PSIA Area Control Specification

Security Classification:	Protected
Version:	3.1
Revision:	Rev1
Control:	Uncontrolled when printed
Date	08/07/2019

Document History

Version	Date	Author	Description
0.1	07/22/10	Marine Drive	Created document, detailed services structure and resource requirements
0.2	08/04/10	Steve Howe, Rich Hinkson	Updated configuration items for all resources
0.3	09/14/10	Marine Drive	Updated events for PSIA CMEM spec
0.4	09/22/10	Marine Drive	SIA alignment – Glossary
0.5	10/26/10	Andy Bulkley	Detailed access control use cases
0.6	12/10/10	Rajesh Kumar TRS	Created and detailed access control specification for all resources and services
0.7	12/20/10	Marine Drive	Updated access control XML attributes and detailed resources
0.8	01/20/11	Rajesh Kumar TRS	Updated additional use cases and addressed review comments
0.9	06/10/11	Neelendra Bhandari	Added schema definition, updated XML and addressed review comments
0.9 Revision 1	07/18/11	Neelendra Bhandari	Re-arranged resource hierarchy to prevent repetition of common objects between intrusion and access control. Simplified schema for easier list and hierarchy management. Added URI and status flags under portals for overriding access control operations. Modified partition status info to have list of active alarms and troubles
0.9 Revision 2	08/02/11	Neelendra Bhandari	Modified events definition and added schema for events. Added GUID for common elements such as time schedules, holidays, permissions, credentials, credential holders etc. Replaced arm-disarm, mask-monitor pairs with arm, mask by supplying the type as QSP. Changed URI to */info to get configuration of various elements. Added addition and modification of holidays, time schedules and permissions
0.9 Revision 3	08/30/11	Neelendra Bhandari	Changed PortalIDList to PartitionMemberIDList in PrivilegeElement under Permission, split Class: System state alarm into state and device_state alarms; added Version attribute for XML elements; moved schema for schedule, holiday and permission out of Area Control xsd; AuthorityLevel element under permission now accepts values beyond 1 to 5; more reasons added to access.granted and access.denied types; added /PSIA/profile resource to section *Service Requirements*; added parent and child partitions configuration to partitions; removed list of alarms, troubles and faults from PartitionStatus (should be covered by CMEM)
0.9 Revision 4	10/28/11	Neelendra Bhandari	Modified schema of type EventData to allow reference to single or multiple Area Control entities; incorporated feedback from RaCM WG; updated Metadata section based on CMEM changes
1.0	11/10/11	Marine Drive Neelendra Bhandari Rajesh Kumar TRS	Incorporated feedback from Systems WG
1.0 r1	22/11/11	Neelendra Bhandari	Incorporated feedback from Video Analytics WG

1.0.r2	Jan 2013	Neelendra Bhandari	Incorporated updates collected ACWG change request spreadsheet
1.0.r3	Feb 12, 2013	Jeff Longo	Incorporated changes for Profiles into the specification
1.0.r4	Feb 25, 2013	Andy Bulkley	Accepting proposed changes from r2 and r3 for first nine sections
1.0.r5	Mar 1, 2013	Andy Bulkley	Changes from 2/28/2013 Area Control WG Conf Call 1. Section 10.1.2 Remove Diagram 2. 10.1.3 Remove diagram 3. InputStatus, added IsTampered example
1.0 r6	Mar 26, 2013	Andy Bulkley	Changes from 3/14/2013 Area Control WG Conf Call 1. Clarify example for AccessOverride Function 2. Add AreaID to diagrams
2.0 r01	October 24, 2013	Andy Bulkley	Add Appendix A - Access Control Profile Up Revision of document to 2.0
2.0.r02	November 6, 2013	Andy Bulkley	Updated Access Control profile, changed CSEC from BASIC to CORE
3.0r01	March 13 2014	Jeffrey Longo	Updated Access Control profile – CMEM to Access Control core profile via ticket 201, updated example in 11.1.1.1 per ticket 29 to comply with CMEM 1.1 schema
3.0r02	April 23, 2014	Ken Larson	1. Clarification of GETs that follow PUTs for status (It is not allowed to perform a GET on maskState, etc.) 2. Correct Outputs/instanceDescription/Devices to Outputs/instanceDescription
3.0r03	April 25, 2014	Ken Larson	1. Reference XSDs on PSIA website rather than embed in this document 2. No PulseData for accessOverride 3. Change PulseData to only have a duration in seconds 4. Changed LatchedPulse state to Pulse
3.0r04	May 1, 2014	Ken Larson	1. Remove paragraph indicating GMCH details to be provided in a subsequent revision 2. Remove AreaControl.Permission class from events 3. Clarify that credentialID 0 is to be used for access.denied events where the credentialID is unknown 4. Top level of RequiredIdentifierList is always LogicalOr
3.0r05	May 7, 2014	Ken Larson	1. Change/clarify details of MIDS to reference exact XML enum values 2. Change Credental and CredentialHolder states in event XML to match StateOfCredential and EnableState
3.0r06	May 20, 2014	Ken Larson	1. Change pulse duration from seconds to milliseconds
3.0r07	June 5, 2014	Jeffrey Longo	Updated RequiredIdentifierList per ticket 63
3.0r08	July 17, 2014	Jeffrey Longo	Updated Access Control Resources to include Partition information from PLAI concept of “presence zones.” Ticket 231.
3.0.r09	August 14, 2014	Ken Larson	1. Most references by local ID changed to use ReferenceID 2. Version attribute on RequiredIdentifierList 3. Clarify LogicalAnd/LogicalOr in RequiredIdentifierList 4. Add Name element in addition to Description, where allowed. 5. Add Unknown and Other to event states to match XSD

			6. Add connectionState events to Portal and Zone 7. Minor cleanup issues
3.0.r10	August 14, 2014	Ken Larson	1. Remove Supervised from OutputInfo 2. Remove *ListUrl elements
3.0r11	August 20, 2014	Ken Larson	1. Add Name to DescriptionInfo resources 2. Clarify panic vs duress
3.0r12	August 21, 2014	Ken Larson	1. Change PortalStatus and events to use PortalOpenState, PortalForcedState, PortalHeldState enumerations 2. Use event enums in PortalStatus, InputStatus, OutputStatus, ZoneStatus, PartitionStatus: ConnectionState, LatchState, InputState, OutputState, etc. 3. Split SystemState enum into SystemFaultState and MaintenanceState, and use in SystemStatus as well as events 4. Split PowerState enum into ACPowerState and BatteryPowerState, and use in SystemStatus/PowerStatus as well as events 5. Change IsArmed to ArmState enum in PartitionStatus, remove ArmType from ArmState, and add ArmType to EventValueState. Arming events now use both ArmState and ArmType. 6. Split IntrusionAlarmState into IntrusionAlarmState and IntrusionAlarmType, so that the former has Alarm/OK, and the latter has Intrusion, Fire, Panic, etc. Added IntrusionAlarmType to EventValueState, and intrusion alarm events now use both of these elements. Also added IntrusionAlarm and IntrusionAlarmType to PartitionStatusForIntrusion. 7. Change IsDeviceInAlarm for PortalStatus into PortalAlarm PortalAlarmState enum 8. Add Environmental as a type of alarm 9. Synchronize ZoneType and IntrusionAlarmType 10. Define forced and held in glossary 11. Clarify Name vs GivenName MiddleName Surname 12. Change “Released” to “Active” for InputState 13. Clarify fault conditions (intrusion vs input) 14. Split out maskState for Portal into forcedMaskState and heldMaskState 15. Clarify local source id to be Local ID of the Partition, Portal, Zone, Input, or Output 16. Create a new glossary section for terms common to access control and intrusion 17. Make ArmsLobby and DisarmsLobby into reference IDs 18. access.granted/duress not valid in MIDS type string, removed from example
3.0r13	August 22, 2014	Ken Larson	1. Move presence resources to same section as other Partition resources 2. Refactor presence list and count to have their own resources apart from info, use the term occupancy, allow credentials as well as holders, and make UID/GUID-agnostic 3. Clarify Maintenance in glossary 4. Clarify authority levels in glossary 5. Add ZoneArmingType, change example to have a Fire sensor 6. Replace SystemFaultState with TamperState for SystemStatus and related event

3.0r14	August 24, 2014	Ken Larson	1. Clarify that REX, door contact, and door strike are not Inputs/Outputs 2. Add (optional) battery/power status to Portal, Zone, Input, Output, including events. 3. Unify Comms and ConnectionState (use ConnectionState for SystemStatus) 4. Clarify user code, user authority level in glossary. Remove user number since it was unclear and referenced nowhere in the specification or XSD 5. Rename InputFaultState to SupervisionFaultState 6. General cleanup 7. Add Subscriber Number to glossary 8. Clarify that Connected means online, as opposed to "enabled".
3.0r15	August 27, 2014	Ken Larson	1. Rename SupervisionFaultState to InputSupervisionFaultState 2. Add output supervision
3.0r16	August 29, 2014	Ken Larson	1. Clarify that externalOnly is optional, and change default from true to false 2. Change state changing query parameters to required, eliminate defaults 3. Notate all query parameters as either [optional] or [required] 4. Make all 'state' query parameters consistent capitalization. A few were 'State' instead of 'state' 5. Clarify that occupant count and occupants list do not have to match exactly. 6. Move enums out of EventValueState and into MIDS type 7. Clarify that Inputs are allowed under Zones, but not required. 8. Add a general considerations section.
3.0r17	August 30, 2014	Ken Larson	1. Change PartitionMemberID to ZoneID, PortalID, and/or both. Change DeviceID to InputID, OutputID, and/or both 2. InputInfo, OutputInfo, InputStatus, OutputStatus able to be queried by portalID, zoneID, and/or partitionID 3. It is now allowed for a Zone and a Portal to have the same local ID. It is now allowed for an Input and an Output to have the same local ID 4. Document local/global IDs. 5. Change EnableState to StateOfCredentialHolder 6. Add externalOnly to InputStatus and OutputStatus queries, to be consistent with InputInfo and OutputInfo 7. Allow masking of all portals by partition 8. Allow LatchState and AccessOverrideState changes by partition
3.0r18	September 18, 2014	Jeffrey Longo	1. Updated document number from 2.0.1 to 3.0 due to magnitude of release 2. Ticket 296: Credentials and CredentialHolders should only be accessed via secure transport
3.0r19	September 29, 2014	Ken Larson	1. Ticket 241: Clarify sorting for range requests when global ID is used. 2. Ticket 279: add ValueEncoding for PIN and card number values
3.0r20	October 8, 2014	Jeffrey Longo	1. Updated examples from ticket 209. 2. Added resource /PSIA/AreaControl/CredentialFormat/info and info/<id>

			3. PLAI Profile with Functional Role Option 4. Added metata for assigning credentials and provided example 5. Provided metadata example for HostGrant option 6. Formatting
3.0r21	December 5, 2014	Ken Larson	1. Ticket 369 – Allow local ID of 0 to be a legitimate local ID, not reserved to mean “null”, “none”, “default”, or “outside” anymore. 2. Ticket 372 – pulse of Portal LatchState is for a single door cycle, not to unlock for an extended period of time
3.0r22	January 22,2015	Jeffrey Longo	1. Fix examples from updated XSD. 2. PLAI Raw Location option proposal.
3.0r23	February 23, 2015	Jeffrey Longo	1. Initial updates to reflect Topologies 2. Added /PSIA/AreaControl/Roles per PLAI
3.0r24	February 11, 2015	Praveen Jha	In Section 12.1 update the index as Optional
3.0 r25	March 3, 2015	Jeffrey Longo	1. Additional topology updates 2. Updates from CMEM 3.0
3.0 r26	March 12, 2015	Jeffrey Longo	3. Final Version 3 Modifications by group. Version Approved at Area Control WG meeting 3/12/15
3.0 r27	November 22, 2018	Kostas Papadopoulos	1. Added support for Raw Bits.
3.1 r01	July 8, 2019	Kostas Papadopoulos	1. Renamed CredentialHolderID to CredentialHolderUID 2. Added UID example with curly braces. 3. Renamed CredentialID to Credential UID 4. Added UID example with curly braces. 5. Added Email & DomainName in CredentialHolders. 6. Made curly braces to follow the windows registry standard mandatory for all guids. 7. Added two new fields, LastModifiedDate, CreationDate into CredentialInfo

Disclaimer

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING

ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, PSIA disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and PSIA disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

No license, express or implied, by estoppels or otherwise, to any PSIA or PSIA member intellectual property rights is granted herein.

Except that, a license is hereby granted by PSIA to copy and reproduce this specification for internal use only.

Contact the Physical Security Interoperability Alliance at info@psialliance.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

Table of Contents

1 Introduction

This document specifies an interface that enables access control and intrusion systems/devices to communicate with various security systems in a standard way. This eliminates the need for device driver customization in order to achieve interoperability among products from different manufacturers. The intent of this specification is to improve the interoperability of IP-based physical security products from different vendors.

This document is based on the PSIA Service Model specification. Some, but not all, of the information within the Service Model specification is repeated in this document for the sake of completeness. However, readers and implementers are expected to be familiar with the PSIA Service Model and the REST concepts that form the foundation of the PSIA's protocols.

Scope

This document is a contribution to the Physical Security Interoperability Alliance Area Control Working Group submitted under the PSIA Intellectual Property Policy for defining access control and intrusion system model operating in a shared network using standards based protocols.

The profiles defined in the Appendix of this document identify the resource and service requirements to conform to each profile in addition to identifying what PSIA Topology they conform to. Additional mandatory PSIA resources and services are defined in PSIA Service Model, Common Metadata and Event Model (CMEM), and Common Security Model (CSEC) specifications and are identified by the topology.

Problem Statement

This access control and intrusion system model addresses initial use cases outlined in the document PSIA ACWG-UC "Area Control Fundamental Use Cases" Revision 0.8 of 7/14/2009 and PSIA ACWG-UC "Access Control Fundamental Use Cases" Revision 0.1 of 10/25/2010. In this model, an area control system hosts the PSIA access or intrusion client application. As of version 2 of this specification, these use cases as well as additional related use cases are encapsulated into functional Profiles, which are defined later in this specification.

Conformance

The devices claiming conformance to any profile listed in this document will implement PSIA compliant web services and adhere to the PSIA Service Model, PSIA Common Security Model (CSEC), PSIA Common Metadata and Event Model specifications as defined in referenced documents below.

1.1 References

[RFC 1945]	"Hypertext Transfer Protocol -- HTTP/1.0", T. Berners-Lee et al, May 1996. URL:http://www.ietf.org/rfc/rfc1945.txt
[RFC 2068]	"Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et al, January 1997. URL:http://www.ietf.org/rfc/rfc2068.txt
[RFC 2616]	"Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding et al, June 1999. URL:http://www.ietf.org/rfc/rfc2616.txt
[RFC 2782]	"A DNS RR for specifying the location of services (DNS SRV)", A. Gulbrandsen et al, February 2000. URL:http://www.ietf.org/rfc/rfc2782.txt
PSIA Service Model specification	Service Model Version 2.0 r01, October 2013
PSIA Common Metadata and Event Model Specification	Common Metadata/Event Management Specification Version 2.0 r08, October 2013
PSIA Common Security Service Specification	Common Security Service Specification Version 2.0 r01.1, October 2013

Terminology used in this document

Terminology	Description
Intrusion System: Some Intrusion System terminology has been sourced from Security Industry Association, Glossary Project.	
Zone	A dedicated input to the control panel containing one or more initiating devices (or points) which will trip that input upon activation of any one initiating device. A zone may or may not be able to distinguish between which initiating device initiated the signal.
Partition	A defined area within the security system that can be armed and disarmed independently of the other area(s), but operated under a single system control (Dedicated or shared user interfaces may be used to operate a partition.)
Lobby Partition (Common Partition)	A common partition that is shared by users of other partitions in a building. This shared partition may be assigned as a "common lobby" partition for the system. This is typical for multi tenant facility where the common pathways and lobby are controlled by a common partition. This option employs logic for automatic arming and disarming of the common lobby. Partitions may be set to affect and/or attempt to arm the common lobby. This will affect the way the lobby will react when arming or disarming activity occurs in another partition.
User Authority Level	A user authority level determines what functions a user may perform when interacting with the system (using their User Code). User authority levels can vary from 1 – 5; 1 being highest (the upper limit of 5 is not enforced by this specification). Panel manufacturer can define the authority levels. Examples of authority levels defined by a manufacturer level could include: • **Installer** (Has access to all partitions and can configure panels completely.) • **Administrator** (Has access to all partitions and can create other users in the panel.) • **Operator** (Has access only to assigned partitions and has no privilege to configure.)
User	A User is a person who is allowed access to physical area which is controlled by an intrusion system. Depending on the authority level, the user may interact with the system to perform various actions (e.g. arm, disarm) Users may be assigned authority levels via their credentials. Note: To provide common area control services for access control and intrusion, the term 'Credential holder' is used interchangeably with 'User'.
User Code	The numeric sequence of digits that is used by a user to authenticate to/interact with the intrusion system.

	Note: To provide common area control services for access control and intrusion, the term 'PIN' is used interchangeably with 'User Code'. A 'PIN' is an identifier and is a part of a Credential..
Arming Station or User Interface	A means of manually controlling a security system which may also include indicating devices providing status information about the system. Also known as a Keypad. Generally, a valid User Code must be entered on the arming station to perform actions.
Trouble	A potentially problematic condition which is often lesser in severity than an Alarm. Trouble conditions can include communications problems, supervisory faults, power and battery problems, etc., occurring in the system, system accessories or connected wiring
Alarm	A condition requiring immediate action, such as an alarm initiated from an intrusion detector, door switch, or the like. Alarms typically are transmitted to a host or central station, and often sound an audible alarm sound.
Intrusion Fault	An electrical condition that compromises an indicating or initiating circuit, or an open circuit indicating that a zone is open This condition typically appears when a partition is in disarmed state. Not to be confused with an Input Supervision Fault, or Output Supervision Fault.
Arm	Arm means an action used to activate an intrusion detection system, or partition thereof. An armed system or partition goes into an alarm state when one or more Zones are tripped/activated. **Arming types:** **Arm Away or Full**- An armed state of an intrusion detection system where all zones and sensors are activated **Arm Stay or Home**- An armed state of a security system where some zones or sensors are active while other zones or sensors are made inactive, allowing occupants to be inside the protected premises without causing an alarm
Disarm	Disarm means an action used to deactivate a security system, or a part thereof, so that it will not generate intrusion alarms
Bypass	An operation to temporarily disable a point of protection (window, door, smoke detector etc.)
Input	An input is an optional device which can be a part of a Zone. In the event that a Zone combines multiple inputs, or lower-level status information associated with a Zone's (single or multiple) inputs, an implementation may find it useful to model inputs. There is no requirement for Inputs in the Intrusion profiles, however. An input may be associated with one or more Zones (or Portals). An input may be associated with a Partition.

Output	A device or appliance externally connected to a control unit that is employed to assure proper operation of a system or to provide supplementary initiation, signaling and/or annunciation. Examples of outputs are: annunciators, end-of-line resistors or diodes, auxiliary relays, remote switches, wired and wireless zone expanders, wireless transmitters (for linkage to remote annunciators/auxiliary relays) and the like. Also known as a control unit accessory. An output may be associated with one or more Zones (or Portals). An output may be associated with a Partition.
Exit Delay	The period of time allowed after arming a security system, to exit the premises through specified zones, without tripping an alarm
Entry Delay	The period of time allowed, after entry to the premises, to disarm the security system before the panel initiates an alarm transmission sequence
Duress Alarm	A type of alarm used to indicate that a person is being forced to do something against their will, which is generally reported to a host or central station, but does sound a local alarm.
Panic Alarm	A type of manual activation of an alarm which is generally reported to a host or central station, and also locally sounds an alarm.
Subscriber Number	A unique number which identifies the customer and location of an intrusion panel to a central monitoring station.

Access Control System:	
Partition (Access Area)	A physical area within an access control system, where entry and/or exit is regulated by access-controlled Portals. Credential holders present Credentials to Portals in order to enter and/or exit Partitions Also known as an “Access Area” or “Area”.
Permission (Access Level)	A Permission in access control allows Credentials physical access to Portals (and therefore Partitions). The Permission defines which Portals and Partitions are accessible, and during which Time Schedules. Also known as an “Access Level”
Credential Holder	A person, who, through assigned Credentials with Permissions, can have physical access to Partitions by entering through Portals.
Credential	Credential contains identification data such as card, PIN or biometric information containing encoded data that is read by an access control Portal. Access control system can use this credential information to authenticate the credential/holder and give them physical access. A person can hold one or more credentials with different access levels.

