Credential

When the device denies access and the credential can be identified, it shall provide the following event:

```
Topic: tns1:AccessControl/Denied/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

If the command was triggered as a result of the ExternalAuthorization command, the data element External shall be set to true, otherwise the element is optional.

The data element CredentialHolderName is optional.

#### 5.7.6.4 CredentialNotFound

##### 5.7.6.4.1 General

Under some circumstances a device may be not able to resolve authentication data to a credential token. Whenever this happens the device should provide a corresponding event message as per 5.7.6.4.2.

Depending on which authentication factors were used message payload may differ. Currently only a single subset is defined.

##### 5.7.6.4.2 Card

Whenever there is no credential matching the request stored in the device, the device shall provide the following event:

```
Topic: tns1:AccessControl/Denied/CredentialNotFound/Card

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="Card" Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

The content of the Card string is vendor specific. It may contain the complete identification string of the card, part of this information or remain empty.

### 5.7.7 Duress

The device that signals support for Duress capability for a particular access point instance shall provide the following event whenever a condition of duress is detected.

```
Topic: tns1:AccessControl/Duress

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

The data parametersCredentialToken and CredentialHolderName are optional and may be omitted for anonymous access.

### 5.7.8 External authorization (Override)

#### 5.7.8.1 General

The device that signals support for ExternalAuthorization capability for a particular access point instance shall provide events defined in 5.7.8 whenever it requests for external

authorization. These notification messages shall be used in conjunction with corresponding Access Control service operations which provide feedback to device.

#### 5.7.8.2 Anonymous

Whenever a device that signals AnonymousAccess capability for a particular access point instance requests an external agent to authorize a person when credential information is not available, for example when a REX has been pressed and operator's confirmation is needed, it shall provide the following event:

```
Topic: tns1:AccessControl/Request/Anonymous
<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.8.3 Credential

Whenever the device requests an external agent to authorize a person, the device shall send the following event:

```
Topic: tns1:AccessControl/Request/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

CredentialHolderName is optional and may be omitted or an empty string can be used if it cannot be resolved.

#### 5.7.8.4 Timeout

A device shall provide the following event, whenever an external authorization request times out:

```
Topic: tns1:AccessControl/Request/Timeout

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.8.5 Example

A client that implements support for external authorization typically listens to the AccessControl/Request/<subtopic> events. When any of these events arrive, they are evaluated. If access is granted or denied, the ExternalAuthorization command is called on the device.

### 5.7.9 Status changes

#### 5.7.9.1 General

The device shall provide the status change events to inform subscribed clients when EACS entity status is changed. A device shall use the topics defined in 5.7.9 associated with the respective message description.

#### 5.7.9.2 Access point (portal side)

The device that signals support for DisableAccessPoint capability for a particular access point instance shall provide the following event whenever the state, enabled or disabled, of this access point is changed:

```
Topic: tns1:AccessPoint/State/Enabled

<tt:MessageDescriptionIsProperty="true">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
</tt:Data>
</tt:MessageDescription>
```

### 5.7.10 Configuration changes

#### 5.7.10.1 General

Whenever configuration data has been changed, added or been removed, a device shall provide these events to inform subscribed clients.

#### 5.7.10.2 Access point (portal side)

Whenever important configuration data for an access point is changed or an access point is added, the device shall provide the following event:

```
Topic: tns1:Configuration/AccessPoint/Changed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

Whenever an access point is removed, the device shall provide the following event:

```
Topic: tns1:Configuration/AccessPoint/Removed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.10.3 Area

Whenever configuration data for an area is changed or an area is added, the device shall provide the following event:

```
Topic: tns1:Configuration/Area/Changed
```

```
<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AreaToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

Whenever an area is removed, the device shall provide the following event:

```
Topic: tns1:Configuration/Area/Removed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AreaToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

# 6 Door (access point) control

## 6.1 General

This service offers commands to retrieve status information and to control door instances of a device.

## 6.2 Service capabilities

### 6.2.1 General

A device shall provide service capabilities in two ways:

1) With the GetServices method of Device service when IncludeCapability is true. Refer to refer to IEC 68839-11-31 for more details.
2) With the GetServiceCapabilities method.

### 6.2.2 Data structures: ServiceCapabilities

The ServiceCapabilities structure reflects optional functionality of a service. The information is static and does not change during device operation. The following capabilities are available:

- **MaxLimit**
  The maximum number of entries returned by a single Get<Entity>List or Get<Entity> request. The device shall never return more than this number of entities in a single response.

### 6.2.3 GetServiceCapabilities command

This operation returns the capabilities of the door control service. A device shall support this command as described in Table 10.

**Table 10 – GetServiceCapabilities command**

| **GetServiceCapabilities** | Access class: PRE_AUTH |
|---|---|
| **Message name** | **Description** |
| GetServiceCapabilitiesRequest | *This message shall be empty* |
| GetServiceCapabilitiesResponse | *This message contains:*<br>• *"Capabilities": The capability response message contains the requested Access Control service capabilities using a hierarchical XML capability structure.*<br>tac:ServiceCapabilities **Capabilities [1][1]** |
| **Fault codes** | **Description** |
| | *No command specific faults* |

## 6.3 Door (access point) information

### 6.3.1 Data structures

#### 6.3.1.1 DoorInfo

The DoorInfo type represents the door as a physical object. The structure contains information and capabilities of a specific door instance. A device shall provide the following fields for each door instance:

- **Token**
  A service-unique identifier of the door.
- **Name**
  A user readable name. It shall be up to 64 characters.
- **Capabilities**
  The capabilities of the door; is of type DoorCapabilities.

To provide more information, the device may include the following optional field:

- **Description**
  A user readable description. It shall be up to 1 024 characters.

#### 6.3.1.2 DoorCapabilities

DoorCapabilities reflect optional functionality of a particular physical entity. Different door instances may have different set of capabilities. This information may change during device operation, for example if hardware settings are changed. The following capabilities are available:

- **Access**
  Indicates whether or not this door instance supports the AccessDoor command to perform momentary access.
- **AccessTimingOverride**
  Indicates that this door instance supports overriding configured timing in the AccessDoor command.
- **Lock**
  Indicates that this door instance supports the LockDoor command to lock the door.
- **Unlock**
  Indicates that this door instance supports the UnlockDoor command to unlock the door.
- **Block**
  Indicates that this door instance supports the BlockDoor command to block the door.

- **DoubleLock**
  Indicates that this door instance supports the DoubleLockDoor command to lock multiple locks on the door.
- **LockDown**
  Indicates that this door instance supports LockDown (and LockDownRelease) commands to lock the door and put it in LockedDown mode.
- **LockOpen**
  Indicates that this door instance supports LockOpen (and LockOpenRelease) commands to unlock the door and put it in LockedOpen mode.
- **DoorMonitor**
  Indicates that this door instance has a DoorMonitor and supports the DoorPhysicalState event.
- **LockMonitor**
  Indicates that this door instance has a LockMonitor and supports the LockPhysicalState event.
- **DoubleLockMonitor**
  Indicates that this door instance has a DoubleLockMonitor and supports the DoubleLockPhysicalState event.
- **Alarm**
  Indicates that this door instance supports a door alarm and the DoorAlarm event.
- **Tamper**
  Indicates that this door instance has a tamper detector and supports the DoorTamper event.
- **Fault**
  Indicates that this door instance supports a door fault and the DoorFault event.

### 6.3.2 GetDoorInfoList command

This operation requests a list of all DoorInfo items provided by the device. A device shall support this method as described in Table 11.

A call to this method shall return a StartReference when not all data is returned and more data is available. The reference shall be valid for retrieving the next set of data. Refer to 4.7.3 for more details.

The number of items returned shall not be greater than the Limit parameter.

**Table 11 – GetDoorInfoList command**

| **GetDoorInfoList** | Access class: READ_SYSTEM |
|---|---|
| **Message name** | **Description** |
| GetDoorInfoListRequest | *This message contains:*<br>• *"Limit": Maximum number of entries to return. If Limit is omitted or if the value of Limit is higher than what the device supports, then the device shall return its maximum amount of entries.*<br>• *"StartReference": Start returning entries from this start reference. If not specified, entries shall start from the beginning of the dataset.*<br>xs:int **Limit [0][1]**<br>xs:string **StartReference [0][1]** |
| GetDoorInfoListResponse | *This message contains:*<br>• *"NextStartReference": StartReference to use in next call to get the following items. If absent, no more items to get.*<br>• *"DoorInfo": List of DoorInfo items.*<br>xs:string **NextStartReference [0][1]**<br>tdc:DoorInfo **DoorInfo [0][unbounded]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:InvalidStartReference | *StartReference is invalid or has timed out. Client needs to start fetching from the beginning* |

### 6.3.3 GetDoorInfo command

This operation requests a list of DoorInfo items matching the given tokens. A device that provides Door Control service shall support this method as described in Table 12.

The device shall ignore tokens it cannot resolve and may return an empty list if there are no Doors matching specified tokens.

If the number of requested items is greater than MaxLimit, a TooManyItems fault shall be returned.

**Table 12 – GetDoorInfo command**

| **GetDoorInfo** | Access class: READ_SYSTEM |
|---|---|
| **Message name** | **Description** |
| GetDoorInfoRequest | *This message contains:*<br>• *"Token": Tokens of DoorInfo items to get.*<br>pt:ReferenceToken **Token [1][unbounded]** |
| GetDoorInfoResponse | *This message contains:*<br>• *"DoorInfo": List of DoorInfo items.*<br>tdc:DoorInfo **DoorInfo [0][unbounded]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgs<br>ter:TooManyItems | *Too many items were requested, see MaxLimit capability.* |

## 6.4 Door (access point) status

### 6.4.1 General

The state of the door may be affected by a number of operations that can be performed on it depending on its capabilities: LockDoor, UnlockDoor, AccessDoor, BlockDoor, DoubleLockDoor, LockDownDoor, LockDownReleaseDoor, LockOpenDoor and LockOpenReleaseDoor.

### 6.4.2 Data structures

#### 6.4.2.1 DoorState

The DoorState structure contains current aggregate runtime status of door.

The following fields are available:

- **DoorPhysicalState**
  Physical state of the door; it is of type DoorPhysicalState. A device that signals support for DoorMonitor capability for a particular door instance shall provide this field.
- **LockPhysicalState**
  Physical state of the lock; it is of type LockPhysicalState. A device that signals support for LockMonitor capability for a particular door instance shall provide this field.
- **DoubleLockPhysicalState**
  Physical state of the double lock; it is of type LockPhysicalState. A device that signals support for DoubleLockMonitor capability for a particular door instance shall provide this field.
- **Alarm**
  Alarm state of the door; it is of type DoorAlarmState. A device that signals support for Alarm capability for a particular door instance shall provide this field.
- **Tamper**
  Tampering state of the door; it is of type DoorTamper. A device that signals support for Tamper capability for a particular door instance shall provide this field.
- **Fault**
  Fault information for door; it is of type DoorFault. A device that signals support for Fault capability for a particular door instance shall provide this field.
- **DoorMode**
  The logical operating mode of the door; it is of type DoorMode. A device shall report current operating mode in this field.

The following data types define states of DoorState elements.

#### 6.4.2.2 Enumeration: DoorPhysicalState

The physical state of a door. The following values are available:

- **Unknown**
  Value is currently unknown, possibly due to initialization or monitors not giving a conclusive result.
- **Open**
  Door is open.
- **Closed**
  Door is closed.
- **Fault**
  Door monitor fault is detected.

#### 6.4.2.3 Enumeration: LockPhysicalState

The physical state of a lock, including double lock. The following values are available:

- **Unknown**
  Value is currently not known.
- **Locked**
  Lock is activated.
- **Unlocked**
  Lock is not activated.
- **Fault**
  Lock fault is detected.

#### 6.4.2.4 Enumeration: DoorAlarmState

Describes the state of a door with regard to alarms. The following values are available:

- **Normal**
  No alarm.
- **DoorForcedOpen**
  Door is forced open.
- **DoorOpenTooLong**
  Door is held open too long.

#### 6.4.2.5 DoorTamper

Tampering information for a door. The following fields are available:

- **Reason**
  Optional field; details describing tampering state change, for example reason, place and time.

NOTE All fields, including this one, which are designed to give end-user prompts, can be localized to the customer's native language.

- **State**
  State of the tamper detector; it is of type DoorTamperState.

#### 6.4.2.6 Enumeration: DoorTamperState

Describes the state of a tamper detector. The following values are available:

- **Unknown**
  Value is currently not known.
- **NotInTamper**
  No tampering is detected.
- **TamperDetected**
  Tampering is detected.

#### 6.4.2.7 DoorFault

Fault information for a door. This can be extended with optional attributes in the future. The following fields are available:

- **Reason**
  Optional reason for fault.
- **State**
  Overall fault state for the door; it is of type DoorFaultState. If there are any faults, the value shall be: FaultDetected. Details of the detected fault shall be found in the Reason field, and/or the various DoorState fields and/or in extensions to this structure.

It can be extended with optional attributes in the future.

#### 6.4.2.8 Enumeration: DoorFaultState

Describes the state of a door fault. The following values are available:

- **Unknown**
  Fault state is unknown.
- **NotInFault**
  No fault is detected.
- **FaultDetected**
  Fault is detected.

#### 6.4.2.9 Enumeration: DoorMode

DoorMode parameters describe the mode of operation from a logical perspective. Setting a door mode reflects the intent to set a door in a physical state.

The following values are available:

- **Unknown**
  The mode of operation is unknown.
- **Locked**
  The intention is to set the door to a physical locked state. In this mode the device shall provide momentary access using the AccessDoor method if supported by the door instance.
- **Unlocked**
  The intention is to set the door to a physical unlocked state. Alarms related to door timing operations such as open too long or forced open are masked in this mode.
- **Accessed**
  The intention is to momentarily set the door to a physical unlocked state. After a predefined time the device shall revert the door to its previous mode. Alarms related to timing operations such as door forced open are masked in this mode.
- **Blocked**
  The intention is to set the door to a physical locked state and the device shall not allow AccessDoor requests, i.e. it is not possible for the door to go to the Accessed mode. All other requests to change the door mode are allowed.
- **LockedDown**
  The intention is to set the door to a physical locked state and the device shall only allow the LockDownReleaseDoor request. All other requests to change the door mode are not allowed.
- **LockedOpen**
  The intention is to set the door to a physical unlocked state and the device shall only allow the LockOpenReleaseDoor request. All other requests to change the door mode are not allowed.
- **DoubleLocked**
  The intention is to set the door with multiple locks to a physical double locked state. If the door does not support double locking the devices shall treat this as a normal locked mode. When changing to an unlocked mode from the double locked mode, the physical state of the door may first go to locked state before unlocking.

### 6.4.3 GetDoorState command

This operation requests the state of a door specified by the Token parameter. A device shall be capable of reporting the status of a door using a DoorState structure available from the GetDoorState command as described in Table 13.

**Table 13 – GetDoorState command**

| **GetDoorState** | Access class: READ_SYSTEM_SENSITIVE |
|---|---|
| **Message name** | **Description** |
| GetDoorStateRequest | *This message contains:*<br>• *"Token": Token of the door instance to get the state for.*<br>pt:ReferenceToken **Token [1][1]** |
| GetDoorStateResponse | *This message contains:*<br>• *"DoorState": The state of the door.*<br>tdc:DoorState **DoorState [1][1]** |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |

## 6.5 Door (access point) control commands

### 6.5.1 General

The service control commands contain operations that allow modifying the state of door instances and controlling door instances of a device as described in Table 14.

### 6.5.2 AccessDoor command

This operation allows momentarily access to a door. It is typically invoked when a card holder presents a card to a card reader at the door and is granted access.

The door mode shall change to the Accessed mode. Refer to 6.4.2.9 for more details.

The door shall remain accessible for the defined time. When the time span elapses, the door mode shall change back to its previous state.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

A device that signals support for Access capability for a particular door instance shall support this command. A device that signals support for AccessTimingOverride capability for a particular door instance shall also provide optional timing parameters (AccessTime, OpenTooLongTime and PreAlarmTime) when performing AccessDoor command.

The device shall take the best effort approach for parameters not supported; it shall fall back to preconfigured time or limit the time to the closest supported time if the specified time is out of range.

**Table 14 – AccessDoor command**

| **AccessDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| AccessDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>• *"UseExtendedTime": Optional – Indicates that the configured extended time should be used.*<br>• *"AccessTime": Optional – overrides AccessTime if specified.*<br>• *"OpenTooLongTime": Optional – overrides OpenTooLongTime if specified (DOTL).*<br>• *"PreAlarmTime": Optional – overrides PreAlarmTime if specified.*<br>• *"Extension": Future extension.*<br><br>pt:ReferenceToken **Token [1][1]**<br>xs:boolean **UseExtendedTime [0][1]**<br>xs:duration **AccessTime [0][1]**<br>xs:duration **OpenTooLongTime [0][1]**<br>xs:duration **PreAlarmTime [0][1]**<br>tdc:AccessDoorExtension **Extension [0][1]** |
| AccessDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to Accessed state and unlock the door.* |

### 6.5.3 LockDoor command

This operation allows locking a door as described in Table 15. The door mode shall change to the Locked mode. Refer to 6.4.2.9 for more details.

A device that signals support for the Lock capability for a particular door instance shall support this method.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 15 – LockDoor command**

| **LockDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| LockDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br><br>pt:ReferenceToken **Token [1][1]** |
| LockDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to Locked state.* |

### 6.5.4 UnlockDoor command

This operation allows unlocking a door. The door mode shall change to the Unlocked mode. Refer to 6.4.2.9 for more details.

A device that signals support for Unlock capability for a particular door instance shall support this method as described in Table 16.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 16 – UnlockDoor command**

| **UnlockDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| UnlockDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| UnlockDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to Unlocked state.* |

### 6.5.5 BlockDoor command

This operation allows blocking a door and preventing momentary access (AccessDoor command). The door mode shall change to the Blocked mode. Refer to 6.4.2.9 for more details.

A device that signals support for Block capability for a particular door instance shall support this method as described in Table 17.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 17 – BlockDoor command**

| **BlockDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| BlockDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| BlockDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to Blocked state.* |

### 6.5.6 LockDownDoor command

This operation allows locking and preventing other actions until a LockDownRelease command is invoked. The door mode shall change to the LockedDown mode. Refer to 6.4.2.9 for more details.

The device shall ignore other door control commands until a LockDownRelease command is performed.

A device that signals support for LockDown capability for a particular door instance shall support this command as described in Table 18.

If a device supports DoubleLock capability for a particular door instance, that operation may be engaged as well.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 18 – LockDownDoor command**

| **LockDownDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| LockDownDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| LockDownDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to a LockedDown state.* |

### 6.5.7 LockDownReleaseDoor command

This operation allows releasing the LockedDown mode of a door. The door mode shall change back to its previous/next mode. It is not defined what the previous/next mode shall be, but typically it will be the Locked mode.