Time Schedule	A list of time intervals used to define the period in which access will be granted or restricted to a credential. May also be used to define which required identifiers must be presented (e.g. Card, PIN) at which times.
Portal	An access control point. Often a door, but can also be things like parking gates, turnstiles, etc. A point where access is authenticated by some means (access control reader, biometric device, etc). Commonly, it is a point of transition from one access control area (partition) to another. The portal may combine multiple components (reader, door contact, request to exit, door strike etc.). (Note that these components are not modeled as Inputs or Outputs in this specification) There may be directional information associated with transactions (entering area X, leaving area Y). However, excluding requests-to-exit, each Portal only goes one direction, that is, from one Partition to another. If a credential presentation is required for both directions, then this must be 2 Portals
Input	An auxiliary input device that is connected to an access control panel. The input can be monitored for state changes, and monitoring can be masked which prevents reporting of state changes. Note that hardware that is fully in service of a Portal, such as a request-to-exit (REX), door contact (door position switch) is not intended to be modeled as an Input in this specification. An input may be associated with one or more Portals (or Zones). An input may be associated with a Partition.
Output	An auxiliary output device connected to a panel. These are programmable relays/switches that can be used to perform many different functions. They can be used to turn lights on or off, to control sounders, or to indicate status. Note that hardware that is fully in service of a Portal, such as a door strike, is not intended to be modeled as an Output in this specification. An output may be associated with one or more Portals (or Zones). An output may be associated with a Partition.
Access Granted	A transaction that the access control panel generates if all access decision criteria are positively met, and the Credential/holder has been allowed through the Portal
Access Denied	A transaction that the access control panel generates if not all access decision criteria are positively met, and the Credential/holder is not to be allowed through the Portal.
Pending Access	A transaction that the access control panel generates if the panel needs an external access decision to be made

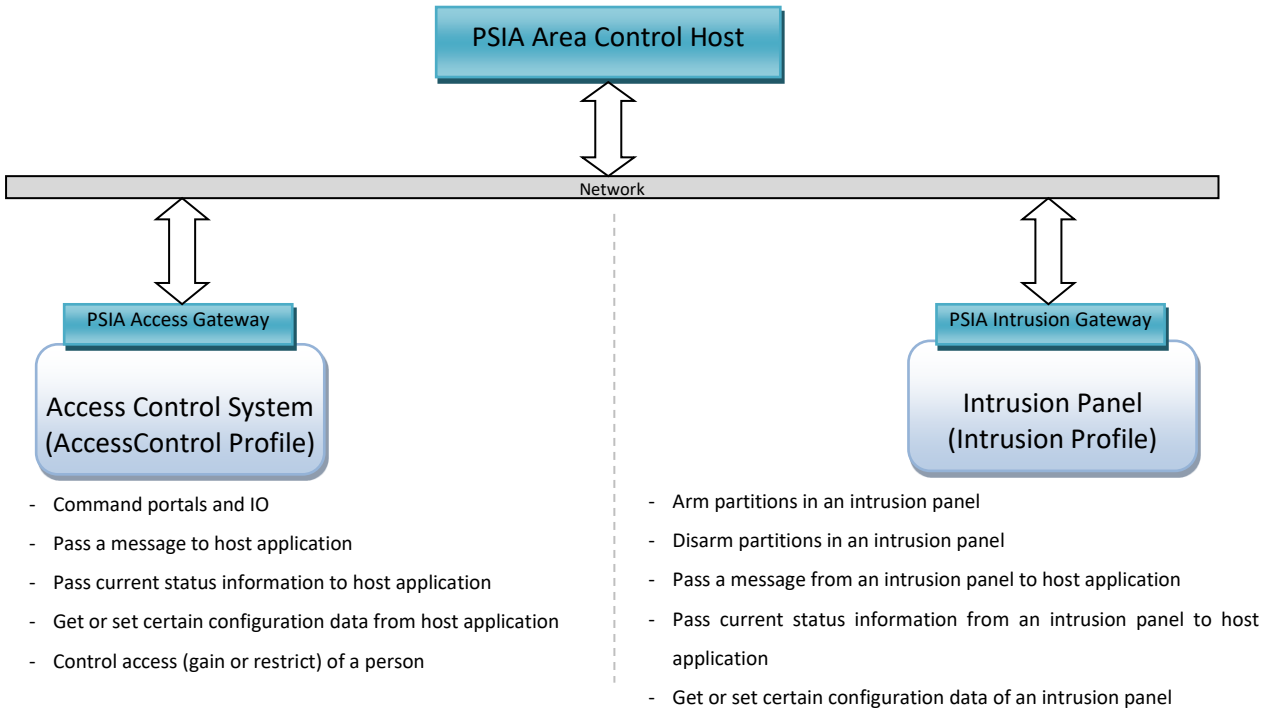
	This is the case when the local access control decision criteria are positively met, but the local access control decision criteria have a flag that external criteria must also me meet, prior to granting access. Also known as “host grant”.
Latched	Latched is a state of the locking device for a portal. It means that the portal is in a locked position.
Unlatched	Unlatched is a state of the locking device for a portal. It means that the portal is in an unlocked position.
Pulsed	An ‘unlatch’, commanded for a predetermined time frame. Often used to let a single credential holder through a Portal.
Monitored	A portal or an input point may be monitored for alarm events. If an alarm event is detected, then an alarm event message is generated.
Masked	A portal or an input point may be masked for alarm events. If a portal or an input point is masked, then it is not monitored, and it will not generate alarm events.
Forced	A portal is considered forced when it has gone from a secure, latched state to being opened, without having been unlatched (unlatched from a valid credential presentation, etc)
Held	A portal is considered held when it has been held or propped open for too long.
Role	Similar to, but a higher-level abstraction than, Permission – one which references Permissions. A Credential Holder can be assigned Roles, a Credential can be assigned Permissions. If a Role implies certain Permissions, and is assigned to a Credential Holder, this means that a Credential assigned to that Credential Holder has those Permissions.
PIN	A (generally) numeric sequence used to gain access to a Portal. Depending on the required identifier types of the Portal, the PIN may be used alone, or in conjunction with a Card or other identifier.

Access Control or Intrusion System:	
Partition Member	A Partition Member may be either a zone or an access point. It may have a group of devices such as inputs and outputs under it. So, a partition may have multiple partition members (zones or/and access points) and each partition member may have devices (inputs or/and outputs) under it. Partition Members allow inputs and outputs to be handled uniformly across intrusion and access control systems.
Maintenance	A state where a device or system is under maintenance, such that state changes such as input faults, portal forced portal held, zone alarms are not to be reported as “real” events and status changes.
External Inputs/Outputs	An External Input or Output is connected to the panel indirectly, for example through a sub-panel. A non-external Input or Output is directly on the panel.
Input Supervision Fault	A condition indicating that an input has been cut, shorted, connected to ground, or is connected to a foreign voltage. The ability to detect this is generally accomplished by using one or more resistors at the end of the line and/or in line, so that these conditions, as well as Normal and Active, all generate distinguishable resistance levels.
Output Supervision Fault	A condition indicating that an output is not functioning.

	Output supervision is typically used in intrusion and fire systems to ensure that an output (to a bell, siren, strobe, etc) is actually functioning.

Overview of PSIA Implementation

1.2 Area Control System Overview



A PSIA Area Control host application shall be able to perform the following use cases when it has devices with multiple profiles in the same network:

- Perform profile functionalities independently of each other
- For any PSIA access control system message (event), the host application shall be able to take all PSIA intended operations for a PSIA intrusion system
- For any PSIA intrusion system message (event), the host application shall be able to take all PSIA intended operations for a PSIA access control system
- PSIA access control system shall be able to control and get/set certain configuration data (PSIA intended) from PSIA intrusion system
- PSIA intrusion system shall be able to control and get/set certain configuration data (PSIA intended) from PSIA access control system
- Retrieve and set certain configuration data from both access control and intrusion systems; the host application can also build common configuration settings and rule engine
- Create common workflow management with system messages (events) and configurations from PSIA access control and intrusion systems combined together

If a device has the ability to perform PSIA intended operations of both access control and intrusion systems then it can implement PSIA services and resources for both access control and intrusion systems detailed in this specification. Most of the services and resources under Area Control are

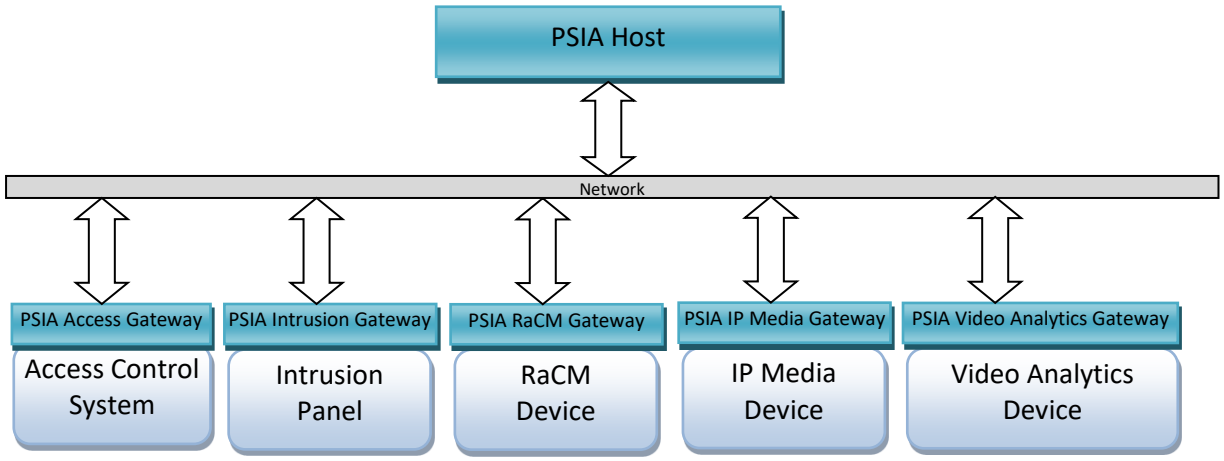
common to the access control and intrusion system profiles to simplify implementation of such devices.

1.3 Area Control Use cases

1.3.1 Primary Use Cases

Primary use cases for use of this specification can be found within the various Profiles included in the appendix of this document.

1.3.2 Use cases with other PSIA systems

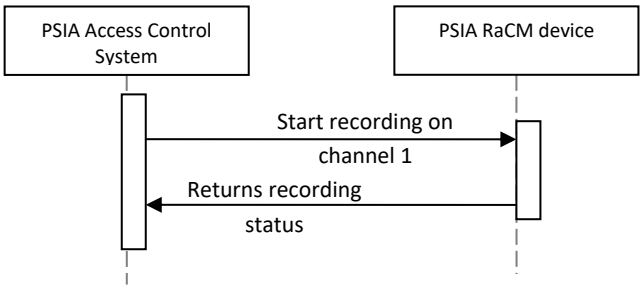


When a PSIA host application has one or more PSIA compatible systems or devices accessible through network:

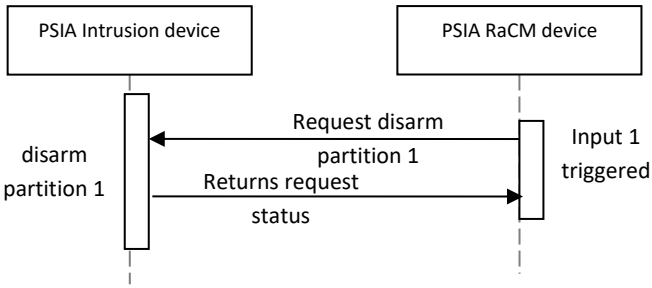
- PSIA host application will be able to perform ALL functionalities of access control system, intrusion system, RaCM (Recording and Content Management), IP Media and video analytics independently of each other
- All events from PSIA compatible systems/devices will be received by the host application and PSIA CMEM specification will be followed for event reception and related actions
- A PSIA compatible device/system can take up all PSIA intended operations with any other PSIA compatible devices/systems.

  Examples:

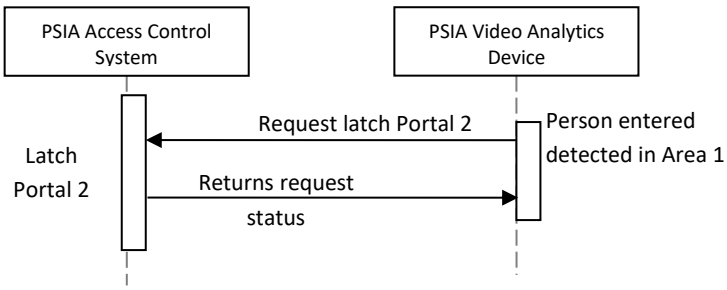
  - A PSIA access control system shall able to request a PSIA RaCM device to start recording on one or more tracks (by using the manual recording resources).



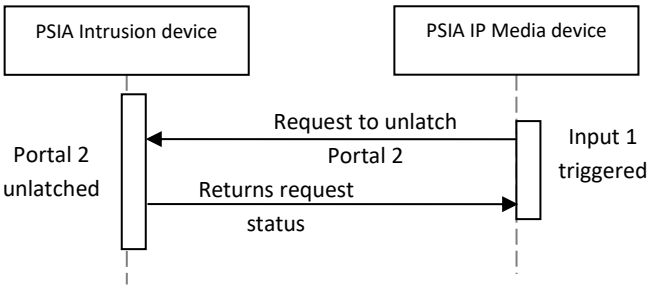
- In the event of an input device trigger, a PSIA RaCM device can request a PSIA intrusion device to arm/disarm a particular partition



- A PSIA access control system can latch a particular portal when requested by a PSIA video analytics device.



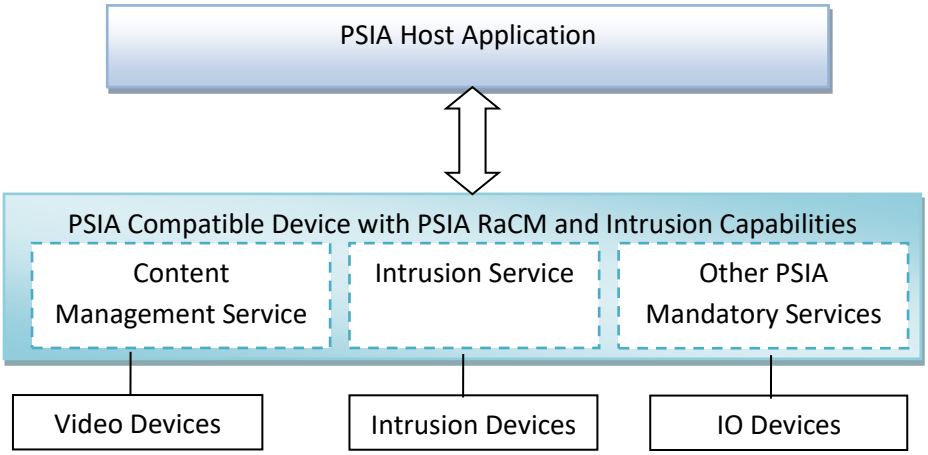
- An input trigger from a PSIA IP media device can unlatch a PSIA access control system portal.



- A PSIA IP media, IP video analytics or RaCM device can also implement PSIA Area control specification along with its respective PSIA specification only if it has the ability to perform operations intended for PSIA access control or intrusion or both.

Example:

If a PSIA RaCM device has ability to perform PSIA intended operations of intrusion system then the device can also implement this PSIA Area Control specification for intrusion devices.



PSIA Host Application shall be able to perform PSIA intended operations of both intrusion and RaCM services with this device.

Technology Requirement

To implement PSIA access control or intrusion services in a system or device, that system or device must support and be capable of implementing technology requirements listed in PSIA Service Model specification.

General Considerations

1.4 Profile Features

While this document defines both Intrusion and Access Control features, they are not mutually exclusive. While most implementations are expected to implement either the Intrusion or Access Control features, all profiles have been designed in such a way that an implementation may implement one, any number, or all, of them.

1.5 Global and Local IDs

Local ID is a required element for all resources. Global ID is required by profiles which use Internet topologies. If Global ID is required, this also means that resources can be queried by Global ID individually, or for lists, using startID/lastID/count.

Note that Global IDs always require {curly braces} around them. And when used in URIs, curly braces must be escaped.

Local ID 0 is a valid local ID. Local Id 0xffffffff is reserved by GMCH to mean no local ID. Implementations may not use the 0xffffffff local ID to refer to actual resources.

1.6 Requesting Ranges

When requesting lists, startID/lastID/count may be used to query a specific range of the list.

In general, startID and lastID are local IDs. For resources where global IDs are supported by the implementation (or required by the profile), startID and lastID may be global IDs.

If a range is requested, by default the range is interpreted to be a range within a total list by local ID. If, however, the startID or lastID is specified as a global ID, then the range is taken from the total list ordered by global ID.

If a PSIA client desires the range to be taken from a total list sorted by global ID, with no startID or lastID, but with a count, then it is necessary to make the request with the startID set to a global ID which is all 0s (yet is still a valid global ID with the braces and dashes).

Resource Requirements

/PSIA/AreaControl

Command	GET	PUT	POST	DEL
Configuration	?			
Status	?			

1.6.1 /PSIA/AreaControl/Partitions

Command	GET	PUT	POST	DEL
Partitions/info	?			
Partitions/status	?			
Partitions/armState		?		
Partitions/instanceDescription/<partitionID>	?	?		
Partitions/status/<partitionID>	?			
Partitions/armState/<partitionID>		?		
Partitions/occupants/<partitionID>	?			
Partitions/occupantCount/<partitionID>	?			

1.6.2 /PSIA/AreaControl/PartitionMembers/Zones

Command	GET	PUT	POST	DEL
Zones/info	?			
Zones/status	?			
Zones/instanceDescription/<zoneID>	?	?		
Zones/status/<zoneID>	?			
Zones/bypassState/<zoneID>		?		

1.6.3 /PSIA/AreaControl/PartitionMembers/Portals

Command	GET	PUT	POST	DEL
Portals/info	⍰			
Portals/status	⍰			
Portals/latchState		⍰		
Portals/accessOverride		⍰		
Portals/forcedMaskState		⍰		
Portals/heldMaskState		⍰		
Portals/instanceDescription/<portalID>	⍰	⍰		
Portals/status/<portalID>	⍰			
Portals/latchState/<portalID>		⍰		
Portals/accessOverride/<portalID>		⍰		
Portals/forcedMaskState/<portalID>		⍰		
Portals/heldMaskState/<portalID>		⍰		
Portals/accessResponse/<portalID>		⍰		
Portals/requiredIdentifierTypes/<portalID>	⍰	⍰		

1.6.4 /PSIA/AreaControl/Devices/Inputs

Command	GET	PUT	POST	DEL
Inputs/info	⍰			
Inputs/status	⍰			
Inputs/maskState		⍰		
Inputs/instanceDescription/<inputID>	⍰	⍰		
Inputs/status/<inputID>	⍰			
Inputs/maskState/<inputID>		⍰		

1.6.5 /PSIA/AreaControl/Devices/Outputs

Command	GET	PUT	POST	DEL
Outputs/info	?			
Outputs/status	?			
Outputs/instanceDescription/<outputID>	?	?		
Outputs/status/<outputID>	?			
Outputs/state/<outputID>		?		

1.6.6 /PSIA/AreaControl/Holidays

Command	GET	PUT	POST	DEL
Holidays/info	?		?	
Holidays/info/<holidayID>	?	?		?

1.6.7 /PSIA/AreaControl/TimeSchedules

Command	GET	PUT	POST	DEL
TimeSchedules/info	?		?	
TimeSchedules/info/<timeScheduleID>	?	?		?

1.6.8 /PSIA/AreaControl/Permissions

Command	GET	PUT	POST	DEL
Permissions/info	?		?	
Permissions/info/<permissionID>	?	?		?

1.6.9 /PSIA/AreaControl/CredentialFormats

Command	GET	PUT	POST	DEL
CredentialFormats/info	?		?	
CredentialFormats /info/<FormatID>	?	?		?

1.6.10 /PSIA/AreaControl/Credentials

Command	GET	PUT	POST	DEL
Credentials/info	?		?	
Credentials/info/<credentialID>	?	?		?
Credentials/state/<credentialID>	?	?		
Credentials/track/<credentialID>	?			
Credentials/count	?			

1.6.11 /PSIA/AreaControl/CredentialHolders

Command	GET	PUT	POST	DEL
CredentialHolders/info	?		?	
CredentialHolders/info/<credentialHolderID>	?	?		?
CredentialHolders/state/<credentialHolderID>	?	?		
CredentialHolders/track/<credentialHolderID>	?			
CredentialHolders/count	?			

1.7 /PSIA/Metadata

The following resources and services are provided for intrusion and access control systems:

Command	GET	PUT	POST	DEL
Metadata	?			
Metadata/index	?			
Metadata/description	?			
Metadata/metadataList	?	?		
Metadata/sessionSupport	?			
Metadata/channels	?			
Metadata/stream	?	?	?	?
Metadata/broadcasts (Optional/ Dependent)	?			
Metadata/Actions (For future versions)				

The /PSIA/Metadata service and its resources are detailed further in PSIA Common Metadata/Event Management Specification (CMEM). As of revision 2 of the CMEM Specification, the above resources are broken down into common profiles and are included in the profiles in this specification.

All HTTP methods that are not listed for any given resource above are invalid and will result in a HTTP '405' status code if issued against that resource. The status codes are explained in more detail in section "HTTP Status Codes and REST" of the PSIA Service Model specification.

Service Command Details for Intrusion

1.8 /PSIA/AreaControl

1.8.1 /PSIA/AreaControl

URI	/PSIA/AreaControl		Type	Service
Function	Intrusion service			
Methods	Query String(s)	Inbound Data	Return Result	

Notes:

Provides services to command and configure an intrusion panel.

1.8.2 /PSIA/AreaControl/configuration

URI	/PSIA/AreaControl/configuration		Type	Resource
Function	Used to retrieve the panel configuration			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<AreaControlConfiguration>	

Notes:

<AreaControlConfiguration> is explained with an example below; refer to the XSD schema for details.

Reponse to a GET request will be:

```
<AreaControlConfiguration version="1.0" xmlns="urn:psialliance-org">
  <CapabilityList>
    <Capability>Intrusion</Capability>
  </CapabilityList>
  <Manufacturer>Manufacturer name</Manufacturer>
  <ModelName>Model name</ModelName>
  <HardwareVersion>Rev 2.3</HardwareVersion>
  <FirmwareVersion>R1.0</FirmwareVersion>
  <AlarmServiceProvider>
    <CompanyName>Company Name</CompanyName>
    <OfficeTelephone>+918011111111</OfficeTelephone>
    <InstallerName>Installer name</InstallerName>
    <InstalledDate>2011-02-16T00:00:00+05:30</InstalledDate>
    <InstalledToUL>UL1024,UL609,UL1076</InstalledToUL>
  </AlarmServiceProvider>
  <AlarmReporting>
    <PrimaryReporting>
      <Path>+91 80 26588360</Path>
      <Format>SIA</Format>
    </PrimaryReporting>
    <AlternateReportingList>
      <AlternateReporting>
```



```
      <Path>+91 80 26588361</Path>
      <Format>SIA</Format>
    </AlternateReporting>
    <AlternateReporting>
      <Path>+91 80 26588362</Path>
      <Format>SIA</Format>
    </AlternateReporting>
  </AlternateReportingList>
 </AlarmReporting>
</AreaControlConfiguration>
```

1.8.3 /PSIA/AreaControl/status

URI	/PSIA/AreaControl/status		Type	Resource
Function	Used to retrieve status of the panel			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<AreaControlStatus>	

Notes:

<AreaControlStatus> is explained with examples below; refer to the XSD schema for details.

An intrusion system has no tamper issues faults, is not in maintenance mode, has no components in failure, has no communication problems with its upstream host or central station, and both battery and AC power are OK.

When a GET request is sent, the response will be:

```
<AreaControlStatus version="1.0" xmlns="urn:psialliance-org">
 <SystemStatus>
  <Tamper>OK</ Tamper >
  <Maintenance>OK</Maintenance>
  <Device>OK</Device>
 </SystemStatus>
 <CommunicationStatus>
  <Connection>OK</Connection>
 </CommunicationStatus>
 <PowerStatus>
  <BatteryPower>OK</BatteryPower>
  <ACPower>OK</ACPower>
 </PowerStatus>
</AreaControlStatus>
```


1.9 /PSIA/AreaControl/Partitions

1.9.1 /PSIA/AreaControl/Partitions/info

URI	/PSIA/AreaControl/Partitions/info		Type	Resource
Function	Used to get configuration of all partitions from the panel			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PartitionInfoList>	

Notes:

<PartitionInfoList> is explained with an example below; refer to the XSD schema for details.

In this example, a panel has 2 partitions with subcriber numbers 1234 and 5678 repectively.

Reponse to a GET request will be:

```
<PartitionInfoList xmlns="urn:psialliance-org" version="1.0">
 <PartitionInfo version="1.0">
  <ID>1</ID>
  <Name>Partition 1</Name>
  <Description>This is Partition 1</Description>
  <Intrusion>
   <Subscriber>1234</Subscriber>
   <EntryDelay>30</EntryDelay>
   <ExitDelay>60</ExitDelay>
   <IsLobbyPartition>false</IsLobbyPartition>
  </Intrusion>
 </PartitionInfo>
 <PartitionInfo version="1.0">
  <ID>2</ID>
  <Name>Partition 2</Name>
  <Description>This is Partition 2</Description>
  <Intrusion>
   <Subscriber>5678</Subscriber>
   <EntryDelay>30</EntryDelay>
   <ExitDelay>60</ExitDelay>
   <IsLobbyPartition>false</IsLobbyPartition>
  </Intrusion>
 </PartitionInfo>
</PartitionInfoList>
```

Note that Partitions used for intrusion will always have the <Intrusion> element, even if it is empty.

1.9.2 /PSIA/AreaControl/Partitions/status

URI	/PSIA/AreaControl/Partitions/status		Type	Resource
Function	Used to retrieve status of all partitions			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PartitionStatusList>	

Notes:

<PartitionStatusList> is explained with an example below; refer to the XSD schema for details.

In this example, a panel has 2 partitions with partition ID 1 and 2; the partition ID 1 is armed away and partition ID 2 is disarmed.

Response to a GET request will be:

```
<PartitionStatusList version="1.0" xmlns="urn:psialliance-org">
 <PartitionStatus version="1.0">
  <ID>1</ID>
  <Intrusion>
   <Arm>Armed</Arm>
   <ArmType>Away</ArmType>
   <ArmReady>Ready</ArmReady>
   <IntrusionAlarm>OK</IntrusionAlarm>
  </Intrusion>
 </PartitionStatus>
 <PartitionStatus version="1.0">
  <ID>2</ID>
  <Intrusion>
   <Arm>Disarmed</Arm>
   <ArmReady>Ready</ArmReady>
   <IntrusionAlarm>OK</IntrusionAlarm>
  </Intrusion>
 </PartitionStatus>
</PartitionStatusList>
```

1.9.3 /PSIA/AreaControl/Partitions/armState

URI	/PSIA/AreaControl/Partitions/armState		Type	Resource
Function	Used to arm or disarm all partitions.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state		<ResponseStatus>	

Notes:

state can take one of the following values:
Away, Stay, Disarmed

After execution of this command, all partitions will be in the state specified by state parameter.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of partitions: /PSIA/AreaControl/Partitions/status.

1.9.4 /PSIA/AreaControl/Partitions/instanceDescription/<partitionID>

URI	/PSIA/AreaControl/Partitions/instanceDescription/<partitionID>	Type	Resource
Function	Used to get or set the name and/or description of a partition		

Methods	Query String(s)	Inbound Data	Return Result
GET		None	<PartitionDescriptionInfo>
PUT		<PartitionDescriptionInfo>	<ResponseStatus>

Notes:

partitionID will be used as index for the required partition; the value can vary from 0 to N.

<PartitionDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If partitionID is 1 in a GET request, the response will be:

```
<PartitionDescriptionInfo xmlns="urn:psialliance-org" version="1.0">
  <Name>Partition 1</Name>
  <Description>This is Partition 1</Description>
</PartitionDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.9.5 /PSIA/AreaControl/Partitions/status/<partitionID>

URI	/PSIA/AreaControl/Partitions/status/<partitionID>		Type	Resource
Function	Used to retrieve status of a particular partition			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PartitionStatus>	

Notes:

<PartitionStatus> is explained with an example below; refer to the XSD schema for details.

If partitionID is 2 in a GET request, the response will be:

```
<PartitionStatus xmlns="urn:psialliance-org" version="1.0">
  <ID>2</ID>
  <Intrusion>
    <Arm>Disarmed</Arm>
    <ArmReady>Ready</ArmReady>
    <IntrusionAlarm>OK</IntrusionAlarm>
  </Intrusion>
</PartitionStatus>
```

1.9.6 /PSIA/AreaControl/Partitions/armState/<partitionID>

URI	/PSIA/AreaControl/Partitions/armState/<partitionID>		Type	Resource
Function	Used to arm or disarm a particular partition			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state		<ResponseStatus>.	

Notes:

state can take one of the following values:
Away, Stay, Disarmed

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of partition: /PSIA/AreaControl/Partitions/status.

1.10 /PSIA/AreaControl/PartitionMembers/Zones

1.10.1 /PSIA/AreaControl/PartitionMembers/Zones/info

URI	/PSIA/AreaControl/PartitionMembers/Zones/info		Type	Resource
Function	Used to get configuration of all zones from the panel			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] partitionID		<ZoneInfoList>	

Notes:

<ZoneInfoList> is explained with an example below; refer to the XSD schema for details.

In this example, a panel has 2 Interior zones, 1 24-hour zone, and 1 (perimeter) door zone. Zone 1 and Zone 2 are assigned to Partition 1. Zone 3 and Zone 4 are assigned to Partition 2.

The response to a GET request will be:

```
<ZoneInfoList version="1.0" xmlns="urn:psialliance-org">
 <ZoneInfo version="1.0">
  <ID>1</ID>
  <Name>Zone 1</Name>
  <Description>Zone 1: Interior</Description>
  <Type>Intrusion</Type>
  <ArmingType>Interior</ArmingType>
  <AssociatedPartitionID><ID>1</ID></AssociatedPartitionID>
  <ArmWithFault>false</ArmWithFault>
  <CanBeBypassed>false</CanBeBypassed>
 </ZoneInfo>
 <ZoneInfo version="1.0>"<ID>2</ID>
  <Name>Zone 2</Name>
  <Description>Zone 2: Interior</Description>
  <Type>Intrusion</Type>
  <ArmingType>Interior</ArmingType>
  <AssociatedPartitionID><ID>1</ID></AssociatedPartitionID>
  <ArmWithFault>false</ArmWithFault>
  <CanBeBypassed>false</CanBeBypassed>
 </ZoneInfo>
 <ZoneInfo version="1.0>"<ID>3</ID>
  <Name>Zone 3</Name>
  <Description>Zone 3: Fire</Description>
  <Type>Fire</Type>
  <ArmingType>TwentyFourHour</ArmingType>
  <AssociatedPartitionID><ID>2</ID></AssociatedPartitionID>
  <ArmWithFault>false</ArmWithFault>
  <CanBeBypassed>false</CanBeBypassed>
 </ZoneInfo>
 <ZoneInfo version="1.0>"<ID>4</ID>
  <Name>Zone 4</Name>
  <Description>Zone 4: Door</Description>
  <Type>Intrusion</Type>
  <ArmingType>Perimeter</ArmingType>
  <AssociatedPartitionID><ID>2</ID></AssociatedPartitionID>
  <ArmWithFault>false</ArmWithFault>
  <CanBeBypassed>false</CanBeBypassed>
 </ZoneInfo>
```



```
</ZoneInfoList>
```

The parameter partitionID can be supplied to get a filtered <ZoneInfoList> containing zones that belong to the partition ID specified in partitionID.

1.10.2 /PSIA/AreaControl/PartitionMembers/Zones/status

URI	/PSIA/AreaControl/PartitionMembers/Zones/status		Type	Resource
Function	Used to retrieve status of all zones			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] partitionID		<ZoneStatusList>	

Notes:

<ZoneStatusList> is explained with an example below; refer to the XSD schema for details.

In this example, there are 2 zones in the panel; Zone 1 in Partition 1 and Zone 2 in Partition 3. Zone 1 is bypassed and Zone 2 is in alarm.

The response to a GET request will be:

```
<ZoneStatusList xmlns="urn:psialliance-org" version="1.0">
 <ZoneStatus version="1.0">
  <ID>1</ID>
  <Bypass>Bypass</Bypass>
  <IntrusionAlarm>OK</IntrusionAlarm>
  <IntrusionTrouble>OK</IntrusionTrouble>
  <IntrusionFault>OK</IntrusionFault>
 </ZoneStatus>
 <ZoneStatus version="1.0">
  <ID>2</ID>
  <Bypass>Active</Bypass>
  <IntrusionAlarm>OK</IntrusionAlarm>
  <IntrusionTrouble>OK</IntrusionTrouble>
  <IntrusionFault>OK</IntrusionFault>
 </ZoneStatus>
</ZoneStatusList>
```

The parameter partitionID can be supplied to get a filtered <ZoneStatusList> containing zones that belong to the partition ID specified in partitionID.

1.10.3 /PSIA/AreaControl/PartitionMembers/Zones/instanceDescription/<zoneID>

URI	/PSIA/AreaControl/PartitionMembers/Zones/instanceDescription/<zoneID>		Type	Resource
Function	Used to get or set name and/or description of a zone			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<ZoneDescriptionInfo>	
PUT		<ZoneDescriptionInfo>	<ResponseStatus>	

Notes:

zoneID will be used as index for the required zone; the value can vary from 1 to N.

<ZoneDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If zoneID is 1 in a GET request, then the response will be:

```
<ZoneDescriptionInfo version="1.0" xmlns="urn:psialliance-org”>
  <Name>Zone 1</Name>
  <Description>This is Zone 1</Description>
</ZoneDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.10.4 /PSIA/AreaControl/PartitionMembers/Zones/status/<zoneID>

URI	/PSIA/AreaControl/PartitionMembers/Zones/status/<zoneID>		Type	Resource
Function	Used to retrieve status of a particular zone.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<ZoneStatus>	

Notes:

<ZoneStatus> is explained with an example below; refer to the XSD schema for details.

If Zone 102 is bypassed, is in partition 1 and a GET request has been sent with zoneID as 102 then the response will be:

```
<ZoneStatus xmlns="urn:psialliance-org" version="1.0">
  <ID>102</ID>
  <Bypass>Bypassed</Bypass>
  <IntrusionAlarm>OK</IntrusionAlarm>
  <IntrusionTrouble>OK</IntrusionTrouble>
  <IntrusionFault>OK</IntrusionFault>
</ZoneStatus>
```

1.10.5 /PSIA/AreaControl/PartitionMembers/Zones/bypassState/<zoneID>

URI	/PSIA/AreaControl/PartitionMembers/Zones/bypassState/<zoneID>		Type	Resource
Function	Used to bypass a particular zone			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state		<ResponseStatus>	

Notes:

state can take one of the following values:
Active, Bypass

This is explained with an example below; refer to the XSD schema for details.

To bypass zone 102 which is in partition 1, send a PUT request with state as 'Bypass' and zoneID as 102.

To activate zone 102, which is in partition 1, send a PUT request with state as 'Active' and zoneID as 102.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of zone: /PSIA/AreaControl/PartitionMembers/Zones/status

1.11 /PSIA/AreaControl/Permissions

1.11.1 /PSIA/AreaControl/Permissions/info

URI	/PSIA/AreaControl/Permissions/info		Type	Resource
Function	Provides access to all permissions of a panel and allows adding a new permission			
Methods	Query String(s)	Inbound Data	Return Result	
GET			< PermissionInfoList>	
POST		< PermissionInfo>	<ResponseStatus>	

Notes:

<PermissionInfoList> is explained with an example below; refer to the XSD schema for details.

An intrusion panel has two Permissions. Permission 1 has Authority Level 3 in both Partitions 1 & 2, and Permission 2 has Authority Level 4 in Partition 2.

Response to a GET request will be:

```
<PermissionInfoList xmlns="urn:psialliance-org" version="1.0">
 <PermissionInfo version="1.0">
  <ID>1</ID>
  <PrivilegeList>
   <Privilege>
    <Allow>
     <AuthorityLevel>3</AuthorityLevel>
     <PartitionIDList>
      <PartitionID>
       <ID>1</ID>
      </PartitionID>
      <PartitionID>
       <ID>2</ID>
      </PartitionID>
     </PartitionIDList>
    </Allow>
   </Privilege>
  </PrivilegeList>
 </PermissionInfo>
 <PermissionInfo version="1.0">
  <ID>2</ID>
  <PrivilegeList>
   <Privilege>
    <Allow>
     <AuthorityLevel>4</AuthorityLevel>
     <PartitionIDList>
      <PartitionID>
       <ID>2</ID>
      </PartitionID>
     </PartitionIDList>
    </Allow>
   </Privilege>
  </PrivilegeList>
 </PermissionInfo>
</PermissionInfoList>
```

This list of permissions can be obtained using a GET request.

A new permission can be added by passing <PermissionInfo> as inbound data using a POST request; the local ID assigned to the new permission will be returned in the <ResponseStatus>.

1.11.2 /PSIA/AreaControl/Permissions/info/<permissionID>

URI	/PSIA/AreaControl/Permissions/info/<permissionID>		Type	Resource
Function	Used to get or set information of a permission as well as to delete a permission			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PermissionInfo>	
PUT		<PermissionInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

permissionID will be used as index for the required permission; the value can vary from 1 to N.

<PermissionInfo> is explained with an example below; refer to the XSD schema for details.

An intrusion panel with two partitions has a Permission 'Permission 1' with Authority Level 3 in both partitions.

If permissionID is 1 in a GET request, then the response will be:

```
<PermissionInfo version="1.0" xmlns="urn:psialliance-org">
 <ID>1</ID>
 <PrivilegeList>
  <Privilege>
   <Allow>
    <AuthorityLevel>3</AuthorityLevel>
    <PartitionIDList>
     <PartitionID>
      <ID>1</ID>
     </PartitionID>
     <PartitionID>
      <ID>2</ID>
     </PartitionID>
    </PartitionIDList>
   </Allow>
  </Privilege>
 </PrivilegeList>
</PermissionInfo>
```

The information of a permission can be obtained using a GET request and updated using a PUT request. A permission can be deleted by issuing a DELETE request.

1.12 /PSIA/AreaControl/Credentials

1.12.1 /PSIA/AreaControl/Credentials/info

URI	/PSIA/AreaControl/Credentials/info		Type	Resource
Function	Provides access to all Credentials in a panel and allows adding a new Credential			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] credentialHolderUID [optional] identifierValue		<CredentialInfoList>	
POST		<CredentialInfo>	<ResponseStatus>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialInfoList> is explained with an example below; refer to the XSD schema for details.

An intrusion panel has two Credential Holders (users) and 2 Credentials (user codes). User 1 has Credential 1 (User Code 1234) with Authority Level 3 in both partitions 1 & 2 (covered by Permission 1). User 2 has Credential 2 (User Code 5678) with Authority Level 4 in Partition 2 (covered by Permission 2).

Response to a GET request will be:

```
<CredentialInfoList version="1.0" xmlns="urn:psialliance-org">
 <CredentialInfo version="1.0">
  <ID>1</ID>
  <Name>Credential 1</Name>
  <Description>Credential of User 1</Description>
  <AssignedToID><ID>1</ID></AssignedToID>
  <State>Active</State>
  <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
  <IdentifierInfoList>
   <IdentifierInfo>
    <Type>PIN</Type>
    <Value>1234</Value>
    <ValueEncoding>String</ValueEncoding>
   </IdentifierInfo>
  </IdentifierInfoList>
  <PermissionIDList>
   <PermissionID>
    <ID>1</ID>
   </PermissionID>
  </PermissionIDList>
  <RawValue>10011010010</RawValue>
 </CredentialInfo>
 <CredentialInfo version="1.0">
  <ID>2</ID>
  <Name>Credential 2</Name>
  <Description>Credential of User 2</Description>
  <AssignedToID><ID>2</ID></AssignedToID>
  <State>Active</State>
  <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
  <IdentifierInfoList>
```



```
    <IdentifierInfo>
      <Type>PIN</Type>
      <Value>5678</Value>
      <ValueEncoding>String</ValueEncoding>
    </IdentifierInfo>
  </IdentifierInfoList>
  <PermissionIDList>
    <PermissionID>
      <ID>2</ID>
    </PermissionID>
  </PermissionIDList>
  <RawValue>10011010010</RawValue>
 </CredentialInfo>
</CredentialInfoList>
```

A new credential can be added by passing <CredentialInfo> as inbound data using a POST request; the local ID assigned to the new credential will be returned in the <ResponseStatus>.

The parameter credentialHolderUID can be supplied to get a filtered <CredentialInfoList> containing credentials that belong to the credential holder UID specified in credentialHolderUID.

The parameter identifierValue can be supplied to get a filtered <CredentialInfoList> containing credentials that have identifier with this value.

1.12.2 /PSIA/AreaControl/Credentials/info/<credentialUID>

URI	/PSIA/AreaControl/Credentials/info/<credentialUID>		Type	Resource
Function	Used to get or set information of a credential as well as to delete a credential			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialInfo>	
PUT		<CredentialInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

credentialUID will be used as index for the required credential. The value is a GUID with the following syntax: {DF4D37A5-161C-471C-ABF6-0716159E9DA1}

<CredentialInfo> is explained with an example below; refer to the XSD schema for details.

An intrusion panel has a Credential Holder (user) User 1, with Credential “Credential 1”, with User Code 1234. This Credential has Permisison 1.

If credentialID is 1 in a GET request, then the response will be:

```
<CredentialInfo version="1.0" xmlns="urn:psialliance-org">
 <ID>1</ID>
 <Name>Credential 1</Name>
 <Description>Credential of User 1</Description>
 <AssignedToID><ID>1</ID></AssignedToID>
```



```
  <State>Active</State>
  <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
  <IdentifierInfoList>
   <IdentifierInfo>
    <Type>PIN</Type>
    <Value>1234</Value>
    <ValueEncoding>String</ValueEncoding>
   </IdentifierInfo>
  </IdentifierInfoList>
  <PermissionIDList>
   <PermissionID>
    <ID>1</ID>
   </PermissionID>
  </PermissionIDList>
  <RawValue>10011010010</RawValue>
</CredentialInfo>
```

The information of a credential can be obtained using a GET request and created or updated using a PUT request. A credential can be deleted by issuing a DELETE request.

When a credential is created on a system

1.12.3 /PSIA/AreaControl/Credentials/count

URI	/PSIA/AreaControl/Credentials/count		**Type**	Resource
Function	Used to count the available credentials in an access control system.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<CredentialInfoList>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialInfoList> is explained with an example below; refer to the XSD schema for details.