A device that signals support for LockDown capability for a particular door instance shall support this command as described in Table 19.

This method shall only succeed if the current door mode is LockedDown.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 19 – LockDownReleaseDoor command**

| **LockDownReleaseDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| LockDownReleaseDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| LockDownReleaseDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to leave LockedDown state.* |

### 6.5.8 LockOpenDoor command

This operation allows unlocking a door and preventing other actions until LockOpenRelease method is invoked. The door mode shall change to the LockedOpen mode. Refer to 6.4.2.9 for more details.

The device shall ignore other door control commands until a LockOpenRelease command is performed.

A device that signals support for LockOpen capability for a particular door instance shall support this command as described in Table 20.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 20 – LockOpenDoor command**

| **LockOpenDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| LockOpenDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| LockOpenDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to LockedOpen state.* |

### 6.5.9 LockOpenReleaseDoor command

This operation allows releasing the LockedOpen mode of a door. The DoorMode shall change from the LockedOpen mode back to its previous/next mode. It is not defined what the previous/next state mode shall be, but typically it will be the Unlocked mode.

A device that signals support for LockOpen capability for a particular door instance shall support this command as described in Table 21.

This method shall only succeed if the current door mode is LockedOpen.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 21 – LockOpenReleaseDoor command**

| **LockOpenReleaseDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| LockOpenReleaseDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| LockOpenReleaseDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to leave LockedOpen state.* |

### 6.5.10 DoubleLockDoor command

This operation is used for securely locking a door. A call to this method shall change the door mode to DoubleLocked mode. Refer to 6.4.2.9 for more details.

A device that signals support for DoubleLock capability for a particular door instance shall support this command as described in Table 22. Otherwise this method can be performed as a standard Lockdoor operation, see 6.5.3.

If the door has an extra lock that shall be locked as well.

If setting the mode of operation fails, a Failure fault shall be returned. Note that failure in setting the door's physical state will be signalled with events.

**Table 22 – DoubleLockDoor command**

| **DoubleLockDoor** | Access class: ACTUATE |
|---|---|
| **Message name** | **Description** |
| DoubleLockDoorRequest | *This message contains:*<br>• *"Token": Token of the door instance to control.*<br>pt:ReferenceToken **Token [1][1]** |
| DoubleLockDoorResponse | *This message is typically empty, but is extendable* |
| **Fault codes** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *The specified token is not found.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Failed to go to DoubleLocked state.* |

## 6.6 Notification Topics

### 6.6.1 General

Subclause 6.6 defines notification topics specific to Door Control service.

### 6.6.2 Status changes

Whenever a door mode is changed, the device shall provide the following event:

```
Topic: tns1:Door/State/DoorMode

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorMode"/>
  </tt:Data>
</tt:MessageDescription>
```

A device that signals support for DoorMonitor capability for a particular door instance shall provide the following event whenever the physical state of this door is changed:

```
Topic: tns1:Door/State/DoorPhysicalState

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorPhysicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

A device that signals support for LockMonitor capability for a particular door instance shall provide the following event whenever the physical state of this door's lock is changed:

```
Topic: tns1:Door/State/LockPhysicalState

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:LockPhysicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

A device that signals support for DoubleLockMonitor capability for a particular door instance shall provide the following event whenever the physical state of this door's secure lock is changed:

```
Topic: tns1:Door/State/DoubleLockPhysicalState

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:LockPhysicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

A device that signals support for Alarm capability for a particular door instance shall provide the following event whenever the alarm state of this door is changed:

```
Topic: tns1:Door/State/DoorAlarm

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorAlarmState"/>
  </tt:Data>
</tt:MessageDescription>
```

A device that signals support for Tamper capability for a particular door instance shall provide the following event whenever the tamper state of this door is changed:

```
Topic: tns1:Door/State/DoorTamper

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorTamperState"/>
  </tt:Data>
</tt:MessageDescription>
```

A device that signals support for Fault capability for a particular door instance shall provide the following event whenever the fault state of this door is changed:

```
Topic: tns1:Door/State/DoorFault

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorFaultState"/>
    <tt:SimpleItemDescription Name="Reason" Type="xs:string"/>
  </tt:Data>
</tt:MessageDescription>
```

The Reason element may be empty or absent. The device may also skip it unless the fault state is FaultDetected.

### 6.6.3 Configuration changes

Whenever configuration data for a door is changed or a door is added, the device shall provide the following event:

```
Topic: tns1:Configuration/Door/Changed

<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>

</tt:MessageDescription>
```

Whenever a door is removed, the device shall provide the following event:

```
Topic: tns1:Configuration/Door/Removed

<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>

</tt:MessageDescription>
```

# Annex A
(normative)

# Access control interface XML schemata

## A.1 Access control service WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tac="http://www.onvif.org/ver10/accesscontrol/wsdl" name="PACSService"
targetNamespace="http://www.onvif.org/ver10/accesscontrol/wsdl">
  <wsdl:types>
    <xs:schema xmlns:pt="http://www.onvif.org/ver10/pacs"
targetNamespace="http://www.onvif.org/ver10/accesscontrol/wsdl" elementFormDefault="qualified"
version="1.0">
      <xs:import namespace="http://www.onvif.org/ver10/pacs" schemaLocation="types.xsd"/>
      <!--====== types ======-->
      <xs:complexType name="ServiceCapabilities">
        <xs:sequence>
          <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
        </xs:sequence>
        <xs:attribute name="MaxLimit" type="xs:unsignedInt" use="required"/>
        <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <!--================================-->
      <xs:complexType name="AccessPointInfoBase">
        <xs:complexContent>
          <xs:extension base="pt:DataEntity">
            <xs:sequence>
              <xs:element name="Name" type="pt:Name"/>
              <xs:element name="Description" type="pt:Description" minOccurs="0"/>
              <xs:element name="AreaFrom" type="pt:ReferenceToken" minOccurs="0"/>
              <xs:element name="AreaTo" type="pt:ReferenceToken" minOccurs="0"/>
              <xs:element name="EntityType" type="xs:QName" minOccurs="0"/>
              <xs:element name="Entity" type="pt:ReferenceToken"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <!--================================-->
      <xs:complexType name="AccessPointInfo">
        <xs:complexContent>
          <xs:extension base="tac:AccessPointInfoBase">
            <xs:sequence>
              <xs:element name="Capabilities" type="tac:AccessPointCapabilities"/>
              <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <!--================================-->
      <xs:complexType name="AccessPointCapabilities">
        <xs:sequence>
          <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
```

```
    </xs:sequence>
    <xs:attribute name="DisableAccessPoint" type="xs:boolean" use="required"/>
    <xs:attribute name="Duress" type="xs:boolean"/>
    <xs:attribute name="AnonymousAccess" type="xs:boolean"/>
    <xs:attribute name="AccessTaken" type="xs:boolean"/>
    <xs:attribute name="ExternalAuthorization" type="xs:boolean"/>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="AreaInfoBase">
    <xs:complexContent>
      <xs:extension base="pt:DataEntity">
        <xs:sequence>
          <xs:element name="Name" type="pt:Name"/>
          <xs:element name="Description" type="pt:Description" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="AreaInfo">
    <xs:complexContent>
      <xs:extension base="tac:AreaInfoBase">
        <xs:sequence/>
        <xs:anyAttribute processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="AccessPointState">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean"/>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="Decision">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Granted"/>
      <xs:enumeration value="Denied"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="DenyReason">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CredentialNotEnabled"/>
      <xs:enumeration value="CredentialNotActive"/>
      <xs:enumeration value="CredentialExpired"/>
      <xs:enumeration value="InvalidPIN"/>
      <xs:enumeration value="NotPermittedAtThisTime"/>
      <xs:enumeration value="Unauthorized"/>
      <xs:enumeration value="Other"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <!--  Message Request / Response elements  -->
  <xs:element name="GetServiceCapabilities">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
```

```
    </xs:element>
    <!--================================-->
    <xs:element name="GetServiceCapabilitiesResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Capabilities" type="tac:ServiceCapabilities"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAccessPointInfoList">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Limit" type="xs:int" minOccurs="0"/>
          <xs:element name="StartReference" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAccessPointInfoListResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NextStartReference" type="xs:string" minOccurs="0"/>
          <xs:element name="AccessPointInfo" type="tac:AccessPointInfo" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAccessPointInfo">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Token" type="pt:ReferenceToken" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAccessPointInfoResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AccessPointInfo" type="tac:AccessPointInfo" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAreaInfoList">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Limit" type="xs:int" minOccurs="0"/>
          <xs:element name="StartReference" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAreaInfoListResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NextStartReference" type="xs:string" minOccurs="0"/>
          <xs:element name="AreaInfo" type="tac:AreaInfo" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
```

```
</xs:element>
<!--================================-->
<xs:element name="GetAreaInfo">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Token" type="pt:ReferenceToken" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetAreaInfoResponse">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="AreaInfo" type="tac:AreaInfo" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetAccessPointState">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Token" type="pt:ReferenceToken"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetAccessPointStateResponse">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="AccessPointState" type="tac:AccessPointState"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="EnableAccessPoint">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Token" type="pt:ReferenceToken"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="EnableAccessPointResponse">
 <xs:complexType>
  <xs:sequence/>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="DisableAccessPoint">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Token" type="pt:ReferenceToken"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="DisableAccessPointResponse">
 <xs:complexType>
  <xs:sequence/>
 </xs:complexType>
</xs:element>
<!--================================-->
```

```
    <xs:element name="ExternalAuthorization">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AccessPointToken" type="pt:ReferenceToken"/>
          <xs:element name="CredentialToken" type="pt:ReferenceToken" minOccurs="0"/>
          <xs:element name="Reason" type="xs:string" minOccurs="0"/>
          <xs:element name="Decision" type="tac:Decision"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="ExternalAuthorizationResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
  </xs:schema>
</wsdl:types>
<wsdl:message name="GetServiceCapabilitiesRequest">
  <wsdl:part name="parameters" element="tac:GetServiceCapabilities"/>
</wsdl:message>
<wsdl:message name="GetServiceCapabilitiesResponse">
  <wsdl:part name="parameters" element="tac:GetServiceCapabilitiesResponse"/>
</wsdl:message>
<wsdl:message name="GetAccessPointInfoListRequest">
  <wsdl:part name="parameters" element="tac:GetAccessPointInfoList"/>
</wsdl:message>
<wsdl:message name="GetAccessPointInfoListResponse">
  <wsdl:part name="parameters" element="tac:GetAccessPointInfoListResponse"/>
</wsdl:message>
<wsdl:message name="GetAccessPointInfoRequest">
  <wsdl:part name="parameters" element="tac:GetAccessPointInfo"/>
</wsdl:message>
<wsdl:message name="GetAccessPointInfoResponse">
  <wsdl:part name="parameters" element="tac:GetAccessPointInfoResponse"/>
</wsdl:message>
<wsdl:message name="GetAreaInfoListRequest">
  <wsdl:part name="parameters" element="tac:GetAreaInfoList"/>
</wsdl:message>
<wsdl:message name="GetAreaInfoListResponse">
  <wsdl:part name="parameters" element="tac:GetAreaInfoListResponse"/>
</wsdl:message>
<wsdl:message name="GetAreaInfoRequest">
  <wsdl:part name="parameters" element="tac:GetAreaInfo"/>
</wsdl:message>
<wsdl:message name="GetAreaInfoResponse">
  <wsdl:part name="parameters" element="tac:GetAreaInfoResponse"/>
</wsdl:message>
<wsdl:message name="GetAccessPointStateRequest">
  <wsdl:part name="parameters" element="tac:GetAccessPointState"/>
</wsdl:message>
<wsdl:message name="GetAccessPointStateResponse">
  <wsdl:part name="parameters" element="tac:GetAccessPointStateResponse"/>
</wsdl:message>
<wsdl:message name="EnableAccessPointRequest">
  <wsdl:part name="parameters" element="tac:EnableAccessPoint"/>
</wsdl:message>
<wsdl:message name="EnableAccessPointResponse">
  <wsdl:part name="parameters" element="tac:EnableAccessPointResponse"/>
</wsdl:message>
<wsdl:message name="DisableAccessPointRequest">
```

```
  <wsdl:part name="parameters" element="tac:DisableAccessPoint"/>
 </wsdl:message>
 <wsdl:message name="DisableAccessPointResponse">
  <wsdl:part name="parameters" element="tac:DisableAccessPointResponse"/>
 </wsdl:message>
 <wsdl:message name="ExternalAuthorizationRequest">
  <wsdl:part name="parameters" element="tac:ExternalAuthorization"/>
 </wsdl:message>
 <wsdl:message name="ExternalAuthorizationResponse">
  <wsdl:part name="parameters" element="tac:ExternalAuthorizationResponse"/>
 </wsdl:message>
 <wsdl:portType name="PACSPort">
  <wsdl:operation name="GetServiceCapabilities">
   <wsdl:input message="tac:GetServiceCapabilitiesRequest"/>
   <wsdl:output message="tac:GetServiceCapabilitiesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetAccessPointInfoList">
   <wsdl:input message="tac:GetAccessPointInfoListRequest"/>
   <wsdl:output message="tac:GetAccessPointInfoListResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetAccessPointInfo">
   <wsdl:input message="tac:GetAccessPointInfoRequest"/>
   <wsdl:output message="tac:GetAccessPointInfoResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetAreaInfoList">
   <wsdl:input message="tac:GetAreaInfoListRequest"/>
   <wsdl:output message="tac:GetAreaInfoListResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetAreaInfo">
   <wsdl:input message="tac:GetAreaInfoRequest"/>
   <wsdl:output message="tac:GetAreaInfoResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetAccessPointState">
   <wsdl:input message="tac:GetAccessPointStateRequest"/>
   <wsdl:output message="tac:GetAccessPointStateResponse"/>
  </wsdl:operation>
  <wsdl:operation name="EnableAccessPoint">
   <wsdl:input message="tac:EnableAccessPointRequest"/>
   <wsdl:output message="tac:EnableAccessPointResponse"/>
  </wsdl:operation>
  <wsdl:operation name="DisableAccessPoint">
   <wsdl:input message="tac:DisableAccessPointRequest"/>
   <wsdl:output message="tac:DisableAccessPointResponse"/>
  </wsdl:operation>
  <wsdl:operation name="ExternalAuthorization">
   <wsdl:input message="tac:ExternalAuthorizationRequest"/>
   <wsdl:output message="tac:ExternalAuthorizationResponse"/>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="PACSBinding" type="tac:PACSPort">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <!--================================-->
  <wsdl:operation name="GetServiceCapabilities">
   <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/GetServiceCapabilities"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
```

```
    <!--================================-->
    <wsdl:operation name="GetAccessPointInfoList">
      <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/GetAccessPointInfoList"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <!--================================-->
    <wsdl:operation name="GetAccessPointInfo">
      <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/GetAccessPointInfo"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <!--================================-->
    <wsdl:operation name="GetAreaInfoList">
      <soap:operation soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/GetAreaInfoList"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <!--================================-->
    <wsdl:operation name="GetAreaInfo">
      <soap:operation soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/GetAreaInfo"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <!--================================-->
    <wsdl:operation name="GetAccessPointState">
      <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/GetAccessPointState"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <!--================================-->
    <wsdl:operation name="EnableAccessPoint">
      <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/EnableAccessPoint"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
```

```
    </wsdl:output>
   </wsdl:operation>
   <!--================================-->
   <wsdl:operation name="DisableAccessPoint">
    <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/DisableAccessPoint"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <!--================================-->
   <wsdl:operation name="ExternalAuthorization">
    <soap:operation
soapAction="http://www.onvif.org/ver10/accesscontrol/wsdl/ExternalAuthorization"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <!--================================-->
  </wsdl:binding>
</wsdl:definitions>
```

## A.2 Door control service WSDL

The wsdl:definitions provided at the commencement of the following XML schema should be confirmed at the time of implementation to ensure correct operation

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tdc="http://www.onvif.org/ver10/doorcontrol/wsdl" name="DoorControlService"
targetNamespace="http://www.onvif.org/ver10/doorcontrol/wsdl">
  <wsdl:types>
   <xs:schema xmlns:pt="http://www.onvif.org/ver10/pacs"
targetNamespace="http://www.onvif.org/ver10/doorcontrol/wsdl" elementFormDefault="qualified"
version="1.0">
    <xs:import namespace="http://www.onvif.org/ver10/pacs" schemaLocation="types.xsd"/>
    <!--====== types ======-->
    <xs:complexType name="ServiceCapabilities">
     <xs:sequence>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
     </xs:sequence>
     <xs:attribute name="MaxLimit" type="xs:unsignedInt" use="required"/>
     <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <!--================================-->
    <xs:complexType name="DoorInfoBase">
     <xs:complexContent>
      <xs:extension base="pt:DataEntity">
       <xs:sequence>
        <xs:element name="Name" type="pt:Name"/>
```

```
          <xs:element name="Description" type="pt:Description" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DoorInfo">
    <xs:complexContent>
      <xs:extension base="tdc:DoorInfoBase">
        <xs:sequence>
          <xs:element name="Capabilities" type="tdc:DoorCapabilities"/>
          <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DoorCapabilities">
    <xs:sequence>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
    <xs:attribute name="Access" type="xs:boolean"/>
    <xs:attribute name="AccessTimingOverride" type="xs:boolean"/>
    <xs:attribute name="Lock" type="xs:boolean"/>
    <xs:attribute name="Unlock" type="xs:boolean"/>
    <xs:attribute name="Block" type="xs:boolean"/>
    <xs:attribute name="DoubleLock" type="xs:boolean"/>
    <xs:attribute name="LockDown" type="xs:boolean"/>
    <xs:attribute name="LockOpen" type="xs:boolean"/>
    <xs:attribute name="DoorMonitor" type="xs:boolean"/>
    <xs:attribute name="LockMonitor" type="xs:boolean"/>
    <xs:attribute name="DoubleLockMonitor" type="xs:boolean"/>
    <xs:attribute name="Alarm" type="xs:boolean"/>
    <xs:attribute name="Tamper" type="xs:boolean"/>
    <xs:attribute name="Fault" type="xs:boolean"/>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DoorState">
    <xs:sequence>
      <xs:element name="DoorPhysicalState" type="tdc:DoorPhysicalState" minOccurs="0"/>
      <xs:element name="LockPhysicalState" type="tdc:LockPhysicalState" minOccurs="0"/>
      <xs:element name="DoubleLockPhysicalState" type="tdc:LockPhysicalState" minOccurs="0"/>
      <xs:element name="Alarm" type="tdc:DoorAlarmState" minOccurs="0"/>
      <xs:element name="Tamper" type="tdc:DoorTamper" minOccurs="0"/>
      <xs:element name="Fault" type="tdc:DoorFault" minOccurs="0"/>
      <xs:element name="DoorMode" type="tdc:DoorMode"/>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="DoorPhysicalState">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Unknown"/>
      <xs:enumeration value="Open"/>
      <xs:enumeration value="Closed"/>
      <xs:enumeration value="Fault"/>
```