When a GET request is sent then the expected response will be:

```
<CredentialInfoList version="1.0" xmlns="urn:psialliance-org" countApplied="1"/>
```

Please note that no CredentialHo,ders should be returned in this call.

1.13 /PSIA/AreaControl/CredentialHolders

1.13.1 /PSIA/AreaControl/CredentialHolders/info

URI	/PSIA/AreaControl/CredentialHolders/info		Type	Resource
Function	Provides access to all users of a panel and allows adding a new user			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialHolderInfoList>	
POST		<CredentialHolderInfo>	<ResponseStatus>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialHolderInfoList> is explained with an example below; refer to the XSD schema for details.

An intrusion panel has two Credential Holders (users), User 1 and User 2.

Response to a GET request will be:

```
<CredentialHolderInfoList xmlns="urn:psialliance-org" version="1.0">
 <CredentialHolderInfo version="1.0">
  <ID>1</ID>
  <Name>User 1</Name>
  <Description>This is User 1</Description>
  <State>Active</State>
 </CredentialHolderInfo>
 <CredentialHolderInfo version="1.0">
  <ID>2</ID>
  <Name>User 2</Name>
  <Description>This is User 2</Description>
  <State>Active</State>
 </CredentialHolderInfo>
</CredentialHolderInfoList>
```

This list of users can be obtained using a GET request.

A new user can be added by passing <CredentialHolderInfo> as inbound data using a POST request; the local ID assigned to the new credential holder will be returned in the <ResponseStatus>.

1.13.2 /PSIA/AreaControl/CredentialHolders/info/<credentialHolderID>

URI	/PSIA/AreaControl/CredentialHolders/info/<credentialHolderUID>		Type	Resource
Function	Used to get or set information of a user as well as to delete a user			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialHolderInfo>	
PUT		<CredentialHolderInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

credentialHolderUID will be used as index for the required user. The value is a GUID with the following syntax: {DF4D37A5-161C-471C-ABF6-0716159E9DA1}

<CredentialHolderInfo> is explained with an examples below; refer to the XSD schema for details.

For a very basic example, if credentialHolderUID is {DF4D37A5-161C-471C-ABF6-0716159E9DA1} in a GET request, then the response will be:

```
<CredentialHolderInfo xmlns="urn:psialliance-org" version="1.0">
  <ID>1</ID>
  <UID>{3d17502d-c70f-4f35-8e91-82ab9e26ea28}</UID>
  <Name>User 1</Name>
  <Description>This is User 1</Description>
  <State>Active</State>
</CredentialHolderInfo>
```

This information can be obtained using a GET request and updated or created using a PUT request. A user can be deleted by issuing a DELETE request.

The following is an example of a more complex CredentialHolder object that has a UUID attribute to uniquely identify the individual across various systems as well as a few roles identified by UUID as well.

```
<CredentialHolderInfo xmlns="urn:psialliance-org" version="1.0">
  <ID>1</ID>
  <UID>{3d17502d-c70f-4f35-8e91-82ab9e26ea28}</UID>
  <Name>Doe, John</Name>
  <Description>This is John Doe</Description>
  <State>Active</State>
  <GivenName>John</GivenName>
  <Surname>Doe</Surname>
  <ActiveTill>2014-01-01T00:00:00+05:30</ActiveTill>
  <Disability>False</Disability>
  <Email>jdoe@example.com</ Email >
  <DomainName>EXAMPLE\jdoe</Disability>
  <RoleIDList>
    <RoleID>
      <GUID>{b3f08942-3d38-4afb-b2cf-ec826ba2d74f}</GUID>
    </RoleID>
    <RoleID>
      <GUID>{b6c46898-4b02-4e41-b050-d2a06a32b97f}</GUID>
    </RoleID>
  <RoleIDList>
</CredentialHolderInfo>
```

1.13.1 /PSIA/AreaControl/CredentialHolders/count

URI	/PSIA/AreaControl/CredentialHolders/count	Type	Resource
Function	Used to count the available CredentialHolders in an access control system.		

Methods	Query String(s)	Inbound Data	Return Result
GET			<CredentialHolderInfoList>

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialHolderInfoList> is explained with an example below; refer to the XSD schema for details.

When a GET request is sent then the expected response will be:

```
<CredentialHolderInfoList version="1.0" xmlns="urn:psialliance-org" countApplied="2"/>
```

Please note that no CredentialHo,ders should be returned in this call.

1.14 /PSIA/AreaControl/Devices/Outputs

1.14.1 /PSIA/AreaControl/Devices/Outputs/info

URI	/PSIA/AreaControl/Devices/Outputs/info	Type	Resource
Function	Used to get configuration of all outputs from the panel		
Methods	Query String(s)	Inbound Data	Return Result
GET	[optional] externalOnly [optional] portalID [optional] zoneID [optional] partitionID		<OutputInfoList>

Notes:

<OutputInfoList> is explained with an example below; refer to the XSD schema for details.

In this example, a panel has two outputs (strobes) associated with Partitions 1 and 2 repectively. The response to a GET request will be:

```
<OutputInfoList xmlns="urn:psialliance-org" version="1.0">
 <OutputInfo version="1.0">
  <ID>1</ID>
  <Name>Output 1</Name>
  <Description>This is Output 1</Description>
  <IsExternal>true</IsExternal>
  <PartitionID><ID>1</ID></PartitionID>
 </OutputInfo>
 <OutputInfo version="1.0">
  <ID>2</ID>
  <Name>Output 2</Name >
  <Description>This is Output 2</Description>
  <IsExternal>true</IsExternal>
  <PartitionID><ID>2</ID></PartitionID>
 </OutputInfo>
</OutputInfoList>
```

externalOnly value can be either ‘true’ or ’false’; the default value is ‘false’.
When value of externalOnly is ‘true’, then response to GET request contains external output devices only. When

value of externalOnly is 'false', then response to GET request contains external as well as internal output devices.

The parameters portalID, zoneID, or partitionID can be supplied to get a filtered <OutputInfoList> containing outputs associated with the specified Portal, Zone, or Partition.

1.14.2 /PSIA/AreaControl/Devices/Outputs/status

URI	/PSIA/AreaControl/Devices/Outputs/status		Type	Resource
Function	Used to retrieve status of all outputs			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] externalOnly [optional] portalID [optional] zoneID [optional] partitionID		<OutputStatusList>	

Notes:

<OutputStatusList> is explained with an example below; refer to the XSD schema for details.

In this example, outputs 1 and 2 are both connected (online), and off. Response to a GET request will be:

```
<OutputStatusList xmlns="urn:psialliance-org" version="1.0">
 <OutputStatus version="1.0">
  <ID>1</ID>
  <Connection>OK</Connection>
  <Output>Off</Output>
 </OutputStatus>
 <OutputStatus version="1.0">
  <ID>2</ID>
  <Connection>OK</Connection>
  <Output>Off</Output>
 </OutputStatus>
</OutputStatusList>
```

If output 1 is On in the above example, response to a GET request will be:

```
<OutputStatusList xmlns="urn:psialliance-org" version="1.0">
 <OutputStatus version="1.0">
  <ID>1</ID>
  <Connection>OK</Connection>
  <Output>On</Output>
 </OutputStatus>
 <OutputStatus version="1.0">
  <ID>2</ID>
  <Connection>OK</Connection>
  <Output>Off</Output>
 </OutputStatus>
</OutputStatusList>
```

If there are no ouputs configured, then response to a GET request will be:


```
<OutputStatusList version="1.0" xmlns="urn:psialliance-org"/>
```

externalOnly value can be either 'true' or 'false'; the default value is 'false'.
When value of externalOnly is 'true', then response to GET request contains external output devices only. When value of externalOnly is 'false', then response to GET request contains external as well as internal output devices.

The parameters portalID, zoneID, or partitionID can be supplied to get a filtered <OutputStatusList> containing outputs associated with the specified Portal, Zone, or Partition.

1.14.3 /PSIA/AreaControl/Devices/Outputs/instanceDescription/<outputID>

URI	/PSIA/AreaControl/Devices/Outputs/instanceDescription/<outputID>		Type	Resource
Function	Used to get or set name and/or description of an output			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<OutputDescriptionInfo>	
PUT		<OutputDescriptionInfo>	<ResponseStatus>	

Notes:

outputID will be used as index for the required output; the value can vary from 1 to N.

<OutputDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If outputID is 1 in a GET request, then the response will be:

```
<OutputDescriptionInfo version="1.0" xmlns="urn:psialliance-org">
  <Name>Output 1</Name>
  <Description>This is Output 1</Description>
</OutputDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.14.4 PSIA/AreaControl/Devices/Outputs/status/<outputID>

URI	/PSIA/AreaControl/Devices/Outputs/status/<outputID>		Type	Resource
Function	Used to retrieve status of a particular output			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<OutputStatus>	

Notes:

<OutputStatus> is explained with an example below; refer to the XSD schema for details.

If output 1 is online, and on, and is associated with Partition Member (Zone) 1 and a GET request is sent with outputID as 1, then the response will be:

```
<OutputStatus xmlns="urn:psialliance-org" version="1.0">
  <ID>1</ID>
```



```
  <Connection>OK</Connection>
  <Output>On</Output>
</OutputStatus>
```

1.14.5 /PSIA/AreaControl/Devices/Outputs/state/<outputID>

URI	/PSIA/AreaControl/Devices/Outputs/state/<outputID>		Type	Resource
Function	Used to turn a particular output on or off			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state		<ResponseStatus>	

Notes:

state can take one of the following values:
Off, On

This is explained with an example below; refer to the XSD schema for details.

To turn output 1 ON, sent a PUT request with state as 'On' and outputID as 1.

To turn output 1 OFF, send a PUT request with state as 'Off' and outputID as 1.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of output: PSIA/AreaControl/Devices/Outputs/status

Notes:

1. The optional Area Control services and resources are not explained in this chapter. They are covered under the next chapter on Access Control and can be used as is for intrusion as well.

Service Command Details for Access Control

1.15 /PSIA/AreaControl

URI	/PSIA/AreaControl		Type	Service
Function	Access Control service			
Methods	Query String(s)	Inbound Data	Return Result	

Notes:

Provides services to command and configure an access control system.

1.15.2 /PSIA/AreaControl/configuration

URI	/PSIA/AreaControl/configuration		Type	Resource
Function	Used to retrieve configuration of an access control system			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<AreaControlConfiguration>	

Notes:

Response to a GET request will be:

```
<AreaControlConfiguration xmlns="urn:psialliance-org" version="1.0">
  <CapabilityList>
    <Capability>AccessControl</Capability>
  </CapabilityList>
  <Manufacturer>Manufacturer Name</Manufacturer>
  <ModelName>Model Name</ModelName>
  <HardwareVersion>Rev 2.3</HardwareVersion>
  <FirmwareVersion>R 1.0</FirmwareVersion>
</AreaControlConfiguration>
```

1.15.3 /PSIA/AreaControl/status

URI	/PSIA/AreaControl/status		Type	Resource
Function	Used to retrieve status of the access control system			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<AreaControlStatus>	

Notes:

<AreaControlStatus> is explained with example below; refer to the XSD schema for details.

An access system has no tamper issues, is not in maintenance mode, has no components in failure, and has no communication problems with its upstream host, and both battery and AC power are OK.

When a GET request is sent, the response will be:

```
<AreaControlStatus xmlns="urn:psialliance-org" version="1.0">
  <SystemStatus>
    <Tamper>OK</Tamper>
    <Maintenance>OK</Maintenance>
    <Device>OK</Device>
  </SystemStatus>
  <CommunicationStatus>
    <Connection>OK</Connection>
  </CommunicationStatus>
<PowerStatus>
  <BatteryPower>OK</BatteryPower>
  <ACPower>OK</ACPower>
```



```
</PowerStatus>
 </AreaControlStatus>
```

1.16 /PSIA/AreaControl/PartitionMembers/Portals

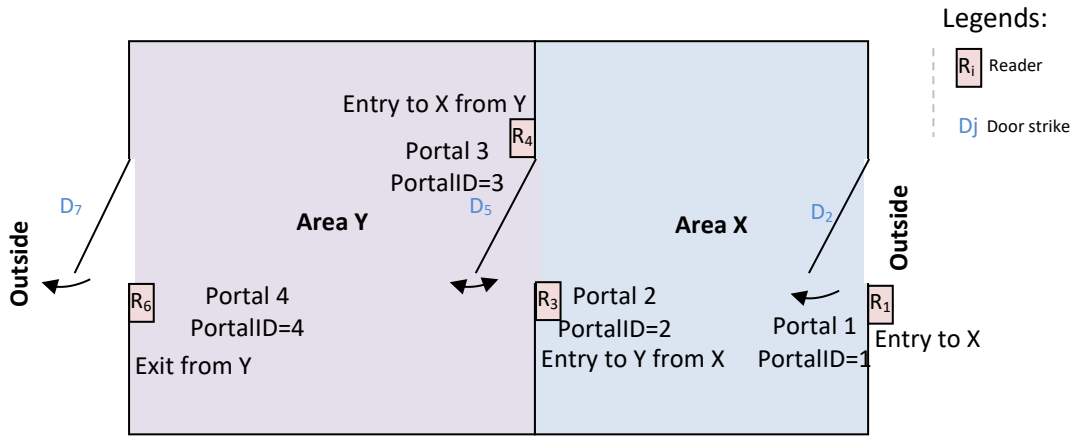
1.16.1 /PSIA/AreaControl/PartitionMembers/Portals/info

URI	/PSIA/AreaControl/PartitionMembers/Portals/info		Type	Resource
Function	Used to get configuration of all portals from the access control system. Note that a Portal only controls access in one direction between Partitions. If access is controlled in both directions, then 2 Portals are required.			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] partitionID		<PortalInfoList>	

Notes:

<PortalInfoList> is explained with an example below; refer to the XSD schema for details.

The following diagram is a simplified access control floor plan.



The above access acontrol system has two Partisions Area X and Area Y and 4 portals as described below:

Portal 1 is used to enter Area X from Outside
Portal 2 is used to enter Area Y from Area X
Portal 3 is used to enter Area X fromArea Y
Portal 4 is used to exit Area Y to Outside

When a GET request is sent, the response will be:

```
<PortalInfoList xmlns="urn:psialliance-org" version="1.0">
 <PortalInfo version="1.0">
  <ID>1</ID>
  <Name>Portal 1</Name>
```



```
  <Description>Entry to Area X from Outside</Description>
  <!--AssociatedPartitionID: missing, the portal is used to exit “outside”.-->
  <!--PartitionTo: The portal is used to enter this partition.-->
  <PartitionTo><ID>1</ID></PartitionTo>
 </PortalInfo>
 <PortalInfo version="1.0">
  <ID>2</ID>
  <Name>Portal 2</Name>
  <Description>Entry to Area Y from Area X</Description>
  <AssociatedPartitionID><ID>1</ID></AssociatedPartitionID>
  <PartitionTo><ID>2</ID></PartitionTo>
 </PortalInfo>
 <PortalInfo version="1.0">
  <ID>3</ID>
  <Name>Portal 3</Name>
  <Description>Entry to Area X from Area Y</Description>
  <AssociatedPartitionID><ID>2</ID></AssociatedPartitionID>
  <PartitionTo><ID>1</ID></PartitionTo>
 </PortalInfo>
 <PortalInfo version="1.0">
  <ID>4</ID>
  <Name>Portal 4</Name>
  <Description>Exit from Area Y to Outside</Description>
  <AssociatedPartitionID><ID>2</ID></AssociatedPartitionID>
  <!--PartitionTo: missing, portal is used to enter the “outside”.-->

 </PortalInfo>
</PortalInfoList>
```

The parameter partitionID can be supplied to get a filtered <PortalInfoList> containing portals that belong to the partition ID specified in partitionID.

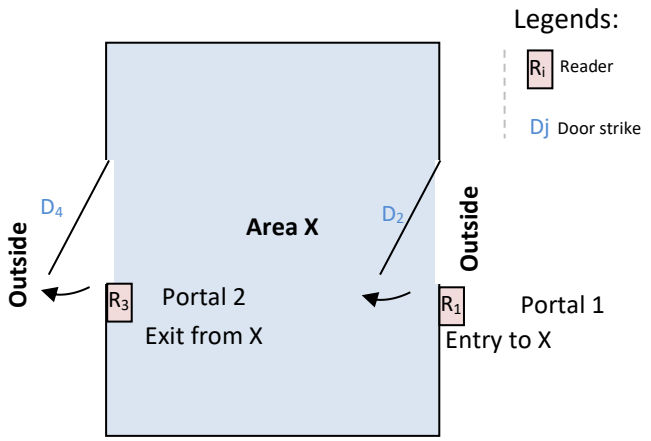
1.16.2 /PSIA/AreaControl/PartitionMembers/Portals/status

URI	/PSIA/AreaControl/PartitionMembers/Portals/status		Type	Resource
Function	Used to retrieve status of all portals			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] partitionID		<PortalStatusList>	

Notes:

<PortalStatusList> is explained with an example below; refer to the XSD schema for details.

The following diagram explains an access control floor plan.



The above access acontrol system has 2 portals (each with a reader and a door strike) as described below:

Portal 1 : used to enter Area X from Outside

Portal 2 : used to exit Area X to Outside

Portal 1 is masked (both forced and held) and Portal 2 is in a forced open condition.

When a GET request is sent, the response will be:

```
<PortalStatusList xmlns="urn:psialliance-org" version="1.0">
 <PortalStatus version="1.0">
  <ID>1</ID>
  <Latch>Unlatched</Latch>
  <PortalOpen>Closed</PortalOpen>
  <PortalForcedMask>Masked</PortalForcedMask>
  <PortalHeldMask>Masked</PortalHeldMask>
  <Tamper>OK</Tamper>
  <PortalForced>OK</PortalForced >
  <PortalHeld>OK</PortalHeld >
  <PortalAlarm>OK</PortalAlarm>
 </PortalStatus>
 <PortalStatus version="1.0">
  <ID>2</ID>
  <Latch>Unlatched</Latch>
  <PortalOpen>Closed</PortalOpen>
  <PortalForcedMask>Monitored</PortalForcedMask>
  <PortalHeldMask>Monitored</PortalHeldMask>
  <Tamper>OK</Tamper>
  <PortalForced>Forced</PortalForced >
  <PortalHeld>OK</PortalHeld >
  <PortalAlarm>OK</PortalAlarm>
 </PortalStatus>
</PortalStatusList>
```

The parameter partitionID can be supplied to get a filtered <PortalStatusList> containing portals that belong to the partition ID specified in partitionID.

1.16.3 /PSIA/AreaControl/PartitionMembers/Portals/latchState

URI	/PSIA/AreaControl/PartitionMembers/Portals/latchState	Type	Resource

Function	Used to latch or unlatch all portals in the access control system, or all in a specified Partition		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
PUT	[required] state [optional] partitionID	<PulseData>	<ResponseStatus>.

Notes:

state can take one of the following values:
Latched, Unlatched, Pulse

When state is 'Pulse', the PUT request may be sent with inbound data as <PulseData>. The absence of PulseData in the inbound data, or an empty PulseData both have the same meaning: that the pulse duration should be the default as defined by the receiving system. The effect will be that the portal will unlatch (unlock), starting from the time of recept of the PUT, for the duration in milliseconds specified in the PulseData (or the default).

The pulsing of the door is to be interpreted as a single door cycle, and the variable pulse time is intended to accommodate extended door times for disability or similar purposes. The pulse time is not intended for putting a portal into an unlatched state for an extended period of time.

Furthermore, the duration of the pulse in the request is to be applied as best as the implementation can. For example, if an access control panel only supports a single standard pulse time and a single disability pulse time, then the panel should "round up" to the nearest pulse time it supports, with a ceiling of the maximum pulse time it supports.

Example:

```
<PulseData version="1.0" xmlns="urn:psialliance-org">
    <DurationMillis>10000</DurationMillis>
</PulseData>
```

To latch all portals (in the system, or by partitionID), send a PUT request with state as 'Latched'

To unlatch all portals (in the system, or by partitionID),, send a PUT request with state as 'Unlatched'

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of portals: /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.4 /PSIA/AreaControl/PartitionMembers/Portals/accessOverride

URI	/PSIA/AreaControl/PartitionMembers/Portals/accessOverride		Type	Resource
Function	Used to latch or unlatch all portals in the access control system (or all portals in a specified Partition) such that normal access control is overridden. The portal may or may not read credentials presented at this time; however, it doesn't take access control decisions on presented credentials when normal access is overridden using this resource. Example: In case of threats, a portal may be latched in a way that it prevents normal access by overriding defined access permissions. Similarly, in case of emergency, a portal may be unlatched in a way that it doesn't require credential holders to present credentials in order to go through the portal.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	[required] accessOverrideState [optional] partitionID		<ResponseStatus>	

Notes:

accessOverrideState can take one of the following values:

Latched, Unlatched, Released

When accessOverrideState is 'Released', the access overridden latch or unlatch is released and the door goes back to normal access state.

To latch all portals (or all portals matching a specified partitionID) such that no access is granted to even authorized credential holders, send a PUT request with accessOverrideState as 'Latched'.

To release access override latch on all portals, send a PUT request with accessOverrideState as 'Released'.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of portals: /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.5 /PSIA/AreaControl/PartitionMembers/Portals/forcedMaskState

URI	/PSIA/AreaControl/PartitionMembers/Portals/forcedMaskState		Type	Resource
Function	Used to mask or enable monitoring of portal forced for all portals, or all portals in a Partition			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	[required] state [optional] partitionID		<ResponseStatus>	

Notes:

state can take one of the following values:
Masked, Monitored

To mask forced for all portals, a PUT request is sent.

To enable monitoring of forced for all portals (or all portals in the Partition matching partitionID),, a PUT request is sent with state as 'Monitored'.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of portals: /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.6 /PSIA/AreaControl/PartitionMembers/Portals/heldMaskState

URI	/PSIA/AreaControl/PartitionMembers/Portals/heldMaskState		Type	Resource
Function	Used to mask or enable monitoring of portal held for all portals, or all portals in a Partition			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state [optional] partitionID		<ResponseStatus>	

Notes:

state can take one of the following values:
Masked, Monitored

To mask held for all portals, a PUT request is sent.

To enable monitoring of held for all portals (or all portals in the Partition matching partitionID), a PUT request is sent with state as 'Monitored'.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of portals: /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.7 /PSIA/AreaControl/PartitionMembers/Portals/instanceDescription/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/instanceDescription/<portalID>		Type	Resource
Function	Used to get or set the name and/or description of a portal			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<PortalDescriptionInfo>	
PUT		<PortalDescriptionInfo>	<ResponseStatus>	

Notes:

portalID will be used as index for the required portal; the value can vary from 0 to N.

<PortalDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If <portalID> is 1 in a GET request, the response will be:

```
<PortalDescriptionInfo xmlns="urn:psialliance-org" version="1.0">
  <Name>Portal 1</Name>
  <Description>This is Portal 1</Description>
```



```
</PortalDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.16.8 /PSIA/AreaControl/PartitionMembers/Portals/status/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/status/<portalID>		Type	Resource
Function	Used to retrieve status of a particular portal			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PortalStatus>	

Notes:

This resource behaves the same as /PSIA/AreaControl/PartitionMembers/Portals/status, but retrieves status of a single portal.