```
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="LockPhysicalState">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Unknown"/>
      <xs:enumeration value="Locked"/>
      <xs:enumeration value="Unlocked"/>
      <xs:enumeration value="Fault"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="DoorAlarmState">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Normal"/>
      <xs:enumeration value="DoorForcedOpen"/>
      <xs:enumeration value="DoorOpenTooLong"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="DoorTamper">
    <xs:sequence>
      <xs:element name="Reason" type="xs:string" minOccurs="0"/>
      <xs:element name="State" type="tdc:DoorTamperState"/>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="DoorTamperState">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Unknown"/>
      <xs:enumeration value="NotInTamper"/>
      <xs:enumeration value="TamperDetected"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="DoorFault">
    <xs:sequence>
      <xs:element name="Reason" type="xs:string" minOccurs="0"/>
      <xs:element name="State" type="tdc:DoorFaultState"/>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="DoorFaultState">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Unknown"/>
      <xs:enumeration value="NotInFault"/>
      <xs:enumeration value="FaultDetected"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="DoorMode">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Unknown"/>
      <xs:enumeration value="Locked"/>
      <xs:enumeration value="Unlocked"/>
      <xs:enumeration value="Accessed"/>
```

```
        <xs:enumeration value="Blocked"/>
        <xs:enumeration value="LockedDown"/>
        <xs:enumeration value="LockedOpen"/>
        <xs:enumeration value="DoubleLocked"/>
      </xs:restriction>
    </xs:simpleType>
    <!--================================-->
    <xs:complexType name="AccessDoorExtension">
      <xs:sequence>
        <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
      </xs:sequence>
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <!--================================-->
    <!--  Message Request / Response elements  -->
    <xs:element name="GetServiceCapabilities">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetServiceCapabilitiesResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Capabilities" type="tdc:ServiceCapabilities"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetDoorInfoList">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Limit" type="xs:int" minOccurs="0"/>
          <xs:element name="StartReference" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetDoorInfoListResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NextStartReference" type="xs:string" minOccurs="0"/>
          <xs:element name="DoorInfo" type="tdc:DoorInfo" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetDoorInfo">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Token" type="pt:ReferenceToken" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetDoorInfoResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DoorInfo" type="tdc:DoorInfo" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
```

```
</xs:element>
<!--================================-->
<xs:element name="GetDoorState">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Token" type="pt:ReferenceToken"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetDoorStateResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="DoorState" type="tdc:DoorState"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="AccessDoor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Token" type="pt:ReferenceToken"/>
      <xs:element name="UseExtendedTime" type="xs:boolean" minOccurs="0"/>
      <xs:element name="AccessTime" type="xs:duration" minOccurs="0"/>
      <xs:element name="OpenTooLongTime" type="xs:duration" minOccurs="0"/>
      <xs:element name="PreAlarmTime" type="xs:duration" minOccurs="0"/>
      <xs:element name="Extension" type="tdc:AccessDoorExtension" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="AccessDoorResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="LockDoor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Token" type="pt:ReferenceToken"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="LockDoorResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="UnlockDoor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Token" type="pt:ReferenceToken"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="UnlockDoorResponse">
  <xs:complexType>
    <xs:sequence/>
```

```
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="BlockDoor">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Token" type="pt:ReferenceToken"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="BlockDoorResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockDownDoor">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Token" type="pt:ReferenceToken"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockDownDoorResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockDownReleaseDoor">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Token" type="pt:ReferenceToken"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockDownReleaseDoorResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockOpenDoor">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Token" type="pt:ReferenceToken"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockOpenDoorResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="LockOpenReleaseDoor">
  <xs:complexType>
   <xs:sequence>
```

```
        <xs:element name="Token" type="pt:ReferenceToken"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="LockOpenReleaseDoorResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="DoubleLockDoor">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Token" type="pt:ReferenceToken"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="DoubleLockDoorResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
 </xs:schema>
</wsdl:types>
<wsdl:message name="GetServiceCapabilitiesRequest">
 <wsdl:part name="parameters" element="tdc:GetServiceCapabilities"/>
</wsdl:message>
<wsdl:message name="GetServiceCapabilitiesResponse">
 <wsdl:part name="parameters" element="tdc:GetServiceCapabilitiesResponse"/>
</wsdl:message>
<wsdl:message name="GetDoorInfoListRequest">
 <wsdl:part name="parameters" element="tdc:GetDoorInfoList"/>
</wsdl:message>
<wsdl:message name="GetDoorInfoListResponse">
 <wsdl:part name="parameters" element="tdc:GetDoorInfoListResponse"/>
</wsdl:message>
<wsdl:message name="GetDoorInfoRequest">
 <wsdl:part name="parameters" element="tdc:GetDoorInfo"/>
</wsdl:message>
<wsdl:message name="GetDoorInfoResponse">
 <wsdl:part name="parameters" element="tdc:GetDoorInfoResponse"/>
</wsdl:message>
<wsdl:message name="GetDoorStateRequest">
 <wsdl:part name="parameters" element="tdc:GetDoorState"/>
</wsdl:message>
<wsdl:message name="GetDoorStateResponse">
 <wsdl:part name="parameters" element="tdc:GetDoorStateResponse"/>
</wsdl:message>
<wsdl:message name="AccessDoorRequest">
 <wsdl:part name="parameters" element="tdc:AccessDoor"/>
</wsdl:message>
<wsdl:message name="AccessDoorResponse">
 <wsdl:part name="parameters" element="tdc:AccessDoorResponse"/>
</wsdl:message>
<wsdl:message name="LockDoorRequest">
 <wsdl:part name="parameters" element="tdc:LockDoor"/>
</wsdl:message>
<wsdl:message name="LockDoorResponse">
 <wsdl:part name="parameters" element="tdc:LockDoorResponse"/>
```

```
</wsdl:message>
<wsdl:message name="UnlockDoorRequest">
  <wsdl:part name="parameters" element="tdc:UnlockDoor"/>
</wsdl:message>
<wsdl:message name="UnlockDoorResponse">
  <wsdl:part name="parameters" element="tdc:UnlockDoorResponse"/>
</wsdl:message>
<wsdl:message name="BlockDoorRequest">
  <wsdl:part name="parameters" element="tdc:BlockDoor"/>
</wsdl:message>
<wsdl:message name="BlockDoorResponse">
  <wsdl:part name="parameters" element="tdc:BlockDoorResponse"/>
</wsdl:message>
<wsdl:message name="LockDownDoorRequest">
  <wsdl:part name="parameters" element="tdc:LockDownDoor"/>
</wsdl:message>
<wsdl:message name="LockDownDoorResponse">
  <wsdl:part name="parameters" element="tdc:LockDownDoorResponse"/>
</wsdl:message>
<wsdl:message name="LockDownReleaseDoorRequest">
  <wsdl:part name="parameters" element="tdc:LockDownReleaseDoor"/>
</wsdl:message>
<wsdl:message name="LockDownReleaseDoorResponse">
  <wsdl:part name="parameters" element="tdc:LockDownReleaseDoorResponse"/>
</wsdl:message>
<wsdl:message name="LockOpenDoorRequest">
  <wsdl:part name="parameters" element="tdc:LockOpenDoor"/>
</wsdl:message>
<wsdl:message name="LockOpenDoorResponse">
  <wsdl:part name="parameters" element="tdc:LockOpenDoorResponse"/>
</wsdl:message>
<wsdl:message name="LockOpenReleaseDoorRequest">
  <wsdl:part name="parameters" element="tdc:LockOpenReleaseDoor"/>
</wsdl:message>
<wsdl:message name="LockOpenReleaseDoorResponse">
  <wsdl:part name="parameters" element="tdc:LockOpenReleaseDoorResponse"/>
</wsdl:message>
<wsdl:message name="DoubleLockDoorRequest">
  <wsdl:part name="parameters" element="tdc:DoubleLockDoor"/>
</wsdl:message>
<wsdl:message name="DoubleLockDoorResponse">
  <wsdl:part name="parameters" element="tdc:DoubleLockDoorResponse"/>
</wsdl:message>
<wsdl:portType name="DoorControlPort">
  <wsdl:operation name="GetServiceCapabilities">
    <wsdl:input message="tdc:GetServiceCapabilitiesRequest"/>
    <wsdl:output message="tdc:GetServiceCapabilitiesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetDoorInfoList">
    <wsdl:input message="tdc:GetDoorInfoListRequest"/>
    <wsdl:output message="tdc:GetDoorInfoListResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetDoorInfo">
    <wsdl:input message="tdc:GetDoorInfoRequest"/>
    <wsdl:output message="tdc:GetDoorInfoResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetDoorState">
    <wsdl:input message="tdc:GetDoorStateRequest"/>
    <wsdl:output message="tdc:GetDoorStateResponse"/>
  </wsdl:operation>
  <wsdl:operation name="AccessDoor">
    <wsdl:input message="tdc:AccessDoorRequest"/>
```

```
    <wsdl:output message="tdc:AccessDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="LockDoor">
    <wsdl:input message="tdc:LockDoorRequest"/>
    <wsdl:output message="tdc:LockDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="UnlockDoor">
    <wsdl:input message="tdc:UnlockDoorRequest"/>
    <wsdl:output message="tdc:UnlockDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="BlockDoor">
    <wsdl:input message="tdc:BlockDoorRequest"/>
    <wsdl:output message="tdc:BlockDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="LockDownDoor">
    <wsdl:input message="tdc:LockDownDoorRequest"/>
    <wsdl:output message="tdc:LockDownDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="LockDownReleaseDoor">
    <wsdl:input message="tdc:LockDownReleaseDoorRequest"/>
    <wsdl:output message="tdc:LockDownReleaseDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="LockOpenDoor">
    <wsdl:input message="tdc:LockOpenDoorRequest"/>
    <wsdl:output message="tdc:LockOpenDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="LockOpenReleaseDoor">
    <wsdl:input message="tdc:LockOpenReleaseDoorRequest"/>
    <wsdl:output message="tdc:LockOpenReleaseDoorResponse"/>
   </wsdl:operation>
   <wsdl:operation name="DoubleLockDoor">
    <wsdl:input message="tdc:DoubleLockDoorRequest"/>
    <wsdl:output message="tdc:DoubleLockDoorResponse"/>
   </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="DoorControlBinding" type="tdc:DoorControlPort">
   <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <!--================================-->
   <wsdl:operation name="GetServiceCapabilities">
    <soap:operation
soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/GetServiceCapabilities"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <!--================================-->
   <wsdl:operation name="GetDoorInfoList">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/GetDoorInfoList"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <!--================================-->
   <wsdl:operation name="GetDoorInfo">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/GetDoorInfo"/>
    <wsdl:input>
```

```
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="GetDoorState">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/GetDoorState"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="AccessDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/AccessDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="LockDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/LockDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="UnlockDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/UnlockDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="BlockDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/BlockDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="LockDownDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/LockDownDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
```

```
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="LockDownReleaseDoor">
    <soap:operation
soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/LockDownReleaseDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="LockOpenDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/LockOpenDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="LockOpenReleaseDoor">
    <soap:operation
soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/LockOpenReleaseDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
  <wsdl:operation name="DoubleLockDoor">
    <soap:operation soapAction="http://www.onvif.org/ver10/doorcontrol/wsdl/DoubleLockDoor"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <!--================================-->
 </wsdl:binding>
</wsdl:definitions>
```

## A.3 Common schema

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

-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:pt="http://www.onvif.org/ver10/pacs" targetNamespace="http://www.onvif.org/ver10/pacs"
elementFormDefault="qualified" version="1.0">
 <!--====== types ======-->
```

```
  <xs:simpleType name="ReferenceToken">
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
      <xs:minLength value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="DataEntity">
    <xs:sequence/>
    <xs:attribute name="token" type="pt:ReferenceToken" use="required"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="Name">
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
      <xs:minLength value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="Description">
    <xs:restriction base="xs:string">
      <xs:maxLength value="1024"/>
      <xs:minLength value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->

</xs:schema>
```

# Annex B
(informative)

## Mapping of mandatory functions in IEC 60839-11-1

The following Tables B.1 to B.6 provide mapping of mandatory and optional requirements in IEC 60839-11-1 to Web services specification. The monitoring (Mon column in table) and administration (Admin column in table) are typically done by a monitoring console.

**Table B.1 – Access point interface requirements**

| Access point interface requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| | **A – Release timing** | | | | | | |
| 1 | The release time shall be system-defined | OP* | OP* | NP | NP | NR | NR |
| 2 | The release time shall be configurable per portal | OP* | OP* | M | M | NR | NS |
| 3 | When the release time is system-defined, the permitted value shall not to be less than 3 s | M | M | NA | NA | NR | NR |
| 4 | When the release time is configurable, several permitted values can be associated to access rights per portal | OP | OP | OP | OP | NR | NR |
| | **B – Access control** | | | | | | |
| 5 | Provide access control for entry into a protected (controlled) area | M | M | M | M | NR | NR |
| 6 | Provide access control for exit from a protected (controlled) area | OP | M | M | M | NR | NR |
| 7 | Hard anti-passback | OP | OP | M | M | NS | NS |
| 8 | Soft anti-passback | OP | OP | OP | OP | NS | NS |
| 9 | Global anti-passback | OP | OP | OP | M | NS | NS |
| 10 | Anti-passback override/disabling | OP | OP | OP | M | NS | NS |
| 11 | Timed anti-passback | OP | OP | OP | M | NS | NS |
| 12 | Access granted conditional upon effective/expiry date | OP | OP | M | M | NS | NS |
| 13 | Access granted conditional upon credential validity (blocked, suspended, invalid) | M | M | M | M | NS | NS |
| 14 | Visitor escorted access | OP | OP | OP | OP | NS | NS |
| 15 | Supervisor mode | OP | OP | OP | OP | NS | NS |
| 16 | Dual occupancy (two or more persons presence check) | OP | OP | OP | OP | NS | NS |
| 17 | Dual access (two-person access) | OP | OP | OP | M | NS | NS |
| 18 | Singularization/anti-tailgating | OP | OP | OP | OP | NS | NS |
| 19 | Elevator control | OP | OP | OP | OP | NS | NS |
| | **C – Access point status monitoring** | | | | | | |
| 20 | Access point/status shall be monitored | OP | M | M | M | S (32: 6.5) | S (32: 6.5) |
| 21 | Access point permitted open time shall be system-defined (recommended open time to be not less than 10 s) | OP | OP* | NP | NP | NS | NS |
| 22 | Access point open time shall be configurable per portal | OP | OP* | M | M | NR | NS |
| 23 | When configurable, several permitted open times may be associated with access rights per access point | OP | OP | OP | OP | NR | NS |

| Access point interface requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| | **D – Input signals** | | | | | | |
| 24 | Digital input signals (i.e. other than communication signals) with an active period exceeding 400 ms shall be processed | OP | M | M | M | S (31:9) | NR |

**Key**

NP: not permitted

OP: optional

M: mandatory

OP*: one of the options in the identified grouping (grey area) shall be implemented

NR: Not relevant to the protocol defined by this document

NS: Not supported by this document

S: Supported (31 refers to IEC60839-1-31 and 32 refers to IEC60839-11-32 followed by clause reference in the respective document)

**Table B.2 – Indication and annunciation requirements**

| Indication and annunciation requirements | | Indication | | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Display | Alert | Logging | 1 | 2 | 3 | 4 | | |
| | **B – Monitoring console (annunciation)** | | | | | | | | | |
| 5 | Visual annunciation is required when access is granted | • | | | OP | OP | OP | OP | S (32: 5.7.3) | S (31: 9.12.3) |
| 6 | Logging is required when access is granted | | | • | OP | OP | M | M | S (32: 5.7.3) | S (31: 9.12.3) |
| 7 | Visual annunciation, alert and logging is required for duress conditions | • | • | • | OP | OP | OP | M | S (32: 5.7.3) | S (31: 9.12.3) |
| 8 | Card usage counter | • | | • | OP | OP | OP | OP | S (32: 5.7.3) | S (31: 9.12.3) |
| 9 | Visual annunciation, alert and logging required for denial of access due to an attempt to use a token with expired validity | • | • | • | OP | OP | OP | M | S (32: 5.7.6) | S (31: 9.12.3) |
| 10 | Visual annunciation, alert and logging required for denial of access due to a configurable number of attempts to use a valid token with invalid memorized information. Where the number of attempts is not configurable it shall be limited to 5 | • | • | • | OP | OP | OP | M | S (32: 5.7.6) | NS |

| Indication and annunciation requirements | | Indication | | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Display | Alert | Logging | 1 | 2 | 3 | 4 | | |
| 11 | Visual annunciation, alert and logging required for denial of access due to a configurable number of sequential attempts to use invalid memorized information (use PIN only for recognition). Where the number of attempts is not configurable it shall be limited to 5 subsequent attempts within 30 s each | • | • | • | OP | OP | NP | NP | S (32: 5.7.6) | NS |
| 12 | Visual indication of access points alerts on the floor plan of the controlled areas | • | | | OP | OP | OP | M | S (32: 6.4.2.1) | S (31: 9.12.3) |
| 13 | Instructions shall be displayed following alerts | • | | | OP | OP | OP | M | S (32: 6.4.2.1) | S (31: 9.12.3) |
| 14 | Transactions | | | • | OP | M | M | M | S (32: 5.7.7) | S (31: 9.12.3) |
| 15 | Visual annunciation and logging for portal open status following access granted. It may be configurable by portal | • | | • | OP | OP | M | M | S (32, 5.7.4) | S (31: 9.12.3) |
| 16 | Visual annunciation, alert and logging for portal remain closed status following access granted. It may be configurable by portal | • | • | • | OP | OP | OP | M | S (32: 5.7.5) | S (31: 9.12.3)) |
| 17 | Access denied. It may be configurable by portal | • | • | • | OP | OP | M | M | S (32: 5.7.6) | S (31: 9.12.3) |
| 18 | Cause of access denial. It may be configurable by portal and/or cause of denial | • | • | • | OP | OP | OP | M | S (32: 5.7.6) | S (31: 9.12.3) |
| 19 | Scheduled or manual portal status change | | | • | OP | OP | M | M | S (32: 6.4.2.1) | NS |
| 20 | Primary power failure | • | • | • | OP | OP | M | M | NS | NS |
| 21 | Primary power restoration | • | | • | OP | OP | M | M | NS | NS |
| 22 | Standby power supply trouble condition (low battery voltage level and no battery present) | • | • | • | OP | OP | M | M | S (31: 8.7.6) | S (31: 8.7.6) |
| 23 | Entering and leaving configuration mode | • | | • | OP | OP | M | M | NR | NR |
| 24 | Loss of communication between access control unit and monitoring console | • | • | • | OP | M | M | M | S (31: 9.2.2) | S (31: 9.2.2) |
| 25 | Roll call | • | | • | OP | OP | M | M | S (32, 5.7.3) | S 31: 9.12.3) |
| 26 | Portal closed following portal forced open or portal opened too long | • | | • | OP | OP | M | M | S (32: 6.4.2.4) | S (31: 9.12.3) |