<PortalStatus> is explained with an example below; refer to the XSD schema for details.

Portal 1 is in Latched state.

If a GET request is sent with portalID as 1, then the response will be:

```
<PortalStatus xmlns="urn:psialliance-org" version="1.0">
  <ID>1</ID>
  <Latch>Latched</Latch>
  <PortalForcedMask>Monitored</PortalForcedMask>
  <PortalHeldMask>Monitored</PortalHeldMask>
  <Tamper>OK</Tamper>
  <PortalAlarm>OK</PortalAlarm>
</PortalStatus>
```

1.16.9 /PSIA/AreaControl/PartitionMembers/Portals/latchState/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/latchState/<portalID>		Type	Resource
Function	Used to latch or unlatch a particular portal in the access control system			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state	<PulseData>	<ResponseStatus>	

Notes:

This request behaves the same as /PSIA/AreaControl/PartitionMembers/Portals/latchState, but latches or unlatches a single portal.

1.16.10 /PSIA/AreaControl/PartitionMembers/Portals/accessOverride/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/accessOverride/<portalID>	Type	Resource

Function	Used to latch or unlatch a particular portal in the access control system such that normal access control is overridden. The portal may or may not read credentials presented at this time; however, it doesn't take access control decisions on presented credentials when normal access is overridden using this resource. Example: In case of threats, a portal may be latched in a way that it prevents normal access by overriding defined access permissions. Similarly, in case of emergency, a portal may be unlatched in a way that it doesn't require credential holders to present credentials in order to go through the portal.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
PUT	[required] accessOverrideState		<ResponseStatus>.

Notes:

This request behaves the same as /PSIA/AreaControl/PartitionMembers/Portals/accessOverride but latches or unlatches a single portal.

accessOverrideState can take one of the following values:

Latched, Unlatched, Released

When accessOverrideState is 'Released', the access overridden latch or unlatch is released and the door goes back to normal access state.

To latch portal 1 such that no access is granted to even authorized credential holders, send a PUT request with portalID as 1 and accessOverrideState as 'Latched'.

To release access override latch on portal 1, send a PUT request with portalID as 1 and accessOverrideState as 'Released'.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of portal: /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.11 /PSIA/AreaControl/PartitionMembers/Portals/forcedMaskState/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/forcedMaskState/<portalID>	**Type**	Resource
Function	Used to mask or enable monitoring of portal forced of a particular portal		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
PUT	[required] state		<ResponseStatus>

Notes:

This request behaves the same as /PSIA/AreaControl/PartitionMembers/Portals/forcedMaskState (refer section 11.2.5), but masks or enables forced monitoring of a single portal.

This is explained with an example below; refer to the XSD schema for details.

To mask forced for Portal 1, send PUT request with portalID as 1.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of portal :. /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.12 /PSIA/AreaControl/PartitionMembers/Portals/heldMaskState/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/heldMaskState/<portalID>		Type	Resource
Function	Used to mask or enable monitoring of portal held of a particular portal			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state		<ResponseStatus>	

Notes:

This request behaves the same as /PSIA/AreaControl/PartitionMembers/Portals/heldMaskState (refer section 11.2.5), but masks or enables held monitoring of a single portal.

This is explained with an example below; refer to the XSD schema for details.

To mask held for Portal 1, send PUT request with portalID as 1.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of the portal :. /PSIA/AreaControl/PartitionMembers/Portals/status

1.16.13 /PSIA/AreaControl/PartitionMembers/Portals/requiredIdentifierTypes/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/requiredIdentifierTypes/<portalID>		Type	Resource
Function	Used to get or update the type of identifiers required at a portal			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<RequiredIdentifierList>	
PUT		<RequiredIdentifierList>	<ResponseStatus>	

Notes:

This should be used in order to set the list of identifiers that a panel must validate against. If a TimeScheduleID is provided (optional), then the required identifiers provided are only required during the time schedule identified (see /PSIA/AreaControl/TimeSchedules/info/<timeScheduleID>).

A GET to this resource will return the required identifiers that are currently in use.

Example 1: 2 factor requiring Card and PIN with no schedule set:

```
<RequiredIdentifierList version="1.0">
        <LogicalOr>
                <RequiredIdentifierInfo>
                        <LogicalAnd>
                                <IdentifierType>Card</IdentifierType>
                                <IdentifierType>PIN</IdentifierType>
```



```
                        </LogicalAnd>
                <RequiredIdentifierInfo>
        </LogicalOr>
</RequiredIdentifierList>
```

Example 2: 2-factor (Card and PIN) and 3-factor (Card and PIN and Biometric) on different time schedules

```
<RequiredIdentifierList version="1.0">
        <LogicalOr>
                <RequiredIdentifierInfo>
                        <LogicalAnd>
                                <IdentifierType>Card</IdentifierType>
                                <IdentifierType>PIN</IdentifierType>
                        </LogicalAnd>
                        <TimeScheduleID><ID>1</ID></TimeScheduleID>
        </RequiredIdentifierInfo>
        <RequiredIdentifierInfo>
                <LogicalAnd>
                        <IdentifierType>Card</IdentifierType>
                        <IdentifierType>PIN</IdentifierType>
                        <IdentifierType>Biometric</IdentifierType>
                </LogicalAnd>
                <TimeScheduleID><ID>2</ID></TimeScheduleID>
        </RequiredIdentifierInfo>
        </LogicalOr>
</RequiredIdentifierList>
```

Example 3: Either Card or PIN can grant entry

```
<RequiredIdentifierList version="1.0">
        <LogicalOr>
        <RequiredIdentifierInfo>
                <LogicalOr>
                        <IdentifierType>Card</IdentifierType>
                        <IdentifierType>PIN</IdentifierType>
                </LogicalOr>
        </RequiredIdentifierInfo>
        </LogicalOr>
</RequiredIdentifierList>
```

Combinations of possible credential types that should be accepted can be expressed using multiple RequiredIdentifierInfo fields with the same (or no) TimeScheduleID.

Example 4: 2 of 3 types required

```
<RequiredIdentifierList version="1.0">
        <LogicalOr>
                <RequiredIdentifierInfo>
                        <LogicalAnd>
```



```
                        <IdentifierType>Card</IdentifierType>
                        <IdentifierType>PIN</IdentifierType>
                    </LogicalAnd>
                </RequiredIdentifierInfo>
                <RequiredIdentifierInfo>
                    <LogicalAnd>
                        <IdentifierType>Card</IdentifierType>
                        <IdentifierType>Biometric</IdentifierType>
                    </LogicalAnd>
                </RequiredIdentifierInfo>
                <RequiredIdentifierInfo>
                    <LogicalAnd>
                        <IdentifierType>PIN</IdentifierType>
                        <IdentifierType>Biometric</IdentifierType>
                    </LogicalAnd>
                </RequiredIdentifierInfo>
            </LogicalOr>
</RequiredIdentifierList>
```

Note that only the inner RequiredIndentiferInfo allows both LogicalAnd and LogicalOr. The outer RequiredIdentifierList only allows LogicalOr. For instances when only one identifier is required, implementations and callers should use LogicalAnd as the inner logical attribute. For example, a Card-only configuration would be:

```
<RequiredIdentifierList version="1.0">
    <LogicalOr>
        <RequiredIdentifierInfo>
            <LogicalAnd>
                <IdentifierType>Card</IdentifierType>
            </LogicalAnd>
        <RequiredIdentifierInfo>
    </LogicalOr>
</RequiredIdentifierList>
```

1.16.14 /PSIA/AreaControl/PartitionMembers/Portals/accessResponse/<portalID>

URI	/PSIA/AreaControl/PartitionMembers/Portals/accessResponse/<portalID>		**Type**	Resource
Function	Used to respond to access.requested events passed via Metadata			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT		Granted or Denied	<ResponseStatus>	

Notes:

```
Sample put

<AccessResponse>granted</AccessResponse>
```



```
Or

<AccessResponse>denied</AccessResponse>
```


1.17 /PSIA/AreaControl/Devices/Inputs

1.17.1 /PSIA/AreaControl/Devices/Inputs/info

URI	/PSIA/AreaControl/Devices/Inputs/info		Type	Resource
Function	Used to retrieve configuration of all input devices			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] externalOnly [optional] portalID [optional] zoneID [optional] partitionID		<InputInfoList>	

Notes:

<InputInfoList> is explained with an example below; refer to the XSD schema for details.

If an access control system has a push button as an input associated with Portal 1 and a GET request is sent, then the response will be:

```
<InputInfoList xmlns="urn:psialliance-org" version="1.0">
 <InputInfo version="1.0">
  <ID>1</ID>
  <Name>Input 1</Name>
  <Description>Push Button at Area X, associated with Portal 1</Description>
  <IsExternal>true</IsExternal>
  <PortalID><ID>1</ID></PortalID>
 </InputInfo>
</InputInfoList>
```

externalOnly value can be either 'true' or 'false', the default value is 'false'.
When value of externalOnly is 'true', then response to GET request contains external input devices only. When value of externalOnly is 'false', then response to GET request contains external as well as internal input devices.

The parameters portalID, zoneID, or partitionID can be supplied to get a filtered <InputInfoList> containing outputs associated with the specified Portal, Zone, or Partition.

1.17.2 /PSIA/AreaControl/Devices/Inputs/status

URI	/PSIA/AreaControl/Devices/Inputs/status		Type	Resource
Function	Used to retrieve status of all inputs			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] externalOnly [optional] portalID [optional] zoneID [optional] partitionID		<InputStatusList>	

Notes:

<InputStatusList> is explained with an example below; refer to the XSD schema for details.

In an access control system, Input 1 is in normal condition and Input 2 is in an supervision fault condition. Both inputs are online. When a GET request is sent, the response will be:

```
<InputStatusList xmlns="urn:psialliance-org" version="1.0">
 <InputStatus version="1.0">
  <ID>1</ID>
  <Connection>OK</Connection>
  <SupervisionFault>OK</SupervisionFault>
  <Mask>Monitored</Mask>
 </InputStatus>
 <InputStatus version="1.0">
  <ID>2</ID>
  <Connection>OK</Connection>
  <SupervisionFault>Fault</SupervisionFault>
  <Mask>Monitored</Mask>
  <Tamper>OK</Tamper>
 </InputStatus>
</InputStatusList>
```

externalOnly value can be either 'true' or 'false', the default value is 'false'.
When value of externalOnly is 'true', then response to GET request contains external input devices only. When value of externalOnly is 'false', then response to GET request contains external as well as internal input devices.

The parameters portalID, zoneID, or partitionID can be supplied to get a filtered <InputStatusList> containing outputs associated with the specified Portal, Zone, or Partition.

1.17.3 /PSIA/AreaControl/Devices/Inputs/maskState

URI	/PSIA/AreaControl/Devices/Inputs/maskState		**Type**	Resource
Function	Used to mask or enable monitoring of all inputs			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	[required] state		<ResponseStatus>	

Notes:

state can take one of the following values:
Masked, Monitored

This is explained with an example below; refer to the XSD schema for details.

An access control system has 2 inputs which are in masked condution. To turn the mask OFF, send a PUT request with state as 'Monitored'.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of inputs: /PSIA/AreaControl/Devices/Inputs/status

1.17.4 /PSIA/AreaControl/Devices/Inputs/instanceDescription/<inputID>

URI	/PSIA/AreaControl/Devices/Inputs/instanceDescription/<inputID>		Type	Resource
Function	Used to get or set name and/or description of an input			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<InputDescriptionInfo>	
PUT		<InputDescriptionInfo>	<ResponseStatus>	

Notes:

inputID will be used as index for the required input; the value can vary from 1 to N.

<InputDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If inputID is 1 in a GET request, then the response will be:

```
<InputDescriptionInfo version="1.0" xmlns="urn:psialliance-org">
  <Name>Input 1</Name>
  <Description>This is Input 1</Description>
</InputDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.17.5 /PSIA/AreaControl/Devices/Inputs/status/<inputID>

URI	/PSIA/AreaControl/Devices/Inputs/status/<inputID>		Type	Resource
Function	Used to retrieve status of a particular input			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<InputStatus>	

Notes:

This resource behaves the same as /PSIA/AreaControl/Devices/Inputs/status (refer section 11.3.2), but retrieves status of a single input.

<InputStatus> is explained with an example below; refer to the XSD schema for details.

An access control system has an input which is in masked condtion, and connected (online). If a GET request is sent with inputID as 1, then the response will be:

```
<InputStatus xmlns="urn:psialliance-org" version="1.0">
  <ID>1</ID>
  <Connection>OK</Connection>
  <SupervisionFault>OK</SupervisionFault>
  <Mask>Masked</Mask>
</InputStatus>
```


1.17.6 /PSIA/AreaControl/Devices/Inputs/maskState/<inputID>

URI	/PSIA/AreaControl/Devices/Inputs/maskState/<inputID>		Type	Resource
Function	Used to mask or enable monitoring for an input			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	[required] state		<ResponseStatus>	

Notes:

This request behaves the same as /PSIA/AreaControl/Devices/Inputs/maskState (refer section 11.3.3) but masks or enables monitoring of a single input.

To mask input 2, send a PUT request with inputID as 2.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of input: /PSIA/AreaControl/Devices/Inputs/status

1.18 /PSIA/AreaControl/Devices/Outputs

1.18.1 /PSIA/AreaControl/Devices/Outputs/info

URI	/PSIA/AreaControl/Devices/Outputs/info		Type	Resource
Function	Used to retrieve configuration of all output devices			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] externalOnly [optional] portalID [optional] zoneID [optional] partitionID		<OutputInfoList>	

Notes:

<OutputInfoList> is explained with an example below; refer to the XSD schema for details.

An access control system has a light control as an output which switches on a light and is associated with Portal 1. When a GET request is sent, then the response will be:

```
<OutputInfoList xmlns="urn:psialliance-org" version="1.0">
 <OutputInfo version="1.0">
  <ID>1</ID>
  <Name>Output 1</Name>
  <Description>Light control for switch on light for Portal 1</Description>
  <IsExternal>true</IsExternal>
  <PortalID><ID>1</ID></PortalID>
 </OutputInfo>
</OutputInfoList>
```

externalOnly value can be either ‘true’ or ’false’; the default value is ‘false’.

When value of externalOnly is 'true', then response to GET request contains external output devices only. When value of externalOnly is 'false', then response to GET request contains external as well as internal output devices.

The parameters portalID, zoneID, or partitionID can be supplied to get a filtered <OutputInfoList> containing outputs associated with the specified Portal, Zone, or Partition.

1.18.2 /PSIA/AreaControl/Devices/Outputs/status

URI	/PSIA/AreaControl/Devices/Outputs/status		Type	Resource
Function	Used to retrieve status of all outputs			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] externalOnly [optional] portalID [optional] zoneID [optional] partitionID		<OutputStatusList>	

Notes:

<OutputStatusList> is explained with an example below; refer to the XSD schema for details.

In an access control system, output 1 is ON and output 2 is OFF. Both are connected (online). When a GET request is sent, the response will be:

```
<OutputStatusList xmlns="urn:psialliance-org" version="1.0">
 <OutputStatus version="1.0">
  <ID>1</ID>
  <Connection>OK</Connection>
  <Output>On</Output>
 </OutputStatus>
 <OutputStatus version="1.0">
  <ID>2</ID>
  <Connection>OK</Connection>
  <Output>Off</Output>
 </OutputStatus>
</OutputStatusList>
```

externalOnly value can be either 'true' or 'false'; the default value is 'false'.
When value of externalOnly is 'true', then response to GET request contains external output devices only. When value of externalOnly is 'false', then response to GET request contains external as well as internal output devices.

The parameters portalID, zoneID, or partitionID can be supplied to get a filtered <OutputStatusList> containing outputs associated with the specified Portal, Zone, or Partition.

1.18.3 /PSIA/AreaControl/Devices/Outputs/instanceDescription/<outputID>

URI	/PSIA/AreaControl/Devices/Outputs/instanceDescription/<outputID>		Type	Resource
Function	Used to get or set name and/ or description of an output			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<OutputDescriptionInfo>	

PUT		<OutputDescriptionInfo>	<ResponseStatus>

Notes:

outputID will be used as index for the required output; the value can vary from 1 to N.

<OutputDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If outputID is 1 in a GET request, then the response will be:

```
<OutputDescriptionInfo version="1.0" xmlns="urn:psialliance-org">
  <Name>Area X light control</Name>
  <Description>Light control for switch on Area X light</Description>
</OutputDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.18.4 /PSIA/AreaControl/Devices/Outputs/status/<outputID>

URI	/PSIA/AreaControl/Devices/Outputs/status/<outputID>	**Type**	Resource
Function	Used to retrieve status of a particular output		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<OutputStatus>

Notes:

This resource behaves the same as /PSIA/AreaControl/Devices/Outputs/status but retrieves status of a single output.

<OutputStatus> is explained with an example below; refer to the XSD schema for details.

An access control system has an output which is on, and connected (online). If a GET request is sent with outputID as 1, then the response will be:

```
<OutputStatus version="1.0" xmlns="urn:psialliance-org">
  <ID>1</ID>
  <Connection>OK</Connection>
  <Output>On</Output>
</OutputStatus>
```

1.18.5 /PSIA/AreaControl/Devices/Outputs/state/<outputID>

URI	/PSIA/AreaControl/Devices/Outputs/state/<outputID>	**Type**	Resource
Function	Used to turn an output point on or off		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
PUT	[required] state		<ResponseStatus>

Notes:

state can take one of the following values:
Off, On

To turn output 1 ON, send a PUT request with state as 'On' and outputID as 1.

In order to confirm the execution of this command, a separate GET request can also be sent to get the status of output: /PSIA/AreaControl/Devices/Outputs/status

1.19 /PSIA/AreaControl/Holidays

1.19.1 /PSIA/AreaControl/Holidays/info

URI	/PSIA/AreaControl/Holidays/info		Type	Resource
Function	Used to retrieve configuration of all holidays			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<HolidayInfoList>	
POST		<HolidayInfo>	<ResponseStatus>	

Notes:

<HolidayInfoList> is explained with an example below; refer to the XSD schema for details.

Following days are marked holidays in an access control system:

1- New Years Day – Jan 1st
2- Christmas Eve – Dec 24th
3- Christmas Day – Dec 25th
4- New Years Eve – Dec 31st

If a GET request is sent, then the response will be:

```
<HolidayInfoList version="1.0" xmlns="urn:psialliance-org">
 <HolidayInfo version="1.0">
  <ID>1</ID>
  <Name>New Years Day</Name>
  <Description>New Years Day – January 1</Description>
  <RecursYearly>true</RecursYearly>
  <StartDate>2011-01-01</StartDate>
  <EndDate>2011-01-01</EndDate>
 </HolidayInfo>
 <HolidayInfo version="1.0">
  <ID>2</ID>
  <Name>Christmas Eve</Name>
  <Description>Christmas Eve – December 24</Description>
  <RecursYearly>true</RecursYearly>
  <StartDate>2011-12-24</StartDate>
  <EndDate>2011-12-24</EndDate>
 </HolidayInfo>
 <HolidayInfo version="1.0">
  <ID>3</ID>
  <Description>Christmas Day</Description>
  <Description>Christmas Day – December 25</Description>
  <RecursYearly>true</RecursYearly>
  <StartDate>2011-12-25</StartDate>
  <EndDate>2011-12-25</EndDate>
 </HolidayInfo>
 <HolidayInfo version="1.0">
  <ID>4</ID>
  <Name>New Years Eve</Name>
  <Description>New Years Eve – December 31</Description>
  <RecursYearly>true</RecursYearly>
  <StartDate>2011-12-31</StartDate>
  <EndDate>2011-12-31</EndDate>
```



```
  </HolidayInfo>
</HolidayInfoList>
```

This list of holidays can be obtained using a GET request.