| Indication and annunciation requirements | | Indication | | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Display | Alert | Logging | 1 | 2 | 3 | 4 | | |
| 27 | All events shall be identified by type, location, time and date of occurrence | • | | • | OP | OP | M | M | S (31: 9.5.1) | NR |
| 28 | Alerts shall contain an indication of their respective priority level if the system allows assigning of such priority levels | • | | • | OP | OP | M | M | NS | NS |
| 29 | Concurrently received alerts shall be displayed by order of priority if the system allows assigning of such priority levels | • | | | OP | OP | M | M | NS | NS |
| 30 | Tamper detection | • | • | • | OP | M | M | M | S (32: 6.6.2) | S (31: 9.12.3) |
| 31 | Portal forced open | • | • | • | OP | M | M | M | S (32: 6.4.2.4) | S (31: 9.12.3) |
| 32 | Visual annunciation, alert and logging for expiry of portal allowed open time (portal opened too long) | • | • | • | OP | M | M | M | S (32: 6.4.2.4) | S (31: 9.12.3) |
| 33 | Card trace | • | | • | OP | OP | OP | M | NS | NS |
| 34 | Reader trace | • | | • | OP | OP | OP | M | NS | NS |
| 35 | Reader condition off-line | • | • | • | OP | OP | OP | M | NS | NS |
| 36 | Locking device abnormal status | • | • | • | OP | OP | OP | M | S (32: 6.6.2) | S (31: 9.12.3) |
| 37 | Annunciation of reaching the limit of 90 % from maximum logging capacity | • | • | • | OP | OP | M | M | NR | NR |
| 38 | Maximum delay time for signals reaching the monitoring console (90 s, 45 s and 15 s) | • | • | • | OP | 90 s | 45 s | 15 s | NR | NR |
| 39 | Maximum delay time for displaying text instructions following alert reaching the monitoring console (5 s) | • | • | | OP | OP | OP | M | NR | NR |
| 40 | Maximum delay time for displaying image and graphics following alert reaching the monitoring console (6 s) | • | • | | OP | OP | OP | 6 s | NR | NR |
| 41 | System shall be capable to assign priority levels to specific *alert events* | • | | | OP | OP | M | M | NR | NR |
| 42 | Alerts received at the monitoring console require acknowledge-ment by the operator | • | • | • | OP | OP | M | M | NR | NR |

| Indication and annunciation requirements | | Indication | | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Display | Alert | Logging | 1 | 2 | 3 | 4 | | |
| 43 | Visual annunciation, alert and logging are required when dual/multiple occupancy conditions are not respected (minimum number of persons not present) | • | • | • | OP | OP | OP | OP | NS | NS |
| 44 | All operator initiated changes shall be logged with type, operator ID, time and date of the occurrence | | | • | OP | OP | OP | M | NR | NR |
| 45 | Operator comments to alerts shall be logged with operator ID, time and date of entering the comment. The specific alert covered by the comments shall be identified | • | | • | OP | OP | OP | M | NR | NR |
| 46 | Accessing logged information for retrieving (e.g. displaying, printing, exporting) events shall be logged with operator ID, time and date of occurrence | | | • | OP | OP | M | M | NR | NR |
| 47 | Minimum number of system events logging capacity on average per reader | | | • | OP | 200 | 500 | 1000 | NR | NR |

**Key**

OP: optional

M: mandatory

NR: Not relevant to the protocol defined by this document

NS: Not supported by this document

S: Supported (31 refers to IEC60839-1-31 and 32 refers to IEC60839-11-32 followed by clause reference in the respective document)

**Table B.3 – Recognition requirements**

| Recognition requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| | **A – Access levels** | | | | | | |
| 1 | The built-in real time clock shall have an accuracy of ±10 seconds a week and be capable of adjusting to daylight saving time, leap year | OP | M | M | M | NR | NR |
| 2 | The system shall be capable of managing multiple time zones | OP | OP | OP | OP | NR | NR |
| 3 | For systems with multiple interconnected control units, the clocks shall be synchronized with the master clock or other reliable synchronization source, at least once every 24 h | OP | OP | M | M | NR | S (31: 8.3.7) |
| 4 | Synchronize the master clock of the system to the official time | OP | OP | OP | M | NR | S (31: 8.3.7) |
| 5 | Real time clock shall be kept for the indicated minimum period of time in case of total power loss (except for loss of data retention battery) | OP | 24 h | 120 h | 120 h | NR | NR |
| 6 | Minimum number of user access levels | 1 | 8 | 16 | 64 | NR | NS |
| 7 | Minimum number of configurable time periods | 0 | 4 | 8 | 16 | NR | NS |
| 8 | Minimum resolution for time within access level includes day of week, hour and minute of day | NA | M | M | M | NR | NS |
| 9 | Minimum resolution for time within access level includes day of month, month and year | NA | OP | OP | M | NR | NS |
| 10 | System shall be capable to handle a number of configurable days (e.g. statutory holidays, special business days and non-business days) | NA | 2 | 16 | 24 | NR | NS |
| 11 | System should be capable of assigning access rights to a group of credentials | OP | OP | OP | OP | NR | NS |
| 12 | System should be capable of changing access rights to a group of credentials in response to emergency conditions | OP | OP | OP | OP | NR | NS |
| | **B – Equipment and methods of recognition** | | | | | | |
| 13 | The system shall assign unique identity to each authorized user | OP | M | M | M | NR | NS |
| 14 | The system shall use memorized information only | OP* | OP* | NP | NP | NR | NS |
| 15 | The system shall use biometrics alone or in combination with other recognition methods | OP* | OP* | OP* | OP* | NR | NS |
| 16 | The system shall use a token | OP* | OP* | OP* | OP* | NR | NS |
| 17 | The system shall use memorized information and a token | OP* | OP* | OP* | OP* | NR | NS |
| 18 | Access shall be denied after each attempt to gain access using a valid token with invalid memorized information, and after a predetermined number of unsuccessful attempts the access rights for that token shall be suspended for a pre-set duration.<br>The number of attempts can be configurable. Where it is not configurable the number of attempts shall be limited to 5 | OP | M | M | M | S (32: 5.7.6) | NS |
| 19 | Access shall be denied after each attempt to gain access with invalid memorized information only. The access shall be suspended after 5 sequential incorrect inputs within a pre-set period of time | OP | OP | NA | NA | S (32: 5.7.6) | NS |

| Recognition requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| 20 | When using biometrics, *FAReff* shall not exceed limits shown for each grade.<br>NOTE 1 *FAReff* = *FAR* (false acceptance rate) when 1:1 comparison is performed (e.g. biometric verification of an identity claimed by memorized information or token) or<br>*FAReff* = *FAR* x *n* when 1:*n* comparison is performed and<br>*n* = number of stored templates (e.g. biometric identification without using memorized information or token).<br>NOTE 2 The *FAR* values are based on the review of the supplied manufacturer's documentation. | 1 % | 0,3 % | 0,3 % | 0,1 % | NR | NR |
| 21 | The minimum ratio between the number of possible user codes and the number of allocated codes shall be at least 1 000 to 1 when the system is using recognition of a valid user by memorized information only<br>e.g.:up to 10 users – 4 digits,<br>up to 100 users – 5 digits,<br>up to 1 000 users – 6 digits, etc | M | M | NA | NA | NR | NR |
| 22 | For systems using recognition by memorized information combined with token or biometrics the memorized information requires 4 digits minimum | OP | OP | M | M | NR | NS |
| 23 | In normal mode of operation the system shall use complete token information (facility code and card number, or unique card number) for recognition | M | M | M | M | NR | NR |
| 24 | System utilizing facility coding to support multiple facility codes | OP | OP | OP | M | NR | NR |
| 25 | In degraded mode of operation the system may use partial token information (e.g. facility code only) for recognition | OP | OP | OP | NP | NR | NR |
| 26 | Tokens with coding system structure visible to unaided human eye shall not be used | M | M | M | M | NR | NR |
| 27 | The token identity number readable on the token not to be a direct representation of the entire coding | M | M | M | M | NR | NR |

**Key**

NP: not permitted

OP: optional

M: mandatory

OP*: one of the options in the identified grouping (grey area) shall be implemented

NA: not applicable

NR: Not relevant to the protocol defined by this document

NS: Not supported by this document

S: Supported (31 refers to IEC60839-1-31 and 32 refers to IEC60839-11-32 followed by clause reference in the respective document)

**Table B.4 – Duress signalling requirements**

| Duress signalling requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| 1 | Enabling of the duress functionality shall be configurable | OP | OP | OP | M | S (32: 5.7.7) | S (32: 5.7.7) |
| 2 | The duress alert at the monitoring console to be distinct from other alerts | M* | M* | M* | M | NR | NR |
| 3 | The operation of the duress initiating device shall not produce a signal which may be audible or visible at the location where the duress has been initiated | M* | M* | M* | M | NR | NR |

**Key**

OP: optional

M: mandatory

M*: mandatory only if optional functionality is supported for the specified grade

NR: Not relevant to the protocol defined by this document

S: Supported (31 refers to IEC60839-1-31 and 32 refers to IEC60839-11-32 followed by clause reference in the respective document)

**Table B.5 – Overriding requirements**

| Overriding requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| 1 | Single free access granting, single portal | OP | OP | M | M | S (32: 6.5.2) | NR |
| 2 | System-wide free access granting | OP | OP | OP | OP | S (32: 6.5.2) | NR |
| 3 | Free access granting until further system command, single portal or group of portals | OP | OP | OP | OP | S (32: 6.5.4) | NR |
| 4 | Scheduled/timed free access granting, single portal or group of portals | OP | OP | OP | OP | NS | NS |
| 5 | The electronic access control system shall not prohibit the free exit granted by other emergency systems (e.g. fire, environmental) | M | M | M | M | NR | NR |
| 6 | Blocking of portal until further system command, single portal or group of portals | OP | OP | OP | OP | S (32: 6.5.5) | NR |
| 7 | Scheduled/timed blocking of portal, single portal or group of portals | NA | OP | OP | OP | NS | NS |

**Key**

OP: optional

M: mandatory

NA: not applicable

NR: Not relevant to the protocol defined by this document

NS: Not supported by this document

S: Supported (31 refers to IEC60839-1-31 and 32 refers to IEC60839-11-32 followed by clause reference in the respective document)

**Table B.6 – System self protection requirements**

| Overriding requirements | | Grade assignment | | | | Mon | Admin |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | |
| 18 | Encryption required for communication signals between components of the EAC system when using publicly shared networks (e.g. the Internet) | OP | OP | M | M | S (31:10) | S (31:10) |

**Key**

OP: optional

M: mandatory

S: Supported (31 refers to IEC60839-1-31 and 32 refers to IEC60839-11-32 followed by clause reference in the respective document)

# Bibliography

IEC 60839-11-31, *Alarm and electronic security systems – Part 11-31: Electronic access control systems – Core interoperability protocol based on Web services*

ONVIF Access Control Specification, *ONVIF Access Control Service Specification Version 1.0.3 June, 2014*
<http://www.onvif.org/specs/srv/access/ONVIF-AccessControl-Service-Spec-v103.pdf>

ONVIF Door Control Specification, *ONVIF Door Control Service Specification Version 1.0.2 June, 2014*
<http://www.onvif.org/specs/srv/door/ONVIF-DoorControl-Service-Spec-v102.pdf>

# SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

# SYSTÈMES D'ALARME ET DE SÉCURITÉ ÉLECTRONIQUES –

## Partie 11-32: Systèmes de contrôle d'accès électronique – Commande de contrôle d'accès en fonction des services Web

## AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.



La Norme internationale IEC 60839 a été établie par le comité d'études 79 de l'IEC: Systèmes d'alarme et de sécurité électroniques.

Le texte de cette norme est issu des documents suivants:

| CDV | Rapport de vote |
|---|---|
| 79/523/CDV | 79/547/RVC |

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 60839, publiées sous le titre général *Systèmes d'alarme et de sécurité électroniques*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. À cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

**IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.**

# INTRODUCTION

Le présent document permet d'établir un système d'alarme et de sécurité électroniques avec des clients, généralement une console de commande, et des dispositifs, généralement une unité de contrôle d'accès, de différents fabricants qui utilisent des interfaces communes et bien définies.

Le présent document spécifie uniquement le flux de données et de commande entre un client et les services sans aucune référence à un dispositif physique quel qu'il soit, dans la mesure où les services exigés pour la mise en œuvre d'un système de contrôle d'accès électronique conforme ne sont pas nécessairement mis en œuvre sur un dispositif unique, c'est-à-dire que tous les services peuvent être exécutés sur un tableau de commande, un logiciel agrégateur d'événements sur PC, etc.

Le présent document ne définit pas la communication interne entre une unité de contrôle d'accès et ses composants s'ils sont mis en œuvre sur un dispositif unique.

Le présent document a été élaboré sur la base des travaux réalisés par le Forum industriel ouvert ONVIF (Open Network Video Interface Forum). La [spécification de contrôle d'accès ONVIF] et la [spécification de commande de portes ONVIF] sont compatibles avec le présent document.

Le présent document s'accompagne d'un ensemble de définitions d'interfaces informatiques:

- WSDL de service de contrôle d'accès, voir l'Article A.1;
- WSDL de service de commande de portes, voir l'Article A.2;
- Schéma commun, voir l'article A.3;

En raison des différences terminologiques entre l'IEC 60839-11-1, l'IEC 60839-11-2 et la spécification ONVIF sur lesquelles est basée la présente partie de l'IEC 60839, il convient qu'un lecteur tienne tout particulièrement compte de l'article termes et définitions.

Les services supplémentaires nécessaires à la configuration d'un système de contrôle d'accès électronique (EACS – Electronic Access Control System) tels que les définitions de programmes, le traitement des règles d'accès, les lecteurs et les identifiants ne relèvent pas du domaine d'application du présent document. Ces services sont traités par d'autres parties de la famille de normes IEC 60839-11-3x.

# SYSTÈMES D'ALARME ET DE SÉCURITÉ ÉLECTRONIQUES –

# Partie 11-32: Systèmes de contrôle d'accès électronique – Commande de contrôle d'accès en fonction des services Web

## 1 Domaine d'application

La présente partie de l'IEC 60839 définit l'interface de services Web pour les systèmes de contrôle d'accès électronique. Ceci inclut l'énumération des composants de systèmes de contrôle d'accès électronique, de leur composition logique, le contrôle de leur état ainsi que leur contrôle. Ceci inclut également le mapping des exigences obligatoires et facultatives conformément à l'IEC 60839-11-1.

Le présent document s'applique uniquement à la sécurité physique. La sécurité physique empêche l'accès physique à un bâtiment, un local, etc., à tout personnel non autorisé, à des agresseurs ou à des intrus occasionnels.

L'utilisation des services Web et la fonctionnalité de gestion de dispositif ne relèvent pas du domaine d'application du présent document. Se reporter à l'IEC 60839-11-31 pour de plus amples informations.

Le présent document n'empêche en aucune façon un fabricant d'ajouter un autre protocole ou d'étendre le protocole défini ici. Se reporter à l'IEC 60839-11-31 pour les règles d'ajout ou d'extension.

## 2 Références normatives

Les documents suivants cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 60839-11-1, *Systèmes d'alarme et de sécurité électroniques – Partie 11-1: Systèmes de contrôle d'accès électronique – Exigences système et exigences concernant les composants*

IEC 60839-11-2, *Systèmes d'alarme et de sécurité électroniques – Partie 11-2: Systèmes de contrôle d'accès électronique – Lignes directrices d'application*

## 3 Termes, définitions et termes abrégés

### 3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans l'IEC 60839-11-1 et l'IEC 60839-11-2, ainsi que les suivants s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

NOTE Lorsque le terme IEC défini dans l'IEC 60839-11-1 et l'IEC 60839-11-2 est différent du terme utilisé dans le présent document, le terme IEC est indiqué entre parenthèses dans l'en-tête de la section.

**3.1.1**
**point d'accès**
**accès contrôlé**
entrée/sortie physique à laquelle l'accès peut être contrôlé par une porte, un tourniquet ou toute autre barrière de sécurité

Note 1 à l'article: Pour les besoins du présent document, le point d'accès est considéré comme étant une composition logique d'une porte physique et d'un ou de plusieurs lecteurs contrôlant l'accès dans une direction.

Note 2 à l'article: Dans le présent document, le terme "porte" a la même signification que le point d'accès ou l'accès contrôlé.

**3.1.2**
**dispositif d'activation d'un point d'accès**
**dispositif d'activation d'un accès contrôlé**
partie d'un système de contrôle d'accès en interface avec une unité de contrôle d'accès qui libère et protège un accès contrôlé selon les règles préétablies

Note 1 à l'article: Dans le présent document, le terme “dispositif de verrouillage de porte” est utilisé.

**3.1.3**
**mode de point d'accès**
mode de fonctionnement logique de l'accès contrôlé indiquant son verrouillage, déverrouillage, blocage, verrouillage en position basse ou verrouillage en position ouverte, etc.

Note 1 à l'article: Dans le présent document, le terme “mode de porte” est utilisé.

**3.1.4**
**capteur d'un point d'accès**
**capteur d'un accès contrôlé**
composant électronique utilisé pour contrôler l'état d'ouverture ou fermeture d'un point d'accès, ou l'état verrouillé/déverrouillé d'un dispositif de verrouillage, voire l'état de protection/absence de protection d'une serrure électromagnétique ou d'une plaque d'armature

Note 1 à l'article: Dans le présent document, le terme “dispositif de contrôle de porte” est utilisé.

**3.1.5**
**neutralisation d'un point d'accès**
**neutralisation d'un accès contrôlé**
émission effective d'une commande manuelle de contournement du mode de fonctionnement préconfiguré (c'est-à-dire libération/protection/blocage) d'un point d'accès

Note 1 à l'article: Dans le présent document, les termes “accès momentané” et “déverrouillé” sont des exemples de neutralisation d'un point d'accès.

**3.1.6**
**alarme**
<système de contrôle d'accès> condition exigeant l'évaluation ou l'intervention d'un individu

Note 1 à l'article: Utilisée souvent dans un système de contrôle d'accès électronique au sens d'une alerte.

Note 2 à l'article: Dans le présent document, le terme “alarme de porte” est utilisé.

**3.1.7**
**client**
demandeur de service

EXEMPLE Gestion du système, annonce, console de commande.

**3.1.8**
**dispositif**
fournisseur de service

EXEMPLE: Unité de contrôle d'accès.

**3.1.9**
**côté accès contrôlé**
composition logique d'une porte physique et d'un ou plusieurs lecteurs contrôlant l'accès dans une direction

### 3.2 Termes abrégés

Pour les besoins du présent document, les termes abrégés donnés dans l'IEC 60839-11-1 et l'IEC 60839-11-2, ainsi que les suivants s'appliquent.

ACMS Access Control Management System (Système de gestion de contrôle d'accès)
BMS Building Management System (Système de gestion des bâtiments)
HTTP Hypertext Transfer Protocol (Protocole HTTP)
PSIM Physical Security Information Management (Gestion de l'information sur la sécurité physique)
SOAP Simple Object Access Protocol (Protocole SOAP)
TLS Transport Layer Security (Sécurité de couche transport)
WSDL Web Services Description Language (Langage de description de services Web)

## 4 Présentation

### 4.1 Interopérabilité

Le présent document fournit de nouvelles possibilités d'interopérabilité en différenciant la configuration de la commande et du contrôle. Dans les systèmes classiques, le système de gestion central transfère toutes les données de configuration vers les dispositifs en cours de démarrage et ne prévoit aucune modification de ces données par d'autres clients. En revanche, un client doit s'attendre à ce que toutes les informations soient archivées sur des dispositifs terminaux et puissent être modifiées par des tiers.

Le système de contrôle d'accès électronique défini dans le présent document repose sur des principes d'architecture orientée service. Ceci autorise des installations dont les différents composants peuvent être remplacés ou mis à jour de manière indépendante.

### 4.2 Traitement des événements

Le traitement des événements constitue une partie fondamentale des opérations de contrôle d'accès. Outre la remise d'événements en temps réel, l'IEC 68839-11-31 prévoit les moyens d'accès aux événements archivés périphériques afin d'assurer leur remise en cas de perte de connexion.

Les événements sont divisés en 3 groupes selon leur origine et leur objectif:

1) Les événements de modification de configuration. Ces événements servent à réaliser une interopérabilité entre plusieurs clients contrôlant simultanément un dispositif unique.
2) Les événements de transaction. Fonctionnalité centrale d'un système de contrôle d'accès électronique qui assure le contrôle quotidien de tous les événements d'accès, y compris les événements à accès autorisé destinés à signifier aux clients toutes les informations détaillées (personne ayant transmis les événements, moment de leur transmission et vraisemblablement leur destination) concernant chaque événement particulier à autorisation d'accès, les événements à refus d'accès (qui peuvent ou non contenir les informations de motivation de ce refus), etc.
3) Les événements d'alarmes et de défauts. Ces événements assurent le contrôle du bon état de fonctionnement permettant aux opérateurs de prendre des mesures en cas de défaillance matérielle, d'intrusion ou de toute autre activité suspecte.

Se reporter à l'IEC 68839-11-31 pour les détails concernant le mécanisme de remise des événements et à 5.7 pour la liste des événements définis par le présent document.

### 4.3 Architecture

Le présent document n'impose aucune configuration spécifique pour le dispositif physique. Le schéma fourni à la Figure 1 n'est pas destiné à servir de modèle, mais plutôt à être utilisé comme référence pour une meilleure compréhension de la spécification indiquée. Différentes configurations physiques d'une porte à contrôle d'accès sont possibles sur la base des définitions ci-dessous.

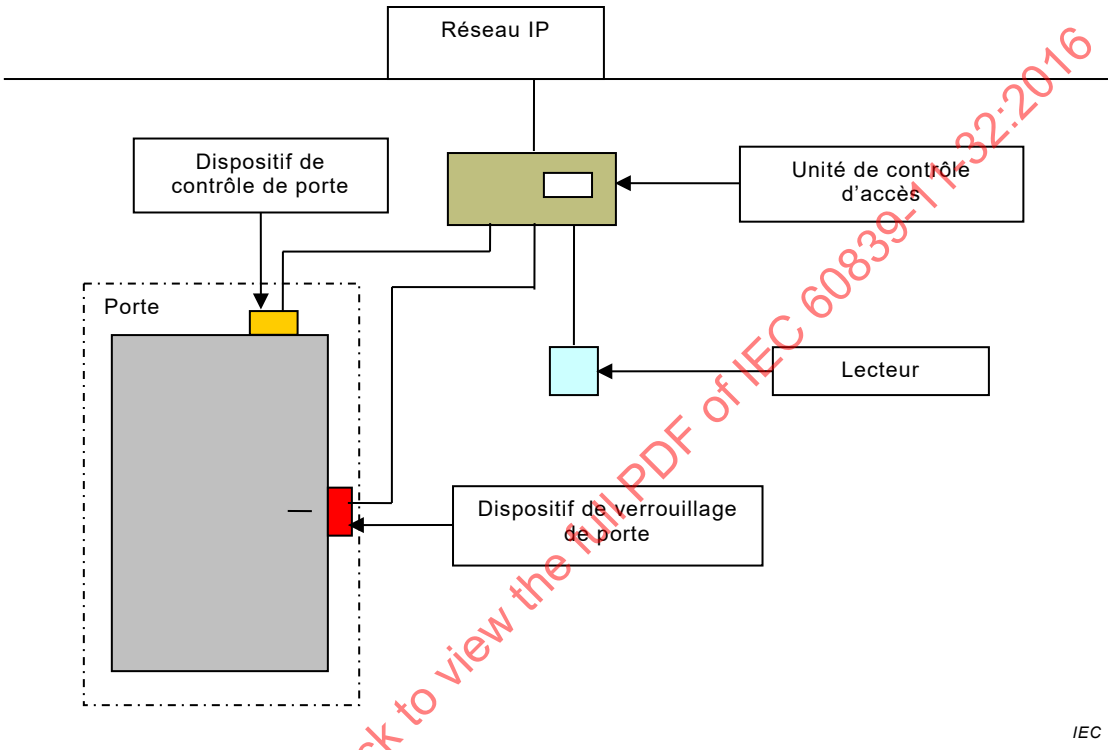


**Figure 1 – Présentation schématique d'une porte à contrôle d'accès**

Une porte commandée par un système de contrôle d'accès électronique est équipée des dispositifs suivants:

- Une unité de contrôle d'accès qui assure les connexions du lecteur, du capteur de porte, du dispositif de verrouillage de porte, ainsi que des entrées et sorties numériques supplémentaires. Ce tableau permet une interaction entre le logiciel et les dispositifs physiques. Parfois, ces tableaux contiennent également un dispositif d'archivage et un système intelligent local afin de fournir une fonctionnalité hors tension, de sorte que la porte fonctionne normalement, même en l'absence d'un système de gestion supérieur.
- Un lecteur capable de déchiffrer un identifiant. Dans la plupart des cas, le lecteur est installé uniquement sur le côté extérieur (non protégé) de la porte. Un lecteur de cartes est installé sur les deux côtés de la porte lorsque le système contrôle les sorties de zone par un ou plusieurs individus.
- Un dispositif de contrôle de porte qui informe le tableau de commande de l'ouverture ou de la fermeture de la porte.
- Un dispositif de verrouillage de porte qui peut être manœuvré par l'unité de contrôle d'accès pour libérer la porte, par exemple, en cas de reconnaissance d'un identifiant autorisé.

L'unité de contrôle d'accès est connectée à un système par le réseau IP, généralement une console de commande, à des fins de commande et de configuration.

### 4.4 Autorisation externe (Neutralisation)

L'autorisation externe est une caractéristique qui sert à prendre des décisions d'accès concernant un point d'accès extérieur à l'unité de contrôle d'accès. L'autorisation externe concerne, sans toutefois s'y limiter, une politique interne à l'unité de contrôle d'accès qui délègue les décisions d'accès à une entité extérieure telle qu'un protecteur ou un système de gestion de contrôle d'accès.

### 4.5 Considérations de sécurité

Par hypothèse, le présent document envisage la possibilité d'établir des systèmes de contrôle d'accès électronique qui interagissent au niveau du dispositif. Ceci implique des considérations de sécurité plus importantes que dans le cas d'une interaction client-serveur habituelle. L'IEC 68839-11-31 définit plusieurs mécanismes de cette nature. Ces mécanismes incluent, sans toutefois s'y limiter

- sécurité de couche transport (TLS) pour le cryptage de transport;
- HTTP digest pour l'authentification du client;
- des politiques de gestion et d'accès utilisateur pour l'autorisation du client;
- la gestion des certificats IEEE 802.1X pour l'authentification du serveur et la protection contre la mystification.

Se reporter aux livres blancs et spécifications respectifs pour de plus amples informations.

### 4.6 Commande de portes (point d'accès)

Le service de commande de portes fournit des mécanismes de commande des instances de portes physiques et de contrôle de leur état.

L'entité Porte (Door) dans le présent document peut se rapporter à des objets physiques tels qu'une barrière automatique ou une porte équipée d'un dispositif de verrouillage électrique. Les tourniquets qui peuvent limiter l'accès dans l'une ou l'autre direction peuvent être représentés par une paire de portes.

### 4.7 Considérations de conception

#### 4.7.1 Fonctionnalités de niveau d'instance

Un dispositif unique à système de contrôle d'accès électronique peut avoir divers composants du même type. Par exemple, un contrôleur peut manœuvrer deux portes: une porte à l'entrée du bâtiment qui comporte des dispositifs de verrouillage sécurisé, de contrôle et d'alarme, et une autre porte interne qui peut être uniquement verrouillée et déverrouillée.

Les fonctionnalités peuvent être par conséquent réparties en deux groupes:

- fonctionnalités de service générales;
- fonctionnalités pour une entité particulière au sein du service. Ces fonctionnalités peuvent également être associées à la fonction GetEventProperties afin d'assurer un contrôle plus précis du système.

Se reporter à 5.2 et 6.2 pour de plus amples informations.

#### 4.7.2 Récupération des informations d'état

Le présent document définit deux mécanismes parallèles de récupération des informations d'état pour la plupart des entités:

- Les fonctions Get<*Entity*>State renvoient un instantané cumulé de l'état réel, du mode de fonctionnement et autres informations d'exécution.

- Le service Événement (Event) renvoie des états actualisés et cohérents des entités. Chaque entité fournit un ensemble d'événements (habituellement un événement pour chaque champ du type État (State)) destiné à indiquer à un client les modifications d'état. Dans la mesure où ces événements sont des événements de propriété, un client reçoit l'état réel à chaque initialisation d'un nouvel abonnement.

#### 4.7.3 Récupération de la configuration du système

Le présent document définit plusieurs fonctions Get qui peuvent renvoyer les données de manière progressive. Ces fonctions permettent le traitement d'un grand nombre d'entités bien que les ressources soient fortement limitées.

Ces fonctions, pour renvoyer les données de manière progressive, utilisent un paramètre appelé StartReference. StartReference est un identificateur interne de dispositif utilisé pour poursuivre la récupération des données à partir de la dernière position, et permet à un client d'exécuter un ensemble de données important en blocs plus petits. Le dispositif traite simultanément un nombre raisonnable de paramètres StartReferences différents dont la durée d'application suffisante permet aux clients de récupérer des ensembles de données complets.

Un client transmet toujours la valeur renvoyée d'une demande précédente afin de poursuivre la récupération des données. Les clients utilisent la même référence une seule fois.

Par exemple, le paramètre StartReference peut être le numéro de position de démarrage incrémentiel ou l'identificateur de transaction de base de données sous-jacente.

Le paramètre NextStartReference renvoyé est utilisé comme paramètre de StartReference dans des appels successifs, et peut être modifié par le dispositif lors de chaque appel.

Le pseudo-code suivant démontre comment les informations concernant tous les points d'accès peuvent être obtenues d'un dispositif.

```
StartRef = null
do {
     Response = GetAccessPointInfoList(StartReference = StartRef)
     if (Response.AccessPointInfo != null) {
          AllAccessPoints.Append(Response.AccessPointInfo)
     }
     StartRef = Response.NextStartReference
} while (StartRef != null)
```

## 5 Contrôle d'accès

### 5.1 Généralités

Ce service propose des commandes permettant de récupérer les informations d'état et de contrôler les instances de point d'accès.

### 5.2 Fonctionnalités de service

#### 5.2.1 Généralités

Un dispositif doit fournir des fonctionnalités de service de deux manières différentes:

1) Avec la méthode GetServices du service Device (Dispositif) lorsque IncludeCapability est définie sur true (vrai). Se reporter à l'IEC 68839-11-31 pour de plus amples informations.
2) Avec la méthode GetServiceCapabilities.

### 5.2.2 Structures de données: ServiceCapabilities

Les fonctionnalités de service reflètent la fonctionnalité facultative d'un service. Les informations sont statiques et ne varient pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

- **MaxLimit**
  Nombre maximal d'entités renvoyées par une demande Get<Entity>List ou Get<Entity>request unique. Le dispositif ne doit jamais renvoyer dans une réponse unique un nombre d'entités plus important que ce nombre.

### 5.2.3 Commande GetServiceCapabilities

Cette opération renvoie les fonctionnalités du service de contrôle d'accès. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 1.

**Tableau 1 – Commande GetServiceCapabilities**

| **GetServiceCapabilities** | Classe d'accès: PRE_AUTH |
|---|---|
| **Nom de message** | **Description** |
| GetServiceCapabilitiesRequest | *Ce message doit être vide* |
| GetServiceCapabilitiesResponse | *Ce message contient:*<br>• *"Capabilities": Le message de réponse de fonctionnalité contient les fonctionnalités de service demandées de contrôle d'accès qui utilisent une structure de fonctionnalité XML hiérarchique.*<br>tac:ServiceCapabilities **Capabilities [1][1]** |
| **Codes de défaut** | **Description** |
| | *Pas de défauts spécifiques à la commande* |

## 5.3 Informations concernant le point d'accès (côté accès contrôlé)

### 5.3.1 Structures de données

#### 5.3.1.1 AccessPointInfo

La structure AccessPointInfo contient les informations de base concernant une instance de point d'accès. Une instance de point d'accès définit une entité pour laquelle un identifiant (credential) peut autoriser ou refuser l'accès. La structure AccessPointInfo fournit des informations de base concernant le mode de contrôle d'accès dans une direction pour une porte donnée (d'une zone spécifique à une autre zone spécifique).

Plusieurs instances de point d'accès peuvent couvrir la même porte. Un cas typique est une instance de point d'accès pour l'entrée et une autre pour la sortie, les deux instances faisant référence à la même porte.

Un dispositif doit fournir les champs suivants pour chaque instance de point d'accès:

- **Jeton (Token)**
  Identificateur unique au service du point d'accès.
- **Nom (Name)**
  Nom lisible par l'utilisateur. Il doit être composé de 64 caractères au plus.
- **Entité (Entity)**
  Référence à l'entité utilisée pour le contrôle d'accès. Le type d'entité peut être spécifié par le champ facultatif EntityType expliqué ci-dessous, mais est généralement une porte.
- **Fonctionnalités (Capabilities)**
  Les fonctionnalités du point d'accès.

Afin de fournir de plus amples informations, le dispositif peut inclure les champs facultatifs suivants:

- **Description**
  Description facultative de l'instance de point d'accès lisible par l'utilisateur. Elle doit être composée de 1 024 caractères au plus.
- **AreaFrom**
  Référence facultative à la zone (area) de laquelle l'accès est demandé.
- **AreaTo**
  Référence facultative à la zone à laquelle l'accès est demandé.
- **EntityType**
  Type d'entité facultatif; en son absence, il convient de prendre pour hypothèse un type d'entité Porte tel que défini par le service de commande de porte. Ceci peut être également représenté par la valeur QName “tdc:Door” – où tdc est l'espace de nommage du service Commande de porte (Door Control): http://www.onvif.org/ver10/doorcontrol/wsdl. Ce champ est fourni pour des extensions futures; il permet l'extension d'un point d'accès afin de couvrir des types d'entités également autres que les portes.

#### 5.3.1.2 AccessPointCapabilities

Les fonctionnalités du point d'accès reflètent la fonctionnalité facultative d'une entité physique particulière. Des instances de point d'accès différentes peuvent avoir différents ensembles de fonctionnalités. Ces informations peuvent varier lors du fonctionnement du dispositif, par exemple, en cas de modification des paramètres du matériel. Les fonctionnalités suivantes sont disponibles:

- **DisableAccessPoint**
  Indique si cette instance de point d'accès prend ou non en charge les commandes EnableAccessPoint et DisableAccessPoint.
- **Agression (Duress)**
  Indique si cette instance de point d'accès prend ou non en charge la génération d'événements d'agression.
- **AnonymousAccess**
  Indique si cette instance de point d'accès comporte une entrée REX ou une autre entrée qui autorise un accès anonyme.
- **AccessTaken**
  Indique si cette instance de point d'accès prend ou non en charge la génération d'événements AccessTaken et AccessNotTaken.

  Si les événements AnonymousAccess et AccessTaken sont tous les deux vrais, cela signifie que les versions Anonyme (Anonymous) des fonctionnalités AccessTaken et AccessNotTaken sont également prises en charge.
- **ExternalAuthorization**
  Indique si cette instance AccessPoint prend ou non en charge l'opération ExternalAuthorization et la génération d'événements Demande (Request).

  Si les fonctionnalités AnonymousAccess et ExternalAuthorization sont toutes les deux vraies, cela signifie que la version Anonyme est également prise en charge.

### 5.3.2 Commande GetAccessPointInfoList

Cette opération demande une liste de tous les éléments AccessPointInfo fournis par le dispositif. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 2.

Un appel à cette méthode doit renvoyer un paramètre StartReference lorsque les données ne sont pas toutes renvoyées et lorsqu'un nombre de données plus important est disponible. La référence doit être valide pour pouvoir récupérer l'ensemble de données suivant. Se reporter à 4.7.3 pour de plus amples informations.

Le nombre d'éléments renvoyés ne doit pas être supérieur au paramètre Limite (Limit).