A new holiday can be added by passing <HolidayInfo> as inbound data using a POST request; the local ID assigned to the new holiday will be returned in the <ResponseStatus>.

1.19.2 /PSIA/AreaControl/Holidays/info/<holidayID>

URI	/PSIA/AreaControl/Holidays/info/<holidayID>		Type	Resource
Function	Used to get or set information of a holiday as well as to delete a holiday			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<HolidayInfo>	
PUT		<HolidayInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

holidayID will be used as index for the required holiday; the value can vary from 1 to N.

<HolidayInfo> is explained with an example below; refer to the XSD schema for details.

If holidayID is 1 in a GET request, then the response will be:

```
<HolidayInfo version="1.0" xmlns="urn:psialliance-org">
  <ID>1</ID>
  <Name>New Years Day</Name>
  <Description>New Years Day – January 1</Description>
  <RecursYearly>true</RecursYearly>
  <StartDate>2011-01-01</StartDate>
  <EndDate>2011-01-01</EndDate>
</HolidayInfo>
```

The information of a holiday can be obtained using a GET request and updated using a PUT request. A holiday can be deleted by issuing a DELETE request.

1.20 /PSIA/AreaControl/TimeSchedules

1.20.1 /PSIA/AreaControl/TimeSchedules/info

URI	/PSIA/AreaControl/TimeSchedules/info		Type	Resource
Function	Used to retrieve configuration of all time schedules as well as to add a new time schedule			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<TimeScheduleInfoList>	
POST		<TimeScheduleInfo>	<ResponseStatus>	

Notes:

Time schedules are typically used in access control systems to define when a credential holder should be allowed in or restriced from an access area and are associated with different access levels to grant or deny entry.

<TimeScheduleInfoList> is explained with an example below; refer to the XSD schema for details.

In an access control system, a time schedule is defined for an access area so that entry is restricted on all Saturdays and Sundays from morning 8 AM to evening 6 PM for everyone.

If a GET request is sent, the response will be:

```
<TimeScheduleInfoList version="1.0" xmlns="urn:psialliance-org">
  <TimeScheduleInfo version="1.0">
    <ID>1</ID>
    <Name>Time Schedule 1</Name>
    <Description>Saturday, Sunday 8am-6pm</Description>
    <TimeIntervalInfoList>
      <TimeIntervalInfo>
        <Day>Sunday</Day>
        <StartTime>08:00:00+05:30</StartTime>
        <EndTime>18:00:00+05:30</EndTime>
      </TimeIntervalInfo>
      <TimeIntervalInfo>
        <Day>Saturday</Day>
        <StartTime>08:00:00+05:30</StartTime>
        <EndTime>18:00:00+05:30</EndTime>
      </TimeIntervalInfo>
    </TimeIntervalInfoList>
  </TimeScheduleInfo>
</TimeScheduleInfoList>
```

This list of time schedules can be obtained using a GET request.

A new time schedule can be added by passing <TimeScheduleInfo> as inbound data using a POST request; the local ID assigned to the new schedule will be returned in the <ResponseStatus>.

1.20.2 /PSIA/AreaControl/TimeSchedules/info/<timeScheduleID>

URI	/PSIA/AreaControl/TimeSchedules/info/<timeScheduleID>		Type	Resource
Function	Used to get or set information of a time schedule as well as to delete a time schedule			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<TimeScheduleInfo>	
PUT		<TimeScheduleInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

timeScheduleID will be used as index for the required time schedule; the value can vary from 1 to N.

<TimeScheduleInfo> is explained with an example below; refer to the XSD schema for details.

In an access control system, a time schedule is defined for the purpose of defining a Permission for a Partition so that entry is only allowed on all Saturdays and Sundays from morning 8 AM to evening 6 PM for everyone.

If timeScheduleID is 1 in a GET request, then the response will be:

```
<TimeScheduleInfo version="1.0" xmlns="urn:psialliance-org">
 <ID>1</ID>
 <Name>Time Schedule 1</Name>
 <Description>Saturday, Sunday 8am-6pm</Description>
 <TimeIntervalInfoList>
  <TimeIntervalInfo>
   <Day>Sunday</Day>
   <StartTime>08:00:00+05:30</StartTime>
   <EndTime>18:00:00+05:30</EndTime>
  </TimeIntervalInfo>
  <TimeIntervalInfo>
   <Day>Saturday</Day>
   <StartTime>08:00:00+05:30</StartTime>
   <EndTime>18:00:00+05:30</EndTime>
  </TimeIntervalInfo>
 </TimeIntervalInfoList>
</TimeScheduleInfo>
```

The information of a time schedule can be obtained using a GET request and updated using a PUT request. A time schedule can be deleted by issuing a DELETE request.

1.21 /PSIA/AreaControl/Partitions

1.21.1 /PSIA/AreaControl/Partitions/info

URI	/PSIA/AreaControl/Partitions/info		Type	Resource
Function	Used to retrieve configuration of all partitions (access areas)			
Methods	Query String(s)	Inbound Data	Return Result	
GET			< PartitionInfoList>	

Notes:

<PartitionInfoList> is explained with an example below; refer to the XSD schema for details.

The following example uses 2 partitions: Area X and Area Y:

When a GET request is sent, the reponse will be:

```
<PartitionInfoList version="1.0" xmlns="urn:psialliance-org">
 <PartitionInfo version="1.0">
  <ID>1</ID>
  <Name>Area X</Name>
  <Description>This is Area X</Description>
  <AccessControl/>
 </PartitionInfo>
 <PartitionInfo version="1.0">
  <ID>2</ID>
  <Name>Area Y</Name>
  <Description>This is Area Y</Description>
  <AccessControl/>
 </PartitionInfo>
</PartitionInfoList>
```

Note that a Partition used for access control will always have the <AccessControl> element, even if it is empty.

1.21.2 /PSIA/AreaControl/Partitions/instanceDescription/<partitionID>

URI	/PSIA/AreaControl/Partitions/instanceDescription/<partitionID>		Type	Resource
Function	Used to get or set name and/or description of a Partition (access area)			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<PartitionDescriptionInfo>	
PUT		<PartitionDescriptionInfo>	<ResponseStatus>	

Notes:

partitionID will be used as index for the required Partition (access area).

<PartitionDescriptionInfo> is explained with an example below; refer to the XSD schema for details.

If partitionID is “1” in a GET request, then the response will be:


```
<PartitionDescriptionInfo version="1.0" xmlns="urn:psialliance-org">
  <Name>Area X</Name>
  <Description>This is Area X</Description>
</PartitionDescriptionInfo>
```

This description can be obtained using a GET request and updated using a PUT request.

1.21.3 /PSIA/AreaControl/Partitions/occupants/<PartitionID>

URI	/PSIA/AreaControl/Partitions/occupants/<PartitionID>		Type	Resource
Function	Used to get information occupants of a Partition.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PartitionInfo>	

Notes:

A partition can be used to represent a geographical area or location. It can be used for determining the location of a CredentialHolder.

Note that depending on profile, partition UID may be used in the URI instead of ID (as is true with most resources, depending on profile). In the following example, UID will be used.

Example: If the Partition UID is {303F7EB2-2E43-41E5-9D57-785FE840236A} in a GET request, then the response will be:

```
<PartitionOccupantList version="1.0" xmlns="urn:psialliance-org">
  <Partition>
   <ID>1</ID>
   <UID>{303F7EB2-2E43-41E5-9D57-785FE840236A}</UID>
   <Name>Partition 1</Name>
  </Partition>
  <PartitionOccupant>
   <CredentialHolder>
     <ID>1</ID>
     <UID>{cc26a16f-5854-4206-931e-d82f3be3f534}</UID>
     <Name>Doe, John</Name>
   </CredentialHolder>
  </PartitionOccupant>
  <PartitionOccupant>
   <CredentialHolder>
     <ID>1</ID>
     <UID>{aaf82d5d-1b5e-4f25-a517-6336ad58320e}</UID>
     <Name>Doe, Jane</Name>
   </CredentialHolder>
  </PartitionOccupant>
</PartitionOccupantList>
```

1.21.4 /PSIA/AreaControl/Partitions/occupantCount/<PartitionID>

URI	/PSIA/AreaControl/Partitions/occupantCount/<PartitionID>	Type	Resource
Function	Used to get the occupancy count of a partition.		

Methods	Query String(s)	Inbound Data	Return Result
GET			<PartitionOccupantCount>

Notes:

A GET on this resource retrieves the number of occupants presently in the Partition.

Note that the number of occupantCount in a partition may or may not match up exactly with the list of occupants returned by /PSIA/AreaControl/Partitions/occupants. This is because there are a number of ways that occupants can change partitions "anonymously", for example by using a request-to-exit (REX), or via a pulsed unlatched initiated externally.

Note that depending on profile, partition UID may be used in the URI instead of ID (as is true with most resources, depending on profile). In the following example, UID will be used.

To GET how many occupants are in Partition {303F7EB2-2E43-41E5-9D57-785FE840236A}, if the count was currently 10, the response would be:

<PartitionOccupantCount>10</PartitionOccupantCount>

1.22 /PSIA/AreaControl/Permissions

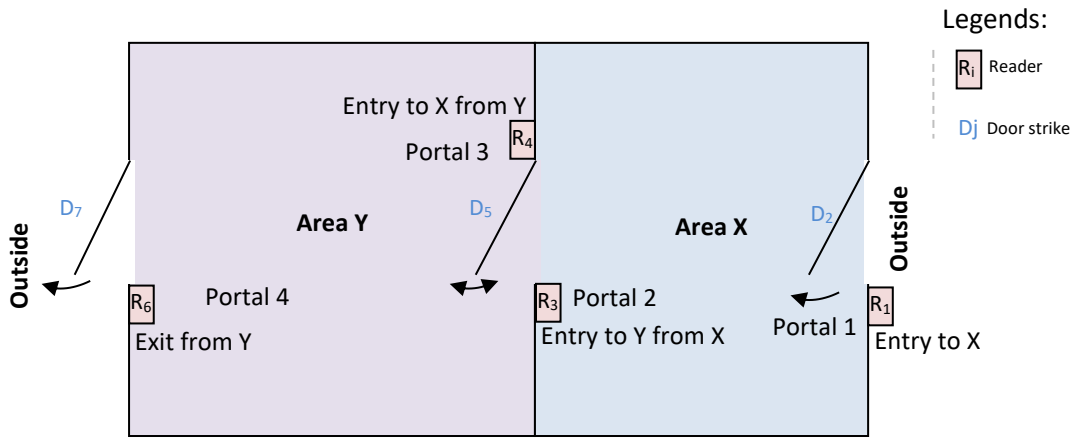
1.22.1 /PSIA/AreaControl/Permissions/info

URI	/PSIA/AreaControl/Permissions/info		Type	Resource
Function	Used to retrieve configuration of all access levels as well as to add a new access level			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PermissionInfoList>	
POST		<PermissionInfo>	<ResponseStatus>	

Notes:

<PermissionInfoList> is explained with an example below; refer to the XSD schema for details.

The following diagram is a simplified access control floor plan.



This access acontrol system has two access areas X and Y and 4 portals (consisting of reader and door strike) as described below:

Portal 1 is used to enter Area X from Outside
Portal 2 is used to enter Area Y from Area X
Portal 3 is used to enter Area X fromArea Y
Portal 4 is used to exit Area Y to Outside

The followings are the different access levels available in this system:

Administrator: Has access to all areas 24X7
Supervisor: Has access to all areas only on working days from 8AM to 6 PM
Employee: Has access to Area X but retricted from area Y on working days from 8AM to 5:50PM; from 5:50 PM to 6 PM, employee has access to Area Y but can't use portal 3 to re-enter Area X.

Let's assume that there are 3 time schedules: Time Schedule 1 (8 AM to 6 PM), Time Schedule 2 (8 AM to 5:50 PM), Time Schedule 3 (5:50 PM to 6 PM).

When a GET request is sent, then the response will be:


```
<PermissionInfoList version="1.0" xmlns="urn:psialliance-org">
 <PermissionInfo version="1.0">
  <ID>1</ID>
  <Name>Administrator</Name>
  <Description>This is Permission Administrator</Description>
  <PrivilegeList>
   <Privilege>
    <Allow>
     <PartitionIDList></PartitionIDList>
    </Allow>
   </Privilege>
  </PrivilegeList>
 </PermissionInfo>
 <PermissionInfo version="1.0">
  <ID>2</ID>
  <Name>Supervisor</Name>
  <Description>This is Permission Supervisor</Description>
  <PrivilegeList>
   <Privilege>
    <Allow>
     <TimeScheduleIDList>
      <TimeScheduleID>
       <ID>1</ID>
```



```
        </TimeScheduleID>
      </TimeScheduleIDList>
      <PartitionIDList></PartitionIDList>
    </Allow>
  </Privilege>
 </PrivilegeList>
</PermissionInfo>
<PermissionInfo version="1.0">
 <ID>3</ID>
 <Name>Employee</Name>
 <Description>This is Permission Employee</Description>
 <PrivilegeList>
  <Privilege>
   <Allow>
    <TimeScheduleIDList>
     <TimeScheduleID>
      <ID>2</ID>
     </TimeScheduleID>
    </TimeScheduleIDList>
    <PartitionIDList>
     <PartitionID>
      <ID>1</ID>
     </PartitionID>
    </PartitionIDList>
   </Allow>
   <Restrict>
    <TimeScheduleIDList>
     <TimeScheduleID>
      <ID>2</ID>
     </TimeScheduleID>
    </TimeScheduleIDList>
    <PartitionIDList>
     <PartitionID>
      <ID>2</ID>
     </PartitionID>
    </PartitionIDList>
   </Restrict>
  </Privilege>
  <Privilege>
   <Allow>
    <TimeScheduleIDList>
     <TimeScheduleID>
      <ID>3</ID>
     </TimeScheduleID>
    </TimeScheduleIDList>
    <PartitionIDList>
     <PartitionID>
      <ID>2</ID>
     </PartitionID>
    </PartitionIDList>
   </Allow>
   <Restrict>
    <TimeScheduleIDList>
     <TimeScheduleID>
      <ID>3</ID>
     </TimeScheduleID>
    </TimeScheduleIDList>
    <PortalIDList>
     <PortalID>
      <ID>3</ID>
```



```
        </PortalID>
      </PortalIDList>
    </Restrict>
  </Privilege>
 </PrivilegeList>
</PermissionInfo>
</PermissionInfoList>
```

This list of permissions can be obtained using a GET request.

A new permission can be added by passing <PermissionInfo> as inbound data using a POST request; the local ID assigned to the new permission will be returned in the <ResponseStatus>.

1.22.2 /PSIA/AreaControl/Permissions/info/<permissionID>

URI	/PSIA/AreaControl/Permissions/info/<permissionID>		Type	Resource
Function	Used to get or set information of an access level as well as to delete an access level			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<PermissionInfo>	
PUT		<PermissionInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

permissionID will be used as index for the required access level; the value can vary from 1 to N.

<PermissionInfo> is explained with an example below; refer to the XSD schema for details.

The followings are the different access levels available in this system:
Administrator: Has access to all areas 24X7
Supervisor: Has access to all areas only on working days from 8AM to 6 PM
Employee: Has access to Area X but retricted from area Y on working days from 8AM to 5:50PM; from 5:50 PM to 6 PM, employee has access to Area Y but can't use portal 3 to re-enter Area X.

Let's assume that there are three time schedules: Time Schedule 1 (8 AM to 6 PM), Time Schedule 2 (8 AM to 5:50 PM), Time Schedule 3 (5:50 PM to 6 PM).

If permissionID is 3 in a GET request, then the response will be:

```
<PermissionInfo version="1.0" xmlns="urn:psialliance-org">
 <ID>3</ID>
 <Name>Employee</Name>
 <Description>This is Permission Employee</Description>
 <PrivilegeList>
  <Privilege>
   <Allow>
    <TimeScheduleIDList>
     <TimeScheduleID>
      <ID>2</ID>
     </TimeScheduleID>
    </TimeScheduleIDList>
```



```
      <PartitionIDList>
        <PartitionID>
          <ID>1</ID>
        </PartitionID>
      </PartitionIDList>
    </Allow>
    <Restrict>
      <TimeScheduleIDList>
        <TimeScheduleID>
          <ID>2</ID>
        </TimeScheduleID>
      </TimeScheduleIDList>
      <PartitionIDList>
        <PartitionID>
          <ID>2</ID>
        </PartitionID>
      </PartitionIDList>
    </Restrict>
  </Privilege>
  <Privilege>
    <Allow>
      <TimeScheduleIDList>
        <TimeScheduleID>
          <ID>3</ID>
        </TimeScheduleID>
      </TimeScheduleIDList>
      <PartitionIDList>
        <PartitionID>
          <ID>2</ID>
        </PartitionID>
      </PartitionIDList>
    </Allow>
    <Restrict>
      <TimeScheduleIDList>
        <TimeScheduleID>
          <ID>3</ID>
        </TimeScheduleID>
      </TimeScheduleIDList>
      <PortalIDList>
        <PortalID>
          <ID>3</ID>
        </PortalID>
      </PortalIDList>
    </Restrict>
  </Privilege>
 </PrivilegeList>
</PermissionInfo>
```


The information of an access level can be obtained using a GET request and updated using a PUT request. An access level can be deleted by issuing a DELETE request.

1.23 /PSIA/AreaControl/CredentialFormats

1.23.1 /PSIA/AreaControl/CredentialFormats/info

URI	/PSIA/AreaControl/CredentialFormats/info		Type	Resource
Function	Used to retrieve card formats or to add a new card format			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CardFormatInfoList>	
POST		<WiegandCardFormatInfo> OR <MagstripeCardFormatInfo>	<ResponseStatus>	

Notes:

<CardFormatInfoList> is explained with an example below; refer to the XSD schema for details.

In this example, a 26 bit Weigand card with facility code of 123 is provided.

```
<CardFormatInfoList version="1.0" xmlns="urn:psialliance-org" >
	<WiegandCardFormatInfo version="1.0">
		<ID>1</ID>
		<Name>26 Bit Weigand</Name>
		<Length>26</Length>
		<DataFieldList>
			<DataField>
				<Type>FacilityCode</Type>
				<Offset>1</Offset>
				<Length>8</Length>
				<Value>123</Value>
				<ValueEncoding>Decimal</ValueEncoding>
			</DataField>
			<DataField>
				<Type>CardNum</Type>
				<Offset>6</Offset>
				<Length>16</Length>
			</DataField>
			<ParityField>
				<ParityType>Even</ParityType>
				<CheckBitLocation>0</CheckBitLocation>
				<Start>0</Start>
				<Length>13</Length>
			</ParityField>
			<ParityField>
				<ParityType>Odd</ParityType>
				<CheckBitLocation>25</CheckBitLocation>
				<Start>12</Start>
				<Length>13</Length>
			</ParityField>
		</DataFieldList>
	</WiegandCardFormatInfo>
</CardFormatInfoList>
```

This format as well as any other formats defined on the system can be obtained using a GET request.

A new format can be added by passing either < WiegandCardFormatInfo > or <MagstripeCardFormatInfo> as inbound data using a POST request; the local ID assigned to the new card format will be returned in the <ResponseStatus>.

Note: Start and Length is the preferred method of specifying parity unless the parity is non-contiguous, in which case a mask can be specified in the form of zeros and ones. Hypothetically, if 26 bit cards used alternating parity the parity specification would be as follows:

```
<...>
        <ParityField>
                <ParityType>Even</ParityType>
                <CheckBitLocation>0</CheckBitLocation>
                <Mask>01010101010101010101010101</Mask>
        </ParityField>
        <ParityField>
                <ParityType>Odd</ParityType>
                <CheckBitLocation>25</CheckBitLocation>
                <Mask>10101010101010101010101010</Mask>
        </ParityField>
<...>
```

1.23.2 /PSIA/AreaControl/CredentialFormats/info/<formatID>

URI	/PSIA/AreaControl/CredentialFormats/info/<formatID>		Type	Resource
Function	Used to get or set a card format as well as delete a card format			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<WiegandCardFormatInfo> OR <MagstripeCardFormatInfo>	
PUT		<WiegandCardFormatInfo> OR <MagstripeCardFormatInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

The information of a card format can be obtained using a GET request and updated or created using a PUT request. A credential can be deleted by issuing a DELETE request.

For more information about Card Formats, see documentation in /PSIA/AreaControl/CredentialFormats/info

1.24 /PSIA/AreaControl/Credentials

1.24.1 /PSIA/AreaControl/Credentials/info

URI	/PSIA/AreaControl/Credentials/info		Type	Resource
Function	Used to retrieve configuration of all access credentials and to add a new credential			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[optional] credentialHolderID [optional] identifierValue		<CredentialInfoList>	

POST		<CredentialInfo>	<ResponseStatus>

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialInfoList> is explained with an example below; refer to the XSD schema for details.