**Tableau 2 – Commande GetAccessPointInfoList**

| **GetAccessPointInfoList** | Classe d'accès: READ_SYSTEM |
|---|---|
| **Nom de message** | **Description** |
| GetAccessPointInfoListRequest | *Ce message contient:*<br>• *"Limit": Nombre maximal d'entrées à renvoyer. Si la Limite est omise ou si la valeur de Limite est supérieure à ce que le dispositif prend en charge, ce dernier doit alors renvoyer son nombre maximal d'entrées.*<br>• *"StartReference": Commencer le renvoi des entrées à partir de cette référence de démarrage. En l'absence de spécification, les entrées doivent commencer du début de l'ensemble de données.*<br>xs:int **Limit [0][1]**<br>xs: string**StartReference [0][1]** |
| GetAccessPointInfoListResponse | *Ce message contient:*<br>• *"NextStartReference": StartReference à utiliser dans le prochain appel pour obtenir les éléments suivants. En son absence, ne pas obtenir d'autres éléments.*<br>• *"AccessPointInfo": Liste des éléments AccessPointInfo.*<br>xs:string **NextStartReference [0][1]**<br>tac:AccessPointInfo**AccessPointInfo [0][non limité]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:InvalidStartReference | *StartReference invalide ou temporisée. Il est nécessaire que le client commence la récupération du début.* |

### 5.3.3 Commande GetAccessPointInfo

Cette opération demande une liste d'éléments AccessPointInfo correspondant aux jetons donnés. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 3.

Le dispositif doit ignorer les jetons qu'il ne peut pas résoudre et doit renvoyer une liste vide si aucun élément ne correspond aux jetons spécifiés. Le dispositif ne doit pas renvoyer d'instance de défaut dans ce cas.

Si le nombre d'éléments demandés est supérieur à la fonctionnalité MaxLimit, une instance de défaut TooManyItems doit être renvoyée.

**Tableau 3 – Commande GetAccessPointInfo**

| **GetAccessPointInfo** | Classe d'accès: READ_SYSTEM |
|---|---|
| **Nom de message** | **Description** |
| GetAccessPointInfoRequest | *Ce message contient:*<br>• *"Token": Jetons des éléments AccessPointInfo à obtenir.*<br>pt:ReferenceToken**Token [1][non limité]** |
| GetAccessPointInfoResponse | *Ce message contient:*<br>• *"AccessPointInfo": Liste des éléments AccessPointInfo.*<br>tac:AccessPointInfo**AccessPointInfo [0][non limité]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgs<br>ter:TooManyItems | *Demande d'un trop grand nombre d'éléments, voir fonctionnalité MaxLimit.* |

## 5.4 Informations concernant la zone

### 5.4.1 Structures de données: AreaInfo

La structure AreaInfo contient les informations de base concernant une zone. Un dispositif doit fournir les champs suivants pour chaque zone:

- **Jeton**
  Identificateur unique au service de la zone.
- **Nom**
  Nom lisible par l'utilisateur. Il doit être composé de 64 caractères au plus. Afin de fournir de plus amples informations, le dispositif peut inclure les champs facultatifs suivants:
- **Description**
  Description de la zone lisible par l'utilisateur. Elle doit être composée de 1 024 caractères au plus.

### 5.4.2 Commande GetAreaInfoList

Cette opération demande une liste de tous les éléments AreaInfo fournis par le dispositif. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 4.

Un appel à cette méthode doit renvoyer un paramètre StartReference lorsque les données ne sont pas toutes renvoyées et lorsqu'un nombre de données plus important est disponible. La référence doit être valide pour pouvoir récupérer l'ensemble de données suivant. Se reporter à 4.7.3 pour de plus amples informations.

Le nombre d'éléments renvoyés ne doit pas être supérieur au paramètre Limite.

**Tableau 4 – Commande GetAreaInfoList**

| **GetAreaInfoList** | Classe d'accès: READ_SYSTEM |
|---|---|
| **Nom de message** | **Description** |
| GetAreaInfoListRequest | *Ce message contient:*<br>• *"Limit": Nombre maximal d'entrées à renvoyer. Si la Limite est omise ou si la valeur de Limite est supérieure à ce que le dispositif prend en charge, ce dernier doit alors renvoyer son nombre maximal d'entrées.*<br>• *"StartReference": Commencer le renvoi des entrées à partir de cette référence de démarrage. En l'absence de spécification, les entrées doivent commencer du début de l'ensemble de données.*<br>xs:int **Limit [0][1]**<br>xs:string **StartReference [0][1]** |
| GetAreaInfoListResponse | *Ce message contient:*<br>• *"NextStartReference": StartReference à utiliser dans le prochain appel pour obtenir les éléments suivants. En son absence, ne pas obtenir d'autres éléments.*<br>• *"AreaInfo": Liste des éléments AreaInfo.*<br>xs:string **NextStartReference [0][1]**<br>tac:AreaInfo**AreaInfo [0][non limité]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:InvalidStartReference | *StartReference invalide ou temporisée. Il est nécessaire que le client commence la récupération du début.* |

### 5.4.3 Commande GetAreaInfo

Cette opération demande une liste d'éléments AreaInfo correspondant aux jetons donnés. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 5.

Le dispositif doit ignorer les jetons qu'il ne peut pas résoudre et peut renvoyer une liste vide si aucun élément ne correspond aux jetons spécifiés.

Si le nombre d'éléments demandés est supérieur à la fonctionnalité MaxLimit, une instance de défaut TooManyItems doit être renvoyée.

**Tableau 5 – Commande GetAreaInfo**

| **GetAreaInfo** | Classe d'accès: READ_SYSTEM |
|---|---|
| **Nom de message** | **Description** |
| GetAreaInfoRequest | *Ce message contient:*<br>• *"Token": Jetons des éléments AreaInfo à obtenir.*<br>pt:ReferenceToken**Token [1][non limité]** |
| GetAreaInfoResponse | *Ce message contient:*<br>• *"AreaInfo": Liste des éléments AreaInfo.*<br>tac:AreaInfo**AreaInfo [0][non limité]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgs<br>ter:TooManyItems | *Demande d'un trop grand nombre d'éléments, voir fonctionnalité MaxLimit.* |

## 5.5 État du point d'accès (côté accès contrôlé)

### 5.5.1 Généralités

L'état du point d'accès est déterminé par un certain nombre d'opérations qui peuvent être réalisées avec lui selon ses fonctionnalités; se reporter aux fonctionnalités du point d'accès en 5.3.1.2.

### 5.5.2 Structures de données: AccessPointState

L'instance AccessPointState contient les informations d'état d'un point d'accès. Un dispositif doit fournir les champs suivants pour chaque instance de point d'accès:

- **Activé (Enabled)**
  Indique que l'instance de point d'accès est activée. Par défaut, cette valeur de champ doit être vraie (true), si la fonctionnalité DisableAccessPoint n'est pas prise en charge.

### 5.5.3 Commande GetAccessPointState

Cette opération demande l'instance AccessPointState pour l'instance de point d'accès spécifiée par l'identifiant Jeton. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 6.

**Tableau 6 – Commande GetAccessPointState**

| **GetAccessPointState** | Classe d'accès: READ_SYSTEM_SENSITIVE |
|---|---|
| **Nom de message** | **Description** |
| GetAccessPointStateRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de point d'accès pour obtenir l'élément AccessPointState.*<br>pt:ReferenceToken **Token [1][1]** |
| GetAccessPointStateResponse | *Ce message contient:*<br>• *"AccessPointState": Élément AccessPointState.*<br>tac:AccessPointState**AccessPointState [1][1]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Instance AccessPoint non trouvée* |

## 5.6 Commandes de contrôle d'accès

### 5.6.1 Généralités

Les commandes de contrôle de service contiennent les opérations qui permettent de modifier les états de point d'accès et de contrôler les points d'accès.

### 5.6.2 Structures de données: énumération Décision

L'énumération Décision (Decision) représente un choix de deux options disponibles pour une demande d'accès:

- **Autorisé (Granted)**
  La décision est d'autoriser l'accès.
- **Refusé (Denied)**
  La décision est de refuser l'accès.

### 5.6.3 Commande EnableAccessPoint

Cette opération permet l'activation d'un point d'accès.

Un dispositif qui indique la prise en charge de la fonctionnalité DisableAccessPoint pour une instance de point d'accès particulière doit prendre en charge cette commande comme décrit dans le Tableau 7.

**Tableau 7 – Commande EnableAccessPoint**

| **EnableAccessPoint** | Classe d'accès: ACTUATE (ACTIONNER) |
|---|---|
| **Nom de message** | **Description** |
| EnableAccessPointRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de point d'accès à activer.*<br>pt:ReferenceToken **Token [1][1]** |
| EnableAccessPointResponse | *Ce message doit être vide* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:ActionNotSupported<br>ter:NotSupported | *L'opération n'est pas prise en charge.* |

### 5.6.4 Commande DisableAccessPoint

Cette opération permet la désactivation d'un point d'accès.

Un dispositif qui indique la prise en charge de la fonctionnalité DisableAccessPoint pour une instance de point d'accès particulière doit prendre en charge cette commande comme décrit dans le Tableau 8.

**Tableau 8 – Commande DisableAccessPoint**

| **DisableAccessPoint** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| DisableAccessPointRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de point d'accès à désactiver.*<br>pt:ReferenceToken **Token [1][1]** |
| DisableAccessPointResponse | *Ce message doit être vide* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:ActionNotSupported<br>ter:NotSupported | *L'opération n'est pas prise en charge.* |

### 5.6.5 Commande ExternalAuthorization

Cette opération permet de prendre une décision Refusé (Denied) ou Autorisé (Granted) à une instance de point d'accès.

Un dispositif qui indique la prise en charge de la fonctionnalité ExternalAuthorization pour une instance de point d'accès particulière doit prendre en charge cette méthode comme décrit dans le Tableau 9.

**Tableau 9 – Commande ExternalAuthorization**

| **ExternalAuthorization** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| ExternalAuthorizationRequest | *Ce message contient:*<br>• *"AccessPointToken": Jeton de l'instance de point d'accès.*<br>• *"CredentialToken": Jeton facultatif de l'Identifiant appelé.*<br>• *"Reason": Raison facultative de la décision.*<br>• *"Decision": Décision – Autorisé ou Refusé.*<br>pt:ReferenceToken **AccessPointToken [1][1]**<br>pt:ReferenceToken **CredentialToken [0][1]**<br>xs:string **Reason [0][1]**<br>tac:Decision **Decision [1][1]** |
| ExternalAuthorizationResponse | *Ce message doit être vide* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Point d'accès non trouvé.* |
| env:Receiver<br>ter:ActionNotSupported<br>ter:NotSupported | *L'opération n'est pas prise en charge.* |



## 5.7 Rubriques de notification

### 5.7.1 Présentation des événements

Le service AccessControl spécifie les événements à utiliser pour les transactions d'accès, par exemple, une demande d'accès est formulée et un accès est autorisé ou refusé, lors de la détection d'une agression, et lors de la modification d'une configuration importante.

Les rubriques principales pour les événements de transaction d'accès sont les suivantes:

- tns1:AccessControl/AccessGranted/ – lors de l'autorisation de l'accès.
- tns1:AccessControl/AccessTaken/ – lorsque l'accès est effectif après autorisation.
- tns1:AccessControl/AccessNotTaken/ – lorsque l'accès n'est pas effectif après autorisation.
- tns1:AccessControl/Denied/ – lors du refus de l'accès.
- tns1:AccessControl/Duress – lors de la détection d'une agression.
- tns1:AccessControl/Request/ – lors de la demande ou de la temporisation d'une autorisation externe.

La rubrique principale pour les actualisations d'état est la suivante:

- tns1:AccessPoint/State/ – pour les actualisations d'état.

Les rubriques principales pour les notifications de modification de configuration sont les suivantes:

- tns1:Configuration/AccessPoint – lors de la modification de AccessPointconfiguration.
- tns1:Configuration/Area – lors de la modification de la configuration de l'entité Zone.

NOTE Le terme “rubrique principale” signifie ici qu'un client peut s'abonner, par exemple, à tns1:AccessControl/AccessGranted, mais l'événement réel envoyé (et ainsi reçu) peut constituer la rubrique principale elle-même ou toute sous-rubrique de cette dernière (telle que tns1:AccessControl/AccessGranted/Credential). De nouvelles sous-rubriques peuvent être définies à l'avenir.

### 5.7.2 Configuration générale des événements de transaction

Tous les événements de transaction utilisent le schéma de message suivant:

```
<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
      Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                              Type="xs:boolean"/>
<tt:SimpleItemDescription Name="CredentialToken"
                              Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                              Type="xs:string"/>
<tt:SimpleItemDescription Name="Reason"
                              Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

lorsque AccessPointToken identifie l'instance AccessPoint où a lieu la prise de la décision.

Les paramètres Données (Data) sont facultatifs. Se reporter aux définitions dans 5.7.3 à 5.7.10 pour leurs exigences. Le paramètre CredentialToken doit être présent et l'élément de données CredentialHolderName peut être présent pour les transactions basées sur les identifiants. Le paramètre facultatif Raison (Reason) contient les informations de raison d'erreur et le paramètre facultatif Externe (External) indique si la transaction a été déclenchée du fait de la commande ExternalAuthorization.

### 5.7.3 Autorisation d'accès (Access granted)

#### 5.7.3.1 Généralités

À chaque décision d'accès positive, c'est-à-dire à chaque autorisation d'accès, un dispositif doit fournir un message d'événement correspondant conformémentà 5.7.3.2 et 5.7.3.3.

L'événement doit être transmis immédiatement une fois la décision prise, que l'accès soit effectif ou non.

#### 5.7.3.2 Anonyme (Anonymous)

À chaque autorisation d'accès à un point d’accès pour un utilisateur anonyme, un dispositif qui indique la fonctionnalité AnonymousAccess pour une instance de point d'accès particulière doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/AccessGranted/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                              Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                              Type="xs:boolean"/>
</tt:Data>
</tt:MessageDescription>
```

Si le déclenchement de la commande est dû à la commande ExternalAuthorization, l'élément de données Externe doit être défini sur true (vrai); à défaut, l'élément est facultatif.

#### 5.7.3.3 Identifiant (Credential)

Chaque fois qu'un identifiant valide satisfait à tous les contrôles nécessaires et qu'un utilisateur ou un détenteur de carte se voit autoriser l'accès à l'instance de point d'accès, mais n'a pas encore utilisé cet accès, voire n'est pas entré ou n'est pas sorti, le dispositif doit fournir les données d'événement suivantes:

```
Topic: tns1:AccessControl/AccessGranted/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

Si le déclenchement de la commande est dû à la commande ExternalAuthorization, l'élément de données Externe doit être défini sur true (vrai); à défaut, l'élément est facultatif.

L'élément de données CredentialHolderName est facultatif.

### 5.7.4 Accès effectif (Access taken)

#### 5.7.4.1 Généralités

Un dispositif qui indique la prise en charge de la fonctionnalité AccessTaken pour une instance de point d'accès particulière doit fournir un événement correspondant destiné à notifier au client tout accès effectif par une personne autorisée.

#### 5.7.4.2 Anonyme

Le dispositif qui indique la prise en charge de la fonctionnalité AnonymousAccess pour une instance de point d'accès particulière doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/AccessTaken/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.4.3 Identifiant

Lorsque le dispositif détecte que l'accès est effectif et que l'identifiant peut être reconnu, il doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/AccessTaken/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
```

```
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>

</tt:MessageDescription>
```

L'élément de données CredentialHolderName est facultatif.

### 5.7.5 Accès non effectif (Access not taken)

#### 5.7.5.1 Généralités

Un dispositif qui indique la prise en charge de la fonctionnalité AccessTaken pour une instance de point d'accès particulière doit fournir un événement correspondant destiné à notifier au client l'accès non effectif dans le délai imparti d'une personne ayant reçu une autorisation d'accès.

#### 5.7.5.2 Anonyme

Le dispositif qui indique la prise en charge de la fonctionnalité AnonymousAccess pour une instance de point d'accès particulière doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/AccessNotTaken/Anonymous

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>

</tt:MessageDescription>
```

#### 5.7.5.3 Identifiant

Lorsque le dispositif détecte que l'accès n'est pas effectif et que l'identifiant peut être reconnu, il doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/AccessNotTaken/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>

</tt:MessageDescription>
```

L'élément de données CredentialHolderName est facultatif.

### 5.7.6 Accès refusé (Access denied)

#### 5.7.6.1 Généralités

Un dispositif doit fournir un des événements conformément à 5.7.6.2, 5.7.6.3 et 5.7.6.4 pour tout refus d'accès à une personne.

Le paramètre Raison doit comporter la raison du refus. Le présent document définit les chaînes suivantes qu'il convient qu'un dispositif utilise:

- **CredentialNotEnabled**
  Le dispositif doit fournir la raison ci-dessus lorsqu'un identifiant valide n'est pas activé ou a été désactivé, par exemple, en raison de la perte de l'identifiant, afin d'empêcher toute entrée non autorisée.
- **CredentialNotActive**
  Le dispositif doit fournir la raison ci-dessus à chaque présentation d'un identifiant valide, bien que non encore actif, par exemple, présentation de l'identifiant avant la date de démarrage.
- **CredentialExpired**
  Le dispositif doit fournir la raison ci-dessus à chaque présentation d'un identifiant valide après sa date de fin de validité.
- **InvalidPIN**
  Le dispositif doit fournir l'événement la raison ci-dessus lorsqu'un code PIN saisi ne correspond pas à l'identifiant.
- **NotPermittedAtThisTime**
  Le dispositif doit fournir la raison ci-dessus à chaque refus d'accès d'un identifiant valide à l'instance AccessPoint demandée, du fait de la non-autorisationmomentanée de l'identifiant.
- **Non autorisé (Unauthorized)**
  Le dispositif doit fournir la raison ci-dessus à chaque défaut d'autorisation de l'identifiant présenté.
- **Autre (Other)**
  Le dispositif doit fournir la raison ci-dessus à chaque refus de la demande et lorsqu'aucun autre événement spécifique ne lui correspond, ou n'est pris en charge par le service.

Des valeurs supplémentaires peuvent être définies soit par des révisions futures du présent document, soit comme des extensions spécifiques au fournisseur. Pour permettre cette option, un client doit traiter les chaînes inconnues comme un paramètre ''Autre'.

Alors que les chaînes Raison définies à l'origine par le présent document n'utilisent pas une syntaxe de style QName, les extensions des valeurs Raison utilisent toujours une syntaxe de style QName. Le préfixe “pt” est réservé à une utilisation par l'ONVIF, c'est-à-dire "pt:<reason>". Il convient que les extensions spécifiques au fournisseur choisissent un préfixe approprié.

Même s'il existe plusieurs raisons à un refus, un dispositif doit transmettre une seule raison et le mode de sélection de la raison par le dispositif dans ce cas spécifique ne relève pas du domaine d'application du présent document.

#### 5.7.6.2 Anonyme

Il convient que le dispositif qui indique la prise en charge de la fonctionnalité AnonymousAccess pour une instance de point d'accès particulière fournisse l'événement suivant en cas de refus d'accès et en l'absence d'informations sur l'identifiant:

```
Topic: tns1:AccessControl/Denied/Anonymous

<tt:MessageDescriptionIsProperty="false">
```

```
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

Si le déclenchement de la commande est dû à la commande ExternalAuthorization, l'élément de données External doit être défini sur true (vrai); à défaut, l'élément est facultatif.

#### 5.7.6.3 Identifiant

Lorsque le dispositif refuse l'accès et que l'identifiant peut être reconnu, il doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/Denied/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="External"
                          Type="xs:boolean"/>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

Si le déclenchement de la commande est dû à la commande ExternalAuthorization, l'élément de données Externe doit être défini sur true (vrai); à défaut, l'élément est facultatif.

L'élément de données CredentialHolderName est facultatif.

#### 5.7.6.4 CredentialNotFound

##### 5.7.6.4.1 Généralités

Dans certaines situations, un dispositif peut ne pas être capable de résoudre les données d'authentification d'un jeton d'identifiant. Chaque fois que cela se produit, il convient que le dispositif fournisse un message d'événement correspondant conformément à 5.7.6.4.2.

La charge utile du message peut différer selon les facteurs d'authentification utilisés. Actuellement, un seul sous-ensemble unique est défini.

##### 5.7.6.4.2 Carte (Card)

À chaque défaut de correspondance entre l'identifiant et la demande archivée dans le dispositif, ce dernier doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/Denied/CredentialNotFound/Card

<tt:MessageDescriptionIsProperty="false">
```

```
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="Card" Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

Le contenu de la chaîne Carte est spécifique au fournisseur. Il peut contenir la chaîne d'identification complète de la carte, une partie de ces informations ou rester vide.

### 5.7.7 Agression (Duress)

Le dispositif qui indique la prise en charge de la fonctionnalité Agression pour une instance de point d'accès particulière doit fournir l'événement suivant à chaque détection d'une condition d'agression.

```
Topic: tns1:AccessControl/Duress

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
<tt:SimpleItemDescription Name="Reason"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

Les paramètres de données CredentialToken et CredentialHolderName sont facultatifs et peuvent être omis pour un accès anonyme.

### 5.7.8 Autorisation externe (Neutralisation)

#### 5.7.8.1 Généralités

Le dispositif qui indique la prise en charge de la fonctionnalité ExternalAuthorization pour une instance de point d'accès particulière doit fournir les événements définis dans 5.7.8 à chacune de ses demandes d'autorisation externe. Ces messages de notification doivent être utilisés conjointement aux opérations de service de contrôle d'accès correspondantes qui fournissent un retour d'information au dispositif.

#### 5.7.8.2 Anonyme

Chaque fois qu'un dispositif qui indique une fonctionnalité AnonymousAccess pour une instance de point d'accès particulière demande à un agent externe d'autoriser l'accès à une personne en l'absence d'information sur l'identifiant, par exemple, lors de la saisie de l'entrée REX et lorsque la confirmation de l'opérateur est nécessaire, il doit fournir l'événement suivant:

```
Topic: tns1:AccessControl/Request/Anonymous
<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.8.3 Identifiant

Chaque fois que le dispositif demande à un agent externe d'autoriser l'accès à une personne, il doit transmettre l'événement suivant:

```
Topic: tns1:AccessControl/Request/Credential

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="CredentialToken"
                          Type="pt:ReferenceToken"/>
<tt:SimpleItemDescription Name="CredentialHolderName"
                          Type="xs:string"/>
</tt:Data>
</tt:MessageDescription>
```

L'élément de données CredentialHolderName est facultatif et peut être omis ou une chaîne vide peut être utilisée si cet élément ne peut être résolu.

#### 5.7.8.4 Temporisation (Timeout)

Un dispositif doit fournir l'événement suivant, à chaque temporisation d'une demande d'autorisation externe:

```
Topic: tns1:AccessControl/Request/Timeout

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken"
                          Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.8.5 Exemple

Un client qui met en œuvre la prise en charge de l'autorisation externe tient compte généralement des événements AccessControl/Request/<subtopic>. Ces événements, lorsqu'ils surviennent, sont évalués. En cas d'autorisation ou de refus d'accès, la commande ExternalAuthorization est appelée sur le dispositif.

### 5.7.9 Modifications d'état

#### 5.7.9.1 Généralités

Le dispositif doit fournir les événements de modification d'état afin d'informer les clients abonnés de la modification de l'état de l'entité EACS. Un dispositif doit utiliser les rubriques définies dans 5.7.9 et associées à la description de message respective.

#### 5.7.9.2 Point d'accès (côté accès contrôlé)

Le dispositif qui indique la prise en charge de la fonctionnalité DisableAccessPoint pour une instance de point d'accès particulière doit fournir l'événement suivant à chaque modification de l'état, activé ou désactivé, de ce point d'accès:

```
Topic: tns1:AccessPoint/State/Enabled

<tt:MessageDescriptionIsProperty="true">
<tt:Source>
```

```
<tt:SimpleItemDescription Name="AccessPointToken"
Type="pt:ReferenceToken"/>
</tt:Source>
<tt:Data>
<tt:SimpleItemDescription Name="State" Type="xs:boolean"/>
</tt:Data>
</tt:MessageDescription>
```

### 5.7.10 Modifications de configuration

#### 5.7.10.1 Généralités

À chaque modification, ajout ou suppression des données de configuration, un dispositif doit fournir ces événements afin d'informer les clients abonnés.

#### 5.7.10.2 Point d'accès (côté accès contrôlé)

À chaque modification des données de configuration importantes pour une instance de point d'accès ou à chaque ajout de cette instance, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Configuration/AccessPoint/Changed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

À chaque suppression de point d'accès, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Configuration/AccessPoint/Removed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AccessPointToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

#### 5.7.10.3 Zone

À chaque modification des données de configuration pour une zone ou à chaque ajout de zone, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Configuration/Area/Changed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AreaToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

À chaque suppression d'une zone, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Configuration/Area/Removed