Access credential 1 (card number "12345") is valid from Jan 1, 2011 and has no expiry date; credential 2 (card number "23456") is inactive. Both cards are of format ID 1, which is a 32 bit card with a Facility Code of 23 and Issue Code of 123. Further details about the card format, including parity specification (when the card format is not proprietary) can be found under the /PSIA/AreaControl/CredentialFormats/ heirarchy of resources. The RawValue of the card represents the binary data stored in the card. The RawValue can be decoded by referencing the CredentialFormat the card represents and applying the CredentialFormat's rules.

When a GET request is sent, the response will be:

```
<CredentialInfoList version="1.0" xmlns="urn:psialliance-org">
 <CredentialInfo version="1.0">
  <ID>1</ID>
  <Name>Credential 1</Name>
  <Description>Assigned to David</Description>
  <AssignedToID><ID>1467</ID></AssignedToID>
  <State>Active</State>
  <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
  <ValidFrom>2011-01-01T00:00:00+05:30</ValidFrom>
  <IdentifierInfoList>
   <IdentifierInfo>
    <Type>Card</Type>
    <Value>12345</Value>
   <ValueEncoding>Decimal</ValueEncoding>
    <Format>
     <ID>1</ID>
     <Name>32 bit</Name>
    </Format>
    <CardComponentList>
     <IdentifierCardComponentInfo>
      <Type>FacilityCode</Type>
      <Value>23</Value>
      <ValueEncoding>Decimal</ValueEncoding>
     </IdentifierCardComponentInfo>
     <IdentifierCardComponentInfo>
      <Type>IssueCode</Type>
      <Value>123</Value>
      <ValueEncoding>Decimal</ValueEncoding>
     </IdentifierCardComponentInfo>
    </CardComponentList>
   </IdentifierInfo>
  </IdentifierInfoList>
  <PermissionIDList>
   <PermissionID>
    <ID>1</ID>
   </PermissionID>
  </PermissionIDList>
  <RawValue>11000000111001</RawValue>
 </CredentialInfo>
 <CredentialInfo version="1.0">
  <ID>2</ID>
```



```
    <Name>Credential 2</Name>
    <Description>Assigned to John</Description>
    <AssignedToID><ID>1471</ID></AssignedToID>
    <State>Inactive</State>
    <IdentifierInfoList>
     <IdentifierInfo>
      <Type>Card</Type>
      <Value>23456</Value>
     <ValueEncoding>Decimal</ValueEncoding>
       <Format>
         <ID>1</ID>
         <Name>32 bit</Name>
       </Format>
       <CardComponentList>
         <IdentifierCardComponentInfo>
           <Type>FacilityCode</Type>
           <Value>23</Value>
           <ValueEncoding>Decimal</ValueEncoding>
         </IdentifierCardComponentInfo>
         <IdentifierCardComponentInfo>
           <Type>IssueCode</Type>
           <Value>123</Value>
           <ValueEncoding>Decimal</ValueEncoding>
         </IdentifierCardComponentInfo>
       </CardComponentList>
     </IdentifierInfo>
    </IdentifierInfoList>
    <PermissionIDList>
     <PermissionID>
      <ID>2</ID>
     </PermissionID>
    </PermissionIDList>
    <RawValue>101101110100000</RawValue>
   </CredentialInfo>
  </CredentialInfoList>
```


This list of credentials can be obtained using a GET request.

A new credential can be added by passing <CredentialInfo> as inbound data using a POST request; the local ID assigned to the new credential will be returned in the <ResponseStatus>.

The parameter credentialHolderID can be supplied to get a filtered <CredentialInfoList> containing credentials that belong to the credential holder ID specified in credentialHolderID.

The parameter identifierValue can be supplied to get a filtered <CredentialInfoList> containing credentials that have identifier with this value.

1.24.2 /PSIA/AreaControl/Credentials/info/<credentialID>

URI	/PSIA/AreaControl/Credentials/info/<credentialID>	Type	Resource
Function	Used to get or set information of a credential as well as to delete a credential		

Methods	Query String(s)	Inbound Data	Return Result
GET			<CredentialInfo>
PUT		<CredentialInfo>	<ResponseStatus>
DELETE			<ResponseStatus>

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

credentialID will be used as index for the required access credential.

<CredentialInfo> is explained with an example below; refer to the XSD schema for details.

Access credential 1 (card number "12345") is valid from Jan 1, 2011 and has no expiry date; credential 2 (card number "23456") is inactive.

If credentialID is 1 in a GET request, then the response will be:

```
<CredentialInfo version="1.0" xmlns="urn:psialliance-org">
 <ID>1</ID>
 <Name>Credential 1</Name>
 <Description>Assigned to David</Description>
 <AssignedToID><ID>1467</ID></AssignedToID>
 <State>Active</State>
 <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
 <ValidFrom>2011-01-01T00:00:00+05:30</ValidFrom>
 <IdentifierInfoList>
  <IdentifierInfo>
   <Type>Card</Type>
   <Value>12345</Value>
  <ValueEncoding>Decimal</ValueEncoding>
   <Format>
    <ID>1</ID>
    <Name>32 bit</Name>
   </Format>
  </IdentifierInfo>
 </IdentifierInfoList>
 <PermissionIDList>
  <PermissionID>
   <ID>1</ID>
  </PermissionID>
 </PermissionIDList>
 <RawValue>101101110100000</RawValue>
</CredentialInfo>
```

The information of a credential can be obtained using a GET request and updated or created using a PUT request. A credential can be deleted by issuing a DELETE request.

1.24.3 /PSIA/AreaControl/Credentials/state/<credentialID>

URI	/PSIA/AreaControl/Credentials/state/<credentialID>	Type	Resource

Function	Used to retrieve or update state of a particular credential		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET		None	<CredentialState>
PUT	[required] activeState		<ResponseStatus>

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

activeState can take one of the following values:
Inactive, Active

<CredentialState> is explained with an example below; refer to the XSD schema for details.

When a GET request is sent with credentialID as 1 (an expired credential), then the response will be:

```
<CredentialState version="1.0" xmlns="urn:psialliance-org">
  <State>Expired</State>
</CredentialState>
```

This state can be obtained using a GET request and updated using a PUT request.

1.24.4 /PSIA/AreaControl/Credentials/track/<credentialID>

URI	/PSIA/AreaControl/Credentials/track/<credentialID>		Type	Resource
Function	Used to track a credential in an access control system. This retrieves the report of granted and denied access events for a given credential.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	[required] fromTime [required] toTime		<CredentialTrackList>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

fromTime and toTime take strings which are in the xs:datetime format.

<CredentialTrackList> is explained with an example below; refer to the XSD schema for details.

Credential number 1 has gained access to Area X at 10:00 AM through Portal 1 and exited at 10:10 AM via Portal 2.

When a GET request is sent with credentialID as 1, fromTime as 2011-01-01T09:00:00+05:30 and toTime as 2011-01-01T10:05:00+05:30, then the response will be:

```
<CredentialTrackList version="1.0" xmlns="urn:psialliance-org">
  <CredentialTrack>
    <TrackTime>2011-01-01T10:00:00+05:30</TrackTime>
    <PartitionID><ID>1</ID></PartitionID>
```



```
    <PortalID><ID>1</ID></PortalID>
    <AccessState>Granted</AccessState>
  </CredentialTrack>
</CredentialTrackList>
```

1.24.5 /PSIA/AreaControl/Credentials/count

URI	/PSIA/AreaControl/Credentials/count		Type	Resource
Function	Used to count the available credentials in an access control system.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialInfoList>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialInfoList> is explained with an example below; refer to the XSD schema for details.

When a GET request is sent then the expected response will be:

```
<CredentialInfoList version="1.0" xmlns="urn:psialliance-org" countApplied="1"/>
```

Please note that no CredentialHo,ders should be returned in this call.

1.25 /PSIA/AreaControl/CredentialHolders

1.25.1 /PSIA/AreaControl/CredentialHolders/info

URI	/PSIA/AreaControl/CredentialHolders/info		Type	Resource
Function	Used to retrieve configuration of all credential holders and to add a new credential holder			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialHolderInfoList>	
POST		<CredentialHolderInfo>	<ResponseStatus>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialHolderInfoList> is explained with an example below; refer to the XSD schema for details.

John Smith holds credential ID 1 and 2 with card numbers “12345” and “23456” respectively; Rob Lee, who is in disabled state, holds credential ID 3 with card number “34567”.

When a GET request is sent, the reponse will be:

```
<CredentialHolderInfoList version="1.0" xmlns="urn:psialliance-org">
 <CredentialHolderInfo version="1.0">
  <ID>1</ID>
  <Name>Smith, John</Name>
  <GivenName>John</GivenName>
  <Surname>Smith</Surname>
  <State>Active</State>
 </CredentialHolderInfo>
 <CredentialHolderInfo version="1.0">
  <ID>2</ID>
  <Name>Lee, Rob</Name>
  <State>Inactive</State>
 </CredentialHolderInfo>
</CredentialHolderInfoList>
```

This list of credential holders can be obtained using a GET request.

Regarding Names: the Name element is a mandatory single-string element used to simplify presentation of names, and to maximize compatibility with systems which store names as a single string, without dividing them into components. There are no constraints on the formatting of the Name element, but if there is a choice in implementation, implementations should use the format “Surname, GivenName MiddleName”. If the underlying system does support name components, then the implementation should support GivenName, MiddleName, and Surname. And of course if a profile requires it, GivenName MiddleName and Surname must be supported.

A new user can be added by passing <CredentialHolderInfo> as inbound data using a POST request; the local ID assigned to the new credential holder will be returned in the <ResponseStatus>.

1.25.2 /PSIA/AreaControl/CredentialHolders/info/<credentialHolderID>

URI	/PSIA/AreaControl/CredentialHolders/info/<credentialHolderID>		Type	Resource
Function	Used to get or set information of a credential holder as well as to delete a credential holder			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialHolderInfo>	
PUT		<CredentialHolderInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

credentialHolderID will be used as index for the required credential holder.

<CredentialHolderInfo> is explained with an example below; refer to the XSD schema for details.

John Smith is an enabled credential holder.

If credentialHolderID is 1 in a GET request, then the response will be:

```
<CredentialHolderInfo version="1.0" xmlns="urn:psialliance-org">
 <ID>1</ID>
 <Name>Smith, John</Name>
 <GivenName>John</GivenName>
 <Surname>Smith</Surname>
 <State>Active</State>
</CredentialHolderInfo>
```

The information of a credential holder can be obtained using a GET request and updated using a PUT request. A credential holder can be deleted by issuing a DELETE request.

The following is an example of a more complex CredentialHolder object that has a UUID attribute to uniquely identify the individual across various systems as well as a few roles identified by UUID as well.

```
<CredentialHolderInfo xmlns="urn:psialliance-org" version="1.0">
 <ID>1</ID>
 <UID>{3d17502d-c70f-4f35-8e91-82ab9e26ea28}</UID>
 <Name>Doe, John</Name>
 <Description>This is John Doe</Description>
 <State>Active</State>
 <GivenName>John</GivenName>
 <Surname>Doe</Surname>
 <ActiveTill>2014-01-01T00:00:00+05:30</ActiveTill>
 <Disability>False</Disability>
 <RoleIDList>
   <RoleID>
     <GUID>{b3f08942-3d38-4afb-b2cf-ec826ba2d74f}</GUID>
   </RoleID>
   <RoleID>
     <GUID>{b6c46898-4b02-4e41-b050-d2a06a32b97f}</GUID>
   </RoleID>
 <RoleIDList>
```



```
</CredentialHolderInfo>
```

1.25.3 /PSIA/AreaControl/CredentialHolders/state/<credentialHolderID>

URI	/PSIA/AreaControl/CredentialHolders/state/<credentialHolderID>		Type	Resource
Function	Used to retrieve or update state of a particular credential holder			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialHolderState>	
PUT	[required] activeState		<ResponseStatus>	

Notes:

activeState value can be either 'Active' or 'Inactive'.

<CredentialHolderState> is explained with an example below; refer to the XSD schema for details.

John Smith's credential holder ID is 1 and his account is disabled. If a GET request is sent with credentialHolderID as 1, then the response will be:

```
<CredentialHolderState version="1.0" xmlns="urn:psialliance-org">
  <State>Inactive</State>
</CredentialHolderState>
```

The credential holder state can be obtained using a GET request and updated using a PUT request.

1.25.4 /PSIA/AreaControl/CredentialHolders/track/<credentialHolderID>

URI	/PSIA/AreaControl/CredentialHolders/track/<credentialHolderID>		Type	Resource
Function	Used to track a credential holder in an access control system. This retrieves the report of granted and denied access events for a given credential holder.			
Methods	Query String(s)	Inbound Data	Return Result	
GET	[required] fromTime [required] toTime		<CredentialHolderTrackList>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

fromTime and toTime take strings which are in the xs:datetime format.

<CredentialHolderTrackList> is explained with an example below; refer to the XSD schema for details.

John Smith (credential holder ID 1) who holds a credential with credential ID 1 (card number "12345") has gained access to Area X at 10:00 AM through Portal 1 and exited at 10:10 AM via Portal 2.

When a GET request is sent with credential holder ID as 1, fromTime as 2011-01-01T09:00:00+05:30 and toTime as 2011-01-01T10:05:00+05:30, then the response will be:

```
<CredentialHolderTrackList version="1.0" xmlns="urn:psialliance-org">
  <CredentialHolderTrack>
    <TrackTime>2011-01-01T10:00:00+05:30</TrackTime>
    <PartitionID><ID>1</ID></PartitionID>
    <PortalID><ID>1</ID></PortalID>
    <CredentialID><ID>1234</ID></CredentialID>
    <AccessState>Granted</AccessState>
  </CredentialHolderTrack>
</CredentialHolderTrackList>
```

1.25.5 /PSIA/AreaControl/CredentialHolders/count

URI	/PSIA/AreaControl/CredentialHolders/count		Type	Resource
Function	Used to count the available CredentialHolders in an access control system.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<CredentialHolderInfoList>	

Notes:

This resource shall only be available via encrypted transport mechanism (HTTPS). If a request is made via unencrypted HTTP, the host should return a HTTP 426 Upgrade header.

<CredentialHolderInfoList> is explained with an example below; refer to the XSD schema for details.

When a GET request is sent then the expected response will be:

```
<CredentialHolderInfoList version="1.0" xmlns="urn:psialliance-org" countApplied="2"/>
```

Please note that no CredentialHo,ders should be returned in this call.

1.26 /PSIA/AreaControl/Roles

1.26.1 /PSIA/AreaControl/Roles/info/<roleUID>

URI	/PSIA/AreaControl/Roles/info/<roleUID>		Type	Resource
Function	Used to get or set information of a role as well as to delete a role			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<RoleInfo>	
PUT		<RoleInfo>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes:

roleUID will be used as index for the required role in the form of a UUID (Global ID)

If roleUID is {05FE374C-C98D-4D81-B936-D4BF698BCF27} in a GET request, then the response will be:

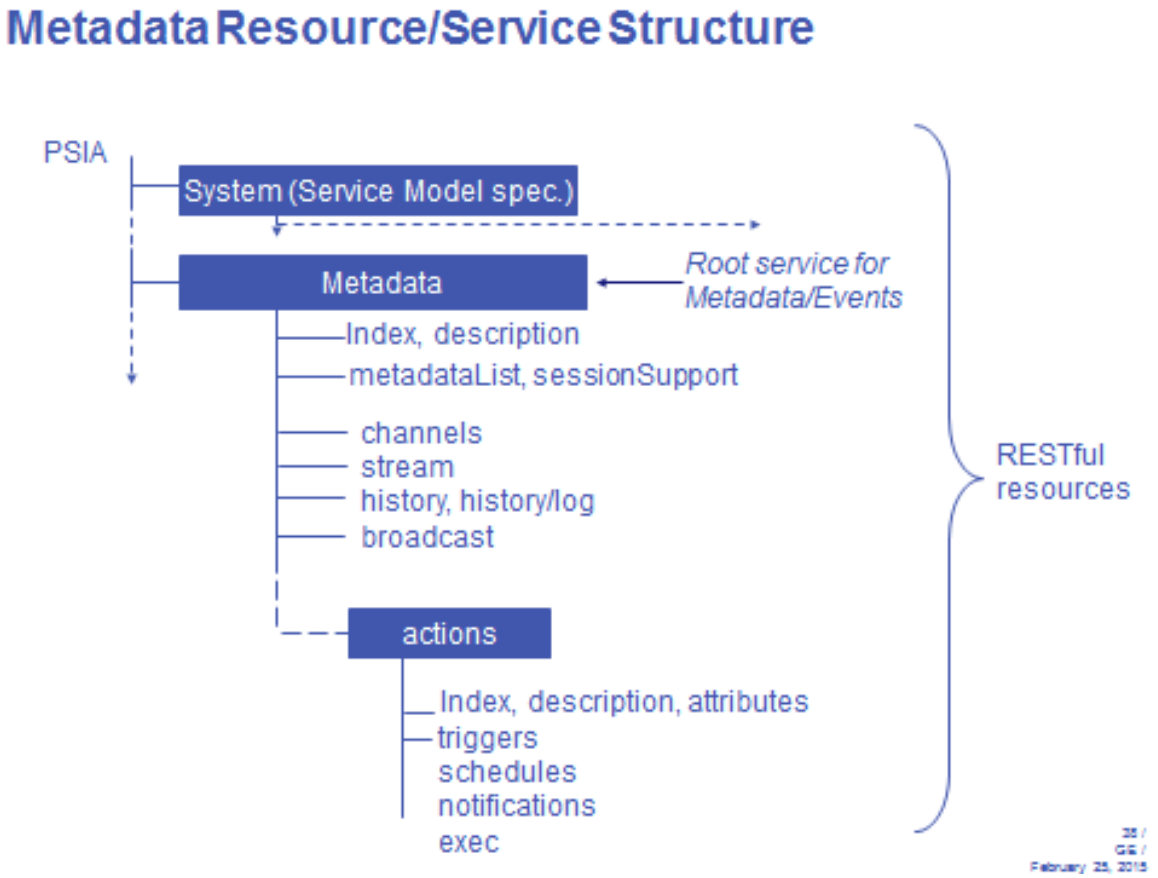
```
<RoleInfo version="1.0" xmlns="urn:psialliance-org">
  <ID>1</ID>
  <UID>{05FE374C-C98D-4D81-B936-D4BF698BCF27}</UID>
  <Name>Role {05FE374C-C98D-4D81-B936-D4BF698BCF27}</Name>   <PermissionIDList>
   <PermissionID>
     <ID>1</ID>
   </PermissionID>
  </PermissionIDList>
</RoleInfo>
```

The information of a role can be obtained using a GET request and updated using a PUT request. A role can be deleted by issuing a DELETE request.

A PUT to Role that completely omits a <PermissionIDList> element should have the effect of leaving the existing PermissionIDList as-is.

Service details for Metadata and Events

1.27 Services and Resources Overview



Some, but not all, of the information within the PSIA CMEM specification is repeated here in this document for the sake of completeness.

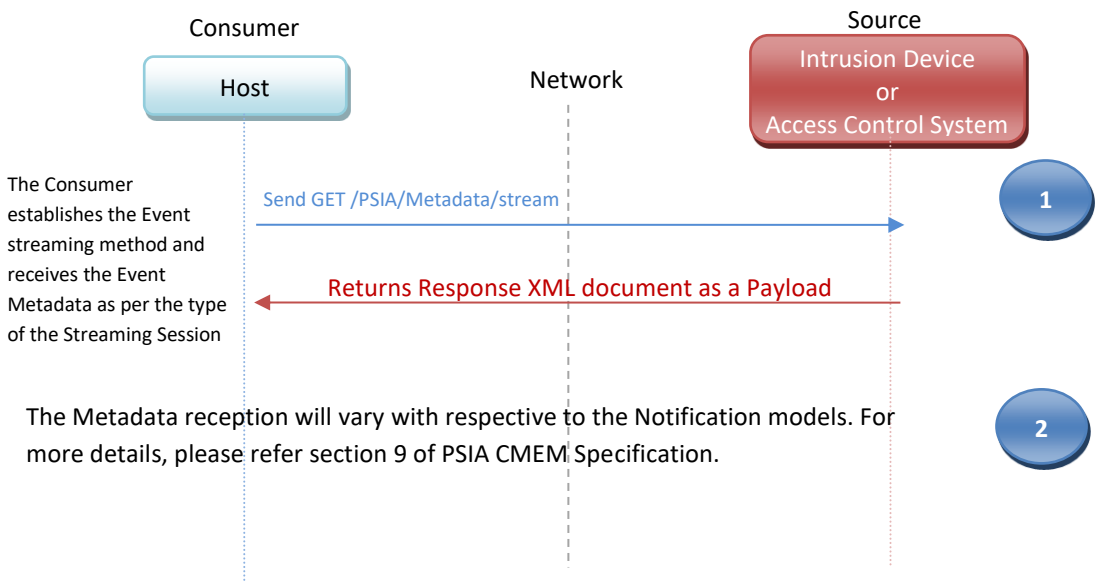
The above Metadata service and resources are described in the table below (refer PSIA CMEM Specification for more details):

Name	Type	Description
Metadata	Service	Base Service resource for all the functional objects within the Metadata Service hierarchy.
index	resource	PSIA defined resource that lists the child-level resources within a service.
description	resource	PSIA-defined resource that describes the functional attributes of a service/resource.
metadataList	resource	Metadata resource that describes all of the active Metadata/Event types active on a particular device.

sessionSupport	resource	Metadata resource that defines all of the transport, format and session parameters offered by a device for transferring metadata information.
channels	resource	Metadata resource that contains all of the attributes and configuration information for all metadata/event input channels to a device or system.
stream	resource	Metadata resource that acts as the access point for creating metadata/event data streams.
broadcasts	resource	If a source node indicates in its 'sessionSupport' properties that it supports multicast sessions for metadata, then this resource object contains the list of active multicast sessions along with their session attributes.
Actions	Service	Metadata service that provides the ability to query, configure and subscribe to specific actions/notifications offered by a device's metadata service for asynchronous 'push' notification using non-PSIA protocol methods (e.g. Email, FTP, etc.)

1.27.1 Establishing the Streaming

The following diagram illustrates the sequence of operations involved in establishing event streaming with an intrusion device or access control system:



1.27.1.1 Step 1: Starting Streaming session

An intrusion device or access control System will start streaming or provide the events when it receives a GET or POST REST request for the URI /PSIA/Metadata/stream from Host application.

The /PSIA/Metadata/stream object is the session setup object for the PSIA Metadata resource hierarchy. All session parameters are 'set' with the 'stream' object for all HTTP/REST managed sessions with an accompanying "MetaSessionParms" schema instance, to setup a metadata stream session of some specified flavor.

URI	/PSIA/Metadata/stream		**Type**	Resource
Requirement Level	- All -			
Function	This resource is the access point on a Metadata source for setting-up Metadata/Event sessions.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET	Conditional	-None-	<ResponseStatus> OR For GETs with Asynch Stream IDs: <MetaSessionParms>	
PUT	N/A	<TBD>	<ResponseStatus w/error code>	
POST	Conditional	<MetaSessionParms>	<ResponseStatus> OR For GETs with Asynch Stream IDs: <MetaSessionParms>	
DELETE	None	None	Delete is only used in cases: A) A consumer wants to terminate the current synchronous session after setting parameters for non-synchronous session B) A consumer wants to delete the session resource instance for an asynchronous HTTP/REST session. In this case, the original session ID MUST be supplied as part of the resource URI (e.g. DELETE /Metadata/stream/9)	
	This Metadata resource is mainly the access point for establishing ('GET'ing) a Metadata/Event stream. Refer CMEM section 10.2.4 for more detail.			

Example: An access control system supports the following events:

Domain: psialliance.org, Class: AreaControl.System, Type: deviceState.*, connectionState.*, batteryPowerState.*, and acPowerState.* (Refer section "12.3.1.5 Step 5: Receiving events" for a list of Domain-Class-Type reserved by Area Control workgroup).

Domain: psialliance.org, Class: AreaControl.Portal, Type: latchState.*, portalOpenState.*, portalForcedState.*, portalHeldState.*, access.granted and access.denied.

The system allows consumers to apply a filter so that events from selected portals can be subscribed to. So, the system publishes events of class AreaControl.System on channel 1 and events of class AreaControl.Portal on channel 2. Further, within the filter on selected portals, the system allows

consumer to apply an additional filter on credential ID's such that consumer may subscribe to access events of selected credentials on selected portals.

The main door of the facility protected by this access control system has In and Out portals 41 and 51 respectively.

If an operator needs to subscribe to all events from the system, MetaSessionParms would be:

```
<MetaSessionParms version="1.1">
 <MetaXportParms>
  <metaSessionID>0</metaSessionID>
  <metaFormat>xml-psia</metaFormat>
  <metaSessionType>RESTSyncSessionTargetSend</metaSessionType>
  <metaSessionFlowType>dataStream</metaSessionFlowType>
 </MetaXportParms>
</MetaSessionParms>
```

****Note, if an unsolicited GET is called on /PSIA/Metadata/stream, then the above MetaSessionParms are assumed.*

If a time and attendance application needs to subscribe only to granted and denied events at main door (portals 41 and 51), MetaSessionParms would be:

```
<MetaSessionParms version="1.1">
 <MetaXportParms>
  <metaSessionID>0</metaSessionID>
  <metaFormat>xml-psia</metaFormat>
  <metaSessionType>RESTSyncSessionTargetSend</metaSessionType>
  <metaSessionFlowType>dataStream</metaSessionFlowType>
  <metadataNameList>
   <metadataIDString>/psialliance.org/AreaControl.Portal/access.denied</metadataIDString>
   <metadataIDString>/psialliance.org/AreaControl.Portal/access.granted</metadataIDString>
  </metadataNameList>
  <metadataXChannelList>
   <metaXChannel>2/41,51</metaXChannel>
  </metadataXChannelList>
 </MetaXportParms>
</MetaSessionParms>
```

If a tracking application needs to subscribe only to granted and denied events for a couple of cards (Credentials with ID 12 & 13 respectively) at (portals 41 & 51) and another card (with credential ID 1024) at all portals, MetaSessionParms would be:

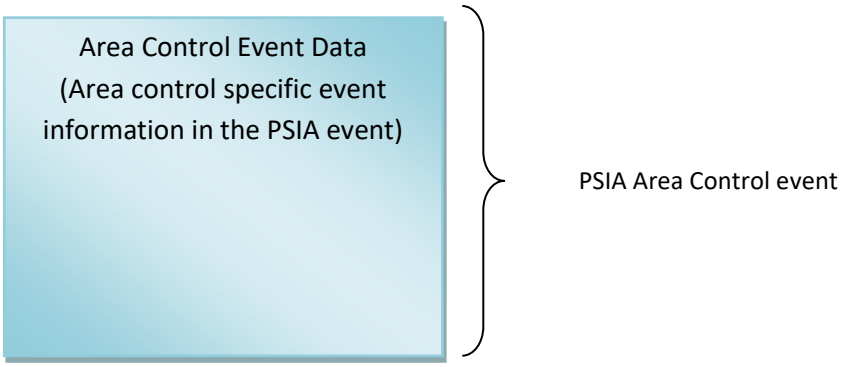
```
<MetaSessionParms version="1.1">
 <MetaXportParms>
  <metaSessionID>0</metaSessionID>
  <metaFormat>xml-psia</metaFormat>
  <metaSessionType>RESTSyncSessionTargetSend</metaSessionType>
  <metaSessionFlowType>dataStream</metaSessionFlowType>
  <metadataNameList>
   <metadataIDString>/psialliance.org/AreaControl.Portal/access.denied</metadataIDString>
   <metadataIDString>/psialliance.org/AreaControl.Portal/access.granted</metadataIDString>
  </metadataNameList>
```



```
    <metadataXChannelList>
     <metaXChannel>2/41,51/12,13</metaXChannel>
     <metaXChannel>2//1024</metaXChannel>
    </metadataXChannelList>
   </MetaXportParms>
  </MetaSessionParms>
```

1.27.1.2 Step 2: Receiving events

All events received from area control devices/systems will follow the data structure shown below:



The event structure is explained below.

Common Metadata/Event Header Fields:

The following table details fields in the common Metadata header

Field	Description
Version	Metadata format version/revision
Metadata ID String (MIDS)	Domain (format)/Class/Type of the corresponding Metadata/Event (see below) Also known as "**metaID**" For Area Control, the metaID is: **Domain** psialliance.org The following 'class' and 'type' fields/tags have been reserved within this Metadata Class. The owning PSIA working group for these metadata classes and types is the ACWG. Note: The table below lists the possible Domain-Class-Type combinations reserved by Area Control workgroup. In general, there is a 1-to-1 correspondance with the various enum elements in the *Status resources. In some cases, like ArmState – ArmType and IntrusionAlarmState – IntrusionAlarmType, the MIDS Type corresponds potentially to a combination of enums. Access granted and denied put the detailed info about the grant/deny into the ValueState.

Domain: psialliance.org		
Type in MIDS or GMCH	**Corresponding XML enum type in Status**	**Corresponding XML enum values in Status, Details**
Class: AreaControl.System		
tamperState.ok tamperState.other tamperState.tamper tamperState.unknown	TamperState	OK Other Tamper Unknown
maintenanceState.ok maintenanceState.other maintenanceState.maintenance maintenanceState.unknown	MaintenanceState	OK Other Maintenance Unknown
deviceState.ok deviceState.other deviceState.failed deviceState.unknown	DeviceState	OK Other Failed Unknown
connectionState.ok connectionState.other connectionState.disconnected connectionState.unknown	ConnectionState	OK Other Disconnected Unknown
acPowerState.ok acPowerState.other acPowerState.failed acPowerState.unknown	ACPowerState	OK Other Failed Unknown
batteryPowerState.ok batteryPowerState.other batteryPowerState.failed batteryPowerState.lowBattery batteryPowerState.unknown	BatteryPowerState	OK Other Failed LowBattery Unknown
Class: AreaControl.Partition		
armState.other armState.armed.other armState.armed.away armState.armed.stay armState.armed.unknown armState.disarmed armState.unknown	ArmState - ArmType	Other Armed - Other Armed - Away Armed - Stay Armed - Unknown Disarmed Unknown
readyState.other readyState.notReady readyState.ready readyState.unknown	ArmReadyState	Other NotReady Ready Unknown
alarmState.ok alarmState.other alarmState.alarm.other alarmState.alarm.panic alarmState.alarm.tamper alarmState.alarm.intrusion alarmState.alarm.fire alarmState.alarm.duress alarmState.alarm.technical alarmState.alarm.environmental alarmState.alarm.unknown alarmState.unknown	IntrusionAlarmState - IntrusionAlarmType	OK Other Alarm - Other Alarm - Panic Alarm - Tamper Alarm - Intrusion Alarm - Fire Alarm - Duress Alarm - Technical Alarm - Environmental Alarm - Unknown Unknown Partition Member (zone) ID Device ID
faultState.ok faultState.other	IntrusionFaultState	OK Other

	faultState.fault faultState.unknown		Fault Unknown Partition Member (zone) ID
	troubleState.ok troubleState.other troubleState.trouble troubleState.unknown	IntrusionTroubleState	OK Other Trouble Unknown Partition Member (zone) ID Device ID
	Class: AreaControl.Zone		
	connectionState.ok connectionState.other connectionState.disconnected connectionState.unknown	ConnectionState	OK Other Disconnected Unknown
	acPowerState.ok acPowerState.other acPowerState.failed acPowerState.unknown	ACPowerState	OK Other Failed Unknown
	batteryPowerState.ok batteryPowerState.other batteryPowerState.failed batteryPowerState.lowBattery batteryPowerState.unknown	BatteryPowerState	OK Other Failed LowBattery Unknown
	bypassState.other bypassState.active bypassState.bypass bypassState.unknown	BypassState	Other Active Bypass Unknown
	alarmState.ok alarmState.other alarmState.alarm.other alarmState.alarm.panic alarmState.alarm.tamper alarmState.alarm.intrusion alarmState.alarm.fire alarmState.alarm.duress alarmState.alarm.technical alarmState.alarm.environmental alarmState.alarm.unknown alarmState.unknown	IntrusionAlarmState - IntrusionAlarmType	OK Other Alarm - Other Alarm - Panic Alarm - Tamper Alarm - Intrusion Alarm - Fire Alarm - Duress Alarm - Technical Alarm - Environmental Alarm - Unknown Unknown Device ID
	troubleState.ok troubleState.other troubleState.trouble troubleState.unknown	IntrusionTroubleState	OK Other Trouble Unknown Device ID
	faultState.ok faultState.other faultState.fault faultState.unknown	IntrusionFaultState	OK Other Fault Unknown
	Class: AreaControl.Portal		
	connectionState.ok connectionState.other connectionState.disconnected connectionState.unknown	ConnectionState	OK Other Disconnected Unknown
	acPowerState.ok acPowerState.other acPowerState.failed	ACPowerState	OK Other Failed

	acPowerState.unknown		Unknown
	batteryPowerState.ok batteryPowerState.other batteryPowerState.failed batteryPowerState.lowBattery batteryPowerState.unknown	BatteryPowerState	OK Other Failed LowBattery Unknown
	latchState.other latchState.latched latchState.unlatched latchState.unknown	LatchState	Other Latched Unlatched Unknown
	portalOpenState.other portalOpenState.open portalOpenState.closed portalOpenState.unknown	PortalOpenState	Other Open Closed Unknown
	portalForcedState.other portalForcedState.ok portalForcedState.forced portalForcedState.unknown	PortalForcedState	Other OK Forced Unknown
	portalHeldState.other portalHeldState.ok portalHeldState.held portalHeldState.unknown	PortalHeldState	Other OK Held Unknown
	portalAlarmState.other portalAlarmState.ok portalAlarmState.alarm portalAlarmState.unknown	PortalAlarmState	Other OK Alarm Unknown
	accessOverrideState.other accessOverrideState.normal accessOverrideState.overridden accessOverrideState.unknown	AccessOverrideState	Other Normal Overridden Unknown
	forcedMaskState.other forcedMaskState.masked forcedMaskState.monitored forcedMaskState.unknown	MaskState	Other Masked Monitored Unknown
	heldMaskState.other heldMaskState.masked heldMaskState.monitored heldMaskState.unknown	MaskState	Other Masked Monitored Unknown
	tamperState.ok tamperState.other tamperState.tamper tamperState.unknown	TamperState	OK Other Tamper Unknown
	access.granted	AccessGrantedReason	In EventValueState: Other OK Duress
	access.requested		{CredentialInfo}
	access.denied	AccessDeniedReason	In EventValueState: Other UnknownCredential InvalidCredential InactiveCredential ExpiredCredential LostCredential

			StolenCredential InvalidFacilityCode InvalidIssueCode AuthenticationTimeout AuthenticationFailure MaxRetriesReached InactiveCredentialHolder ExpiredCredentialHolder NotPermitted NotPermittedAtThisTime AntiPassback AccessOverridden NoAsset NoEscort OccupancyLimitReached CredentialNotPresented UseLimitReached PartitionClosed Unauthorized BiometricMismatch InvalidPIN AccessOverridden
	Class: AreaControl.Input		
	connectionState.ok connectionState.other connectionState.disconnected connectionState.unknown	ConnectionState	OK Other Disconnected Unknown
	acPowerState.ok acPowerState.other acPowerState.failed acPowerState.unknown	ACPowerState	OK Other Failed Unknown
	batteryPowerState.ok batteryPowerState.other batteryPowerState.failed batteryPowerState.lowBattery batteryPowerState.unknown	BatteryPowerState	OK Other Failed LowBattery Unknown
	maskState.other maskState.masked maskState.monitored maskState.unknown	MaskState	Other Masked Monitored Unknown
	faultState.ok faultState.other faultState.cut faultState.short faultState.unknown	InputSupervisionFaultState	OK Other Cut Short Unknown
	state.normal state.active state.other state.unknown	InputState	Normal Active Other Unknown
	tamperState.ok tamperState.other tamperState.tamper tamperState.unknown	TamperState	OK Other Tamper Unknown
	Class: AreaControl.Output		
	connectionState.ok connectionState.other connectionState.disconnected connectionState.unknown	ConnectionState	OK Other Disconnected Unknown
	acPowerState.ok acPowerState.other	ACPowerState	OK Other

	acPowerState.failed acPowerState.unknown		Failed Unknown
	batteryPowerState.ok batteryPowerState.other batteryPowerState.failed batteryPowerState.lowBattery batteryPowerState.unknown	BatteryPowerState	OK Other Failed LowBattery Unknown
	faultState.ok faultState.other faultState.fault faultState.unknown	OutputSupervisionFaultState	OK Other Fault Unknown
	state.off state.on state.other state.unknown	OutputState	Off On Other Unknown
	Class: AreaControl.Holiday		
	activeState.other activeState.inactive activeState.active activeState.unknown	ActiveState	Other Inactive Active Unknown
	Class: AreaControl.TimeSchedule		
	activeState.other activeState.inactive activeState.active activeState.unknown	ActiveState	Other Inactive Active Unknown
	Class: AreaControl.Credential		
	state.inactive state.active state.expired	StateOfCredential	Inactive Active Expired
	assigned.assigned	ReferenceID	{ReferenceID of CredentialHolder being assigned. ID=0 means unassigned}
	Class: AreaControl.CredentialHolder		
	state.active state.inactive	StateOfCredentialHolder	Active Inactive
Source ID	UUID/GUID of the metadata/event source (ISO/IEC 9834-8, ITU X.667)		
Source's Local ID (LID)	Local ID of the Partition, Portal, Zone, Input, or Output		
Time	Absolute time of the generation of the metadata/event info in "xs:dateTime" format		
Priority	Needed for differentiating events in a multi-domain environment		
Link	Multi-event correlation/reference/linkage ID (UUID/GUID)		

Area Control Event Data fields:

The following table details fields in the PSIA Metadata Area Control Event Data:

Field	Description
Value	The value associated with the event
InputID	(List of) Input ID (Optional)
OutputID	(List of) Output ID (Optional)
PortalID	(List of) Portal ID (Optional)
ZoneID	(List of) Zone ID (Optional)

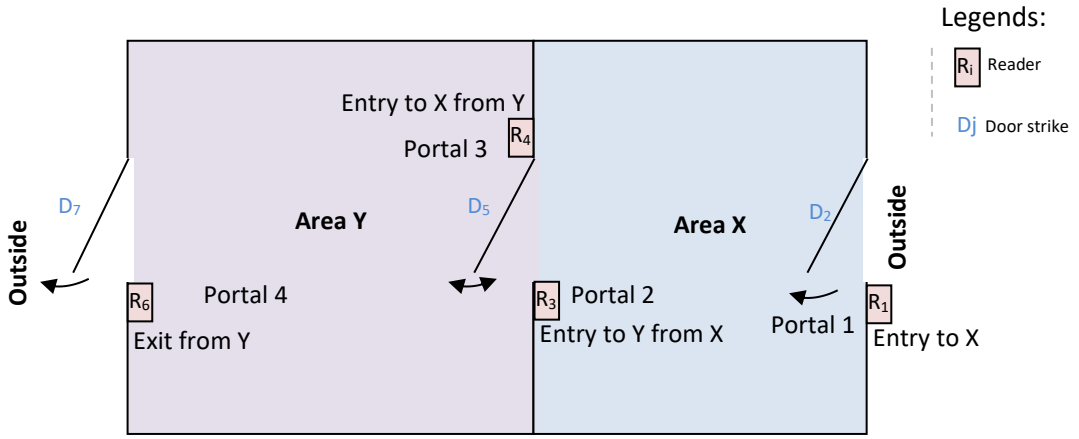
HolidayID	(List of) Holiday ID (Optional)
TimeScheduleID	(List of) TimeSchedule ID (Optional)
PermissionID	(List of) Permission ID (Optional)
CredentialNumber	(List of) Credential Identifier Number (Optional)
CredentialInfo	(List of) CredentialInfo (Optional)
CardFormatID (list or single)	(List of) CardFormats ID (Optional)
CardFormatInfoList or (WiegandCardFormatInfo/ MagstripeCardFormatInfo)	(List of) CardFormatInfoList or single WiegandCardFormatInfo/ MagstripeCardFormatInfo (Optional)
PartitionID (list or single)	(List of) Partition ID (Optional)
PartitionMemberID (list or single)	(List of) Zone or Access Point ID (Optional) Used in case of alarm, trouble or fault on partition
DeviceID (list or single)	(List of) Device ID (Optional) Used in case of alarm, trouble or fault on partition or zone
CredentialID (list or single)	(List of) Credential ID (Optional) Used in case of access events/alarms
CredentialHolderID (list or single)	(List of) Credential holder ID (Optional) Used in case of access events/alarms
Information	Additional details/description of the event

<AreaControlEvent> is explained with examples below; refer to the XSD schema for details.

If an intrusion alarm is generated by Input 4 of Zone 2 in Partition 1 of an intrusion panel, the received event XML will be:

```
<AreaControlEvent version="1.0" xmlns="urn:psialliance-org">
 <MetadataHeader xmlns="urn:psialliance-org">
  <MetaVersion>1.0</MetaVersion>
  <MetaID>/psialliance.org/AreaControl.Partition/alarmState.intrusion/1</MetaID>
  <MetaSourceID>{DF5768C8-E695-4315-A06E-AF49E1409654}</MetaSourceID>
  <MetaSourceLocalID>1</MetaSourceLocalID>
  <MetaTime>2011-01-01T00:00:00+05:30</MetaTime>
  <MetaPriority>3</MetaPriority>
 </MetadataHeader>
 <EventData>
  <InputIDList>
   <InputID>
    <ID>4</ID>
   </InputID>
  </InputIDList>
  <ZoneIDList>
   <ZoneID>
    <ID>2</ID>
   </ZoneID>
  </ZoneIDList>
  <Info>Intrusion alarm reported at Input 4 of Zone 2 in Partition 1</Info>
 </EventData>
</AreaControlEvent>
```


Let's consider events from access control systems next. Below diagram is a simplfied access control floor plan.



This access acontrol system has two access areas X and Y and 4 portals (each consisting of a reader and a door strike) as described below:

Portal 1 is used to enter Area X from Outside
Portal 2 is used to enter Area Y from Area X
Portal 3 is used to enter Area X fromArea Y
Portal 4 is used to exit Area Y to Outside

David Brown (Credential holder ID 20) holding credential ID 4 with card number '23456' is not allowed to enter area Y (area ID