<tt:MessageDescriptionIsProperty="false">
<tt:Source>
<tt:SimpleItemDescription Name="AreaToken" Type="pt:ReferenceToken"/>
</tt:Source>
</tt:MessageDescription>
```

# 6 Commande de portes (point d'accès)

## 6.1 Généralités

Ce service propose des commandes permettant de récupérer les informations d'état et de commander les instances de porte d'un dispositif.

## 6.2 Fonctionnalités de service

### 6.2.1 Généralités

Un dispositif doit fournir des fonctionnalités de service de deux manières différentes:

1) Avec la méthode GetServices du service Dispositif (Device) lorsque IncludeCapability est définie sur true (vrai). Se reporter à l'IEC 68839-11-31 pour de plus amples informations.
2) Avec la méthode GetServiceCapabilities.

### 6.2.2 Structures de données: ServiceCapabilities

La structure ServiceCapabilities reflète la fonctionnalité facultative d'un service. Les informations sont statiques et ne varient pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

- **MaxLimit**
  Nombre maximal d'entités renvoyées par une demande Get<Entity>List ou Get<Entity>request unique. Le dispositif ne doit jamais renvoyer dans une réponse unique un nombre d'entités plus important que ce nombre.

### 6.2.3 Commande GetServiceCapabilities

Cette opération renvoie les fonctionnalités du service de commande de portes. Un dispositif doit prendre en charge cette commande comme décrit dans le Tableau 10.

**Tableau 10 – Commande GetServiceCapabilities**

| **GetServiceCapabilities** | Classe d'accès: PRE_AUTH |
|---|---|
| **Nom de message** | **Description** |
| GetServiceCapabilitiesRequest | *Ce message doit être vide* |
| GetServiceCapabilitiesResponse | *Ce message contient:*<br>• *"Capabilities": Le message de réponse de fonctionnalité contient les fonctionnalités de service demandées de contrôle d'accès qui utilisent une structure de fonctionnalité XML hiérarchique.*<br>tac:ServiceCapabilities **Capabilities [1][1]** |
| **Codes de défaut** | **Description** |
| | *Pas de défauts spécifiques à la commande* |

## 6.3 Informations concernant les portes (point d'accès)

### 6.3.1 Structures de données

#### 6.3.1.1 DoorInfo

Le type DoorInfo représente la porte comme un objet physique. La structure contient les informations et les fonctionnalités d'une instance de porte spécifique. Un dispositif doit fournir les champs suivants pour chaque instance de porte:

- **Jeton**
  Identificateur unique au service de la porte.

- **Nom**
  Nom lisible par l'utilisateur. Il doit être composé de 64 caractères au plus.
- **Fonctionnalités**
  Fonctionnalités de la porte, de type DoorCapabilities.

Afin de fournir de plus amples informations, le dispositif peut inclure le champ facultatif suivant:

- **Description**
  Description lisible par l'utilisateur. Elle doit être composée de 1 024 caractères au plus.

#### 6.3.1.2 DoorCapabilities

Les fonctionnalités DoorCapabilities reflètent la fonctionnalité facultative d'une entité physique particulière. Des instances de porte différentes peuvent avoir différents ensembles de fonctionnalités. Ces informations peuvent varier lors du fonctionnement du dispositif, par exemple, en cas de modification des paramètres du matériel. Les fonctionnalités suivantes sont disponibles:

- **Accès (Access)**
  Indique si cette instance de porte prend en charge la commande AccessDoor pour un accès momentané.
- **AccessTimingOverride**
  Indique que cette instance de porte prend en charge la neutralisation de la temporisation configurée dans la commande AccessDoor.
- **Verrouillage (Lock)**
  Indique que cette instance de porte prend en charge la commande LockDoor pour verrouiller la porte.
- **Déverrouillage (Unlock)**
  Indique que cette instance de porte prend en charge la commande UnLockDoor pour déverrouiller la porte.
- **Blocage (Block)**
  Indique que cette instance de porte prend en charge la commande BlockDoor pour bloquer la porte.
- **Doublelock**
  Indique que cette instance de porte prend en charge la commande DoubleLockDoor pour verrouiller plusieurs verrous de la porte.
- **LockDown**
  Indique que cette instance de porte prend en charge les commandes LockDown (et LockDownRelease) pour verrouiller la porte et la mettre en mode LockedDown.
- **LockOpen**
  Indique que cette instance de porte prend en charge les commandes LockOpen (et LockOpenRelease) pour déverrouiller la porte et la mettre en mode LockedOpen.
- **DoorMonitor**
  Indique que cette instance de porte comporte un DoorMonitor et prend en charge l'événement DoorPhysicalState.
- **LockMonitor**
  Indique que cette instance de porte comporte un LockMonitor et prend en charge l'événement LockPhysicalState.
- **DoubleLockMonitor**
  Indique que cette instance de porte comporte un DoubleLockMonitor et prend en charge l'événement DoubleLockPhysicalState.
- **Alarme (Alarm)**
  Indique que cette instance de porte prend en charge l'alarme de porte et l'événement DoorAlarm.

- **Violation (Tamper)**
  Indique que cette instance de porte comporte un détecteur de violation et prend en charge l'événement DoorTamper.
- **Défaut (Fault)**
  Indique que cette instance de porte prend en charge tout défaut de porte et l'événement DoorFault.

### 6.3.2 Commande GetDoorInfoList

Cette opération demande une liste de tous les éléments DoorInfo fournis par le dispositif. Un dispositif doit prendre en charge cette méthode comme décrit dans le Tableau 11.

Un appel à cette méthode doit renvoyer un paramètre StartReference lorsque les données ne sont pas toutes renvoyées et lorsqu'un nombre de données plus important est disponible. La référence doit être valide pour pouvoir récupérer l'ensemble de données suivant. Se reporter à 4.7.3 pour de plus amples informations.

Le nombre d'éléments renvoyés ne doit pas être supérieur au paramètre Limite.

**Tableau 11 – Commande GetDoorInfoList**

| **GetDoorInfoList** | Classe d'accès: READ_SYSTEM |
|---|---|
| **Nom de message** | **Description** |
| GetDoorInfoListRequest | *Ce message contient:*<br>• *"Limit": Nombre maximal d'entrées à renvoyer. Si la Limite est omise ou si la valeur de Limite est supérieure à ce que le dispositif prend en charge, ce dernier doit alors renvoyer son nombre maximal d'entrées.*<br>• *"StartReference": Commencer le renvoi des entrées à partir de cette référence de démarrage. En l'absence de spécification, les entrées doivent commencer du début de l'ensemble de données.*<br>xs:int **Limit [0][1]**<br>xs:string **StartReference [0][1]** |
| GetDoorInfoListResponse | *Ce message contient:*<br>• *"NextStartReference": StartReference à utiliser dans le prochain appel pour obtenir les éléments suivants. En son absence, ne pas obtenir d'autres éléments.*<br>• *"DoorInfo": Liste des éléments DoorInfo.*<br>xs:string **NextStartReference [0][1]**<br>tdc:DoorInfo **DoorInfo [0][non limité]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:InvalidStartReference | *StartReference invalide ou temporisée. Il est nécessaire que le client commence la récupération du début.* |

### 6.3.3 Commande GetDoorInfo

Cette opération demande une liste d'éléments DoorInfo correspondant aux jetons donnés. Un dispositif qui fournit le service Commande de porte doit prendre en charge cette méthode comme décrit dans le Tableau 12.

Le dispositif doit ignorer les jetons qu'il ne peut pas résoudre et peut renvoyer une liste vide si aucune entité Porte ne correspond aux jetons spécifiés.

Si le nombre d'éléments demandés est supérieur à la fonctionnalité MaxLimit, une instance de défaut TooManyItems doit être renvoyée.

**Tableau 12 – Commande GetDoorInfo**

| **GetDoorInfo** | Classe d'accès: READ_SYSTEM |
|---|---|
| **Nom de message** | **Description** |
| GetDoorInfoRequest | *Ce message contient:*<br>• *"Token": Jetons des éléments DoorInfo à obtenir.*<br>pt:ReferenceToken **Token [1][non limité]** |
| GetDoorInfoResponse | *Ce message contient:*<br>• *"DoorInfo": Liste des éléments DoorInfo.*<br>tdc:DoorInfo **DoorInfo [0][non limité]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgs<br>ter:TooManyItems | *Demande d'un trop grand nombre d'éléments, voir fonctionnalité MaxLimit.* |

## 6.4 État de l'entité Porte (point d'accès)

### 6.4.1 Généralités

L'état de la porte peut être affecté par un certain nombre d'opérations qui peuvent être réalisées sur cette dernière selon ses fonctionnalités: LockDoor, UnlockDoor, AccessDoor, BlockDoor, DoubleLockDoor, LockDownDoor, LockDownReleaseDoor, LockOpenDoor et LockOpenReleaseDoor.

### 6.4.2 Structures de données

#### 6.4.2.1 DoorState

La structure DoorState contient l'état d'exécution cumulé réel de la porte.

Les champs suivants sont disponibles:

- **DoorPhysicalState**
  État physique de la porte, de type DoorPhysicalState. Un dispositif qui indique la prise en charge de la fonctionnalité DoorMonitor pour une instance de porte particulière doit fournir ce champ.
- **LockPhysicalState**
  État physique du dispositif de verrouillage de porte, de type LockPhysicalState. Un dispositif qui indique la prise en charge de la fonctionnalité LockMonitor pour une instance de porte particulière doit fournir ce champ.
- **DoubleLockPhysicalState**
  État physique du dispositif de double verrouillage de porte, de type LockPhysicalState. Un dispositif qui indique la prise en charge de la fonctionnalité DoubleLockMonitor pour une instance de porte particulière doit fournir ce champ.
- **Alarme**
  État d'alarme de la porte, de type DoorAlarmState. Un dispositif qui indique la prise en charge de la fonctionnalité Alarme pour une instance de porte particulière doit fournir ce champ.
- **Violation**
  État de violation de la porte, de type DoorTamper. Un dispositif qui indique la prise en charge de la fonctionnalité Violation pour une instance de porte particulière doit fournir ce champ.
- **Défaut**
  Informations concernant les défauts de la porte, de type DoorFault. Un dispositif qui

indique la prise en charge de la fonctionnalité Défaut pour une instance de porte particulière doit fournir ce champ.

- **DoorMode**
  Mode de fonctionnement logique de la porte, de type DoorMode. Un dispositif doit consigner le mode de fonctionnement réel dans ce champ.

Les types de données suivants définissent les états des éléments DoorState.

#### 6.4.2.2 Énumération: DoorPhysicalState

État physique d'une porte. Les valeurs suivantes sont disponibles:

- **Inconnue (Unknown)**
  La valeur est actuellement inconnue, vraisemblablement en raison de l'initialisation ou du résultat non concluant fourni par les dispositifs de contrôle.
- **Ouverte (Open)**
  La porte est ouverte.
- **Fermée (Closed)**
  La porte est fermée.
- **Défaut**
  Détection d'un défaut du dispositif de contrôle de porte.

#### 6.4.2.3 Énumération: LockPhysicalState

État physique d'un dispositif de verrouillage, y compris un dispositif de double verrouillage. Les valeurs suivantes sont disponibles:

- **Inconnue**
  La valeur n'est actuellement pas connue.
- **Verrouillé (Locked)**
  Le dispositif de verrouillage est activé.
- **Déverrouillé (Unlocked)**
  Le dispositif de verrouillage n'est pas activé.
- **Défaut**
  Détection d'un défaut du dispositif de verrouillage.

#### 6.4.2.4 Énumération: DoorAlarmState

Décrit l'état d'une porte par rapport aux alarmes. Les valeurs suivantes sont disponibles:

- **Normal**
  Aucune alarme.
- **DoorForcedOpen**
  L'ouverture de la porte est forcée.
- **DoorOpenTooLong**
  Le maintien de la porte en position ouverte est trop long.

#### 6.4.2.5 DoorTamper

Informations concernant la violation d'une porte. Les champs suivants sont disponibles:

- **Raison**
  Champ facultatif; description détaillée de la modification de l'état de violation, par exemple, raison, lieu et heure.

NOTE Tous les champs, y compris celui-ci, conçus pour fournir des messages d'invite à l'intention de l'utilisateur final, peuvent être localisés selon la langue maternelle du client.

- **État**
  État du détecteur de violation, de type DoorTamperState.

#### 6.4.2.6 Énumération: DoorTamperState

Décrit l'état d'un détecteur de violation. Les valeurs suivantes sont disponibles:

- **Inconnue**
  La valeur n'est actuellement pas connue.
- **NotInTamper**
  Aucune violation n'est détectée.
- **TamperDetected**
  Violation détectée.

#### 6.4.2.7 DoorFault

Informations concernant les défauts d'une porte. Ceci peut être étendu à l'avenir par des attributs facultatifs. Les champs suivants sont disponibles:

- **Raison**
  Raison facultative d'un défaut.
- **État**
  État de défaut général de la porte, de type DoorFaultState. En cas de défauts avérés, la valeur doit être la suivante: FaultDetected. Les détails du défaut détecté doivent être indiqués dans le champ Raison, et/ou les différents champs DoorState et/ou les extensions de cette structure.

Ceci peut être étendu à l'avenir par des attributs facultatifs.

#### 6.4.2.8 Énumération: DoorFaultState

Décrit l'état d'un défaut de porte. Les valeurs suivantes sont disponibles:

- **Inconnu**
  L'état de défaut est inconnu.
- **NotInFault**
  Aucun défaut n'est détecté.
- **FaultDetected**
  Un défaut est détecté.

#### 6.4.2.9 Énumération: DoorMode

Les paramètres DoorMode décrivent le mode de fonctionnement d'un point de vue logique. Définir un mode de porte reflète l'objectif de définition d'une porte dans un état physique.

Les valeurs suivantes sont disponibles:

- **Inconnu**
  Le mode de fonctionnement est inconnu.
- **Verrouillé**
  L'objectif est de définir la porte à un état physique verrouillé. Dans ce mode, le dispositif doit permettre un accès momentané à l'aide de la méthode AccessDoor si celle-ci est prise en charge par l'instance Porte.
- **Déverrouillé**
  L'objectif est de définir la porte à un état physique déverrouillé. Ce mode masque les alarmes liées aux opérations de temporisation de porte telles qu'une ouverture trop longue ou une ouverture forcée.

- **Accès effectif (Accessed**)
  L'objectif est de définir momentanément la porte à un état physique déverrouillé. Après un temps prédéfini, le dispositif doit redéfinir la porte à son précédent mode. Ce mode masque les alarmes liées aux opérations de temporisation telles qu'une "ouverture de porte forcée".
- **Bloqué**
  L'objectif est de définir la porte à un état physique verrouillé et le dispositif ne doit pas autoriser les demandes AccessDoor, c'est-à-dire qu'il n'est pas possible que la porte soit dans le mode Accès effectif. Toutes les autres demandes de modification du mode de porte sont autorisées.
- **LockedDown**
  L'objectif est de définir la porte à un état physique verrouillé et le dispositif ne doit autoriser que la demande LockDownReleaseDoor. Aucune autre demande de modification du mode de porte n'est autorisée.
- **LockedOpen**
  L'objectif est de définir la porte à un état physique déverrouillé et le dispositif ne doit autoriser que la demande LockOpenReleaseDoor. Aucune autre demande de modification du mode de porte n'est autorisée.
- **DoubleLocked**
  L'objectif est de définir la porte avec plusieurs verrous à un état physique double verrouillé. Si la porte ne prend pas en charge le double verrouillage, les dispositifs doivent le traiter comme un mode verrouillé normal. Lors du passage d'un mode double verrouillage au mode déverrouillé, l'état physique de la porte peut tout d'abord passer à l'état verrouillé avant son déverrouillage.

### 6.4.3 Commande GetDoorState

Cette opération demande l'état d'une porte spécifié par l'entité Jeton. Un dispositif doit être capable de consigner l'état d'une porte en utilisant une structure DoorState proposée par la commande GetDoorState comme décrit dans le Tableau 13.

**Tableau 13 – Commande GetDoorState**

| **GetDoorState** | Classe d'accès: READ_SYSTEM_SENSITIVE |
|---|---|
| **Nom de message** | **Description** |
| GetDoorStateRequest | *Ce message contient:*<br>• *"Token": Jeton destiné à obtenir l'état de l'instance de porte.*<br>pt:ReferenceToken **Token [1][1]** |
| GetDoorStateResponse | *Ce message contient:*<br>• *"DoorState": État de la porte.*<br>tdc:DoorState **DoorState [1][1]** |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |

## 6.5 Commandes pour la commande de portes (point d'accès)

### 6.5.1 Généralités

Les commandes de contrôle de service contiennent les opérations qui permettent de modifier les états des instances de porte et de contrôler les instances de porte d'un dispositif comme décrit dans le Tableau 14.

### 6.5.2 Commande AccessDoor

Cette opération permet l'accès momentané à une porte. Elle est généralement appelée lorsqu'un détenteur de carte présente une carte à un lecteur installé sur la porte et qu'il obtient une autorisation d'accès.

Le mode de porte doit passer au mode Accès effectif. Se reporter à 6.4.2.9 pour de plus amples informations.

La porte doit rester accessible pendant la durée définie. À l'expiration de cette durée, le mode de porte doit revenir à son état précédent.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

Un dispositif qui indique la prise en charge de la fonctionnalité Accès pour une instance de porte particulière doit prendre en charge cette commande. Un dispositif qui indique la prise en charge de la fonctionnalité AccessTimingOverride pour une instance de porte particulière doit également fournir des paramètres de temporisation facultatifs (AccessTime, OpenTooLongTime et PreAlarmTime) lors de l'exécution de la commande AccessDoor.

Le dispositif doit adopter la meilleure approche possible pour les paramètres non pris en charge. Il doit revenir à la durée préconfigurée ou limiter la durée à la durée prise en charge la plus proche si la durée spécifiée se situe hors de toute limite.



**Tableau 14 – Commande AccessDoor**

| **AccessDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| AccessDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>• *"UseExtendedTime": Facultatif – Indique qu'il convient d'utiliser la durée étendue configurée.*<br>• *"AccessTime": Facultatif- neutralise AccessTime lorsque spécifié.*<br>• *"OpenTooLongTime": Facultatif- neutralise OpenTooLongTime lorsque spécifié (DOTL).*<br>• *"PreAlarmTime": Facultatif- neutralise PreAlarmTime lorsque spécifié.*<br>• *"Extension": Extension future.*<br><br>pt:ReferenceToken **Token [1][1]**<br>xs:boolean **UseExtendedTime [0][1]**<br>xs:duration **AccessTime [0][1]**<br>xs:duration **OpenTooLongTime [0][1]**<br>xs:duration **PreAlarmTime [0][1]**<br>tdc:AccessDoorExtension **Extension [0][1]** |
| AccessDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à l'état Accès effectif et du déverrouillage de la porte.* |

### 6.5.3 Commande LockDoor

Cette opération permet de verrouiller une porte comme décrit dans le Tableau 15. Le mode de porte doit passer à l'état Verrouillé. Se reporter à 6.4.2.9 pour de plus amples informations.

Un dispositif qui indique la prise en charge de la fonctionnalité Verrouillage pour une instance de porte particulière doit prendre en charge cette méthode.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. A noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 15 – Commande LockDoor**

| **LockDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| LockDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| LockDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à l'état Verrouillé.* |

### 6.5.4 Commande UnlockDoor

Cette opération permet de déverrouiller une porte. Le mode de porte doit passer au mode Déverrouillé. Se reporter à 6.4.2.9 pour de plus amples informations.

Un dispositif qui indique la prise en charge de la fonctionnalité Déverrouillage pour une instance de porte particulière doit prendre en charge cette méthode comme décrit dans le Tableau 16.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 16 – Commande UnlockDoor**

| **UnlockDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| UnlockDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| UnlockDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à l'état Déverrouillé.* |

### 6.5.5 Commande BlockDoor

Cette opération permet le blocage d'une porte et empêche tout accès momentané (au moyen de la commande AccessDoor). Le mode de porte doit passer au mode Bloqué. Se reporter à 6.4.2.9 pour de plus amples informations.

Un dispositif qui indique la prise en charge de la fonctionnalité Blocage (Block) pour une instance de porte particulière doit prendre en charge cette méthode comme décrit dans le Tableau 17.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 17 – Commande BlockDoor**

| **BlockDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| BlockDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| BlockDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à l'état Bloqué.* |

### 6.5.6 Commande LockDownDoor

Cette opération permet le verrouillage et empêche d'autres actions jusqu'à l'appel d'une commande LockDownRelease. Le mode de porte doit passer au mode LockedDown. Se reporter à 6.4.2.9 pour de plus amples informations.

Le dispositif doit ignorer les autres commandes de commande de portes jusqu'à l'exécution d'une commande LockDownRelease.

Un dispositif qui indique la prise en charge de la fonctionnalité LockDown pour une instance de porte particulière doit prendre en charge cette commande comme décrit dans le Tableau 18.

Si un dispositif prend en charge la fonctionnalité DoubleLock pour une instance de porte particulière, cette opération peut également avoir lieu.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 18 – Commande LockDownDoor**

| **LockDownDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| LockDownDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| LockDownDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à un état LockedDown.* |

### 6.5.7 Commande LockDownReleaseDoor

Cette opération permet la libération du mode LockedDown d'une porte. Le mode de porte doit revenir à son état précédent/suivant. La commande ne définit pas ce que doit être l'état précédent/suivant, mais il s'agit généralement du mode Verrouillé.

Un dispositif qui indique la prise en charge de la fonctionnalité LockDown pour une instance Porte particulière doit prendre en charge cette commande comme décrit dans le Tableau 19.

Cette méthode doit uniquement s'avérer satisfaisante si le mode DoorMode réel est LockedDown.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 19 – Commande LockDownReleaseDoor**

| **LockDownReleaseDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| LockDownReleaseDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| LockDownReleaseDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec de la sortie de l'état LockedDown.* |

### 6.5.8 Commande LockOpenDoor

Cette opération permet le déverrouillage d'une porte et empêche d'autres actions jusqu'à l'appel de la méthode LockOpenRelease. Le mode de porte doit passer au mode verrouillage en position ouverte (LockedOpen). Se reporter à 6.4.2.9 pour de plus amples informations.

Le dispositif doit ignorer les autres commandes de commande de portes jusqu'à l'exécution d'une commande LockOpenRelease.

Un dispositif qui indique la prise en charge de la fonctionnalité LockOpen pour une instance de porte particulière doit prendre en charge cette commande comme décrit dans le Tableau 20.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 20 – Commande LockOpenDoor**

| **LockOpenDoor** | Classe d'accès: ACTUATE |
|---|---|
| **Nom de message** | **Description** |
| LockOpenDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| LockOpenDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à l'état LockedOpen.* |

### 6.5.9 Commande LockOpenReleaseDoor

Cette opération permet la libération du mode LockedOpen d'une porte. Le mode de porte doit quitter le mode verrouillage en position ouverte et revenir à son mode précédent/suivant. La commande ne définit pas ce que doit être le mode précédent/suivant, mais il s'agit généralement du mode Déverrouillé.

Un dispositif qui indique la prise en charge de la fonctionnalité LockOpen pour une instance de porte particulière doit prendre en charge cette commande comme décrit dans le Tableau 21.

Cette méthode doit uniquement s'avérer satisfaisante si le mode de porte réel est LockedOpen.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 21 – Commande LockOpenReleaseDoor**

| **LockOpenReleaseDoor** | **Classe d'accès: ACTUATE** |
|---|---|
| **Nom de message** | **Description** |
| LockOpenReleaseDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| LockOpenReleaseDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec de la sortie de l'état LockedOpen.* |

### 6.5.10 Commande DoubleLockDoor

Cette opération permet le verrouillage sécurisé d'une porte. Un appel de cette méthode doit modifier le mode de porte en état Double verrouillage. Se reporter à 6.4.2.9 pour de plus amples informations.

Un dispositif qui indique la prise en charge de la fonctionnalité DoubleLock pour une instance de porte particulière doit prendre en charge cette commande comme décrit dans le Tableau 22. À défaut, cette méthode peut être appliquée comme une opération de Verrouillage normale, voir 6.5.3.

Si la porte comporte un dispositif de verrouillage supplémentaire, il doit être également verrouillé.

Si la définition du mode de fonctionnement échoue, un mode Défaut de type défaillance doit être renvoyé. À noter que si l'état physique de la porte ne peut être défini, cela est indiqué par des événements.

**Tableau 22 – Commande DoubleLockDoor**

| **DoubleLockDoor** | **Classe d'accès: ACTUATE** |
|---|---|
| **Nom de message** | **Description** |
| DoubleLockDoorRequest | *Ce message contient:*<br>• *"Token": Jeton de l'instance de porte à commander.*<br>pt:ReferenceToken **Token [1][1]** |
| DoubleLockDoorResponse | *Ce message est généralement vide, mais est toutefois extensible* |
| **Codes de défaut** | **Description** |
| env:Sender<br>ter:InvalidArgVal<br>ter:NotFound | *Jeton spécifié non trouvé.* |
| env:Receiver<br>ter:Action<br>ter:Failure | *Échec du passage à l'état DoubleLocked.* |

## 6.6 Rubriques de notification

### 6.6.1 Généralités

Ce Le Paragraphe 6.6 définit les rubriques de notification spécifiques au service Commande de portes.

### 6.6.2 Modifications d'état

À chaque modification d'un mode de porte, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Door/State/DoorMode

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorMode"/>
  </tt:Data>
</tt:MessageDescription>
```

Un dispositif qui indique la prise en charge de la fonctionnalité DoorMonitor pour une instance de porte particulière doit fournir l'événement suivant à chaque modification de l'état physique de cette porte:

```
Topic: tns1:Door/State/DoorPhysicalState

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorPhysicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

Un dispositif qui indique la prise en charge de la fonctionnalité LockMonitor pour une instance de porte particulière doit fournir l'événement suivant à chaque modification de l'état physique du dispositif de verrouillage de cette porte:

```
Topic: tns1:Door/State/LockPhysicalState

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:LockPhysicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

Un dispositif qui indique la prise en charge de la fonctionnalité DoubleLockMonitor pour une instance de porte particulière doit fournir l'événement suivant à chaque modification de l'état physique du dispositif de verrouillage sécurisé de cette porte:

```
Topic: tns1:Door/State/DoubleLockPhysicalState

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:LockPhysicalState"/>
```

```
  </tt:Data>
</tt:MessageDescription>
```

Un dispositif qui indique la prise en charge de la fonctionnalité Alarme pour une instance de porte particulière doit fournir l'événement suivant à chaque modification de l'état d'alarme de cette porte:

```
Topic: tns1:Door/State/DoorAlarm

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorAlarmState"/>
  </tt:Data>
</tt:MessageDescription>
```

Un dispositif qui indique la prise en charge de la fonctionnalité Violation pour une instance de porte particulière doit fournir l'événement suivant à chaque modification de l'état de violation de cette porte:

```
Topic: tns1:Door/State/DoorTamper

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorTamperState"/>
  </tt:Data>
</tt:MessageDescription>
```

Un dispositif qui indique la prise en charge de la fonctionnalité Défaut pour une instance de porte particulière doit fournir l'événement suivant à chaque modification de l'état de défaut de cette porte:

```
Topic: tns1:Door/State/DoorFault

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="tdc:DoorFaultState"/>
    <tt:SimpleItemDescription Name="Reason" Type="xs:string"/>
  </tt:Data>
</tt:MessageDescription>
```

L'élément Raison peut être vide ou absent. Le dispositif peut également ne pas le prendre en compte à moins que l'état de défaut soit FaultDetected.

### 6.6.3 Modifications de configuration

À chaque modification des données de configuration pour une porte ou à chaque ajout d'une porte, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Configuration/Door/Changed

<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>
```

```
</tt:MessageDescription>
```

À chaque suppression d'une porte, le dispositif doit fournir l'événement suivant:

```
Topic: tns1:Configuration/Door/Removed

<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="DoorToken" Type="pt:ReferenceToken"/>
  </tt:Source>

</tt:MessageDescription>
```

# Annexe A
(normative)

# Schéma XML d'interface de contrôle d'accès

## A.1 WSDL de service de contrôle d'accès

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tac="http://www.onvif.org/ver10/accesscontrol/wsdl" name="PACSService"
targetNamespace="http://www.onvif.org/ver10/accesscontrol/wsdl">
  <wsdl:types>
    <xs:schema xmlns:pt="http://www.onvif.org/ver10/pacs"
targetNamespace="http://www.onvif.org/ver10/accesscontrol/wsdl" elementFormDefault="qualified"
version="1.0">
      <xs:import namespace="http://www.onvif.org/ver10/pacs" schemaLocation="types.xsd"/>
      <!--====== types ======-->
      <xs:complexType name="ServiceCapabilities">
        <xs:sequence>
          <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
        </xs:sequence>
        <xs:attribute name="MaxLimit" type="xs:unsignedInt" use="required"/>
        <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <!--================================-->
      <xs:complexType name="AccessPointInfoBase">
        <xs:complexContent>
          <xs:extension base="pt:DataEntity">
            <xs:sequence>
              <xs:element name="Name" type="pt:Name"/>
              <xs:element name="Description" type="pt:Description" minOccurs="0"/>
              <xs:element name="AreaFrom" type="pt:ReferenceToken" minOccurs="0"/>
              <xs:element name="AreaTo" type="pt:ReferenceToken" minOccurs="0"/>
              <xs:element name="EntityType" type="xs:QName" minOccurs="0"/>
              <xs:element name="Entity" type="pt:ReferenceToken"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <!--================================-->
      <xs:complexType name="AccessPointInfo">
        <xs:complexContent>
          <xs:extension base="tac:AccessPointInfoBase">
            <xs:sequence>
              <xs:element name="Capabilities" type="tac:AccessPointCapabilities"/>
              <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <!--================================-->
      <xs:complexType name="AccessPointCapabilities">
        <xs:sequence>
          <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
```

```
    </xs:sequence>
    <xs:attribute name="DisableAccessPoint" type="xs:boolean" use="required"/>
    <xs:attribute name="Duress" type="xs:boolean"/>
    <xs:attribute name="AnonymousAccess" type="xs:boolean"/>
    <xs:attribute name="AccessTaken" type="xs:boolean"/>
    <xs:attribute name="ExternalAuthorization" type="xs:boolean"/>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="AreaInfoBase">
    <xs:complexContent>
      <xs:extension base="pt:DataEntity">
        <xs:sequence>
          <xs:element name="Name" type="pt:Name"/>
          <xs:element name="Description" type="pt:Description" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="AreaInfo">
    <xs:complexContent>
      <xs:extension base="tac:AreaInfoBase">
        <xs:sequence/>
        <xs:anyAttribute processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="AccessPointState">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean"/>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="Decision">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Granted"/>
      <xs:enumeration value="Denied"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="DenyReason">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CredentialNotEnabled"/>
      <xs:enumeration value="CredentialNotActive"/>
      <xs:enumeration value="CredentialExpired"/>
      <xs:enumeration value="InvalidPIN"/>
      <xs:enumeration value="NotPermittedAtThisTime"/>
      <xs:enumeration value="Unauthorized"/>
      <xs:enumeration value="Other"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <!--  Message Request / Response elements  -->
  <xs:element name="GetServiceCapabilities">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
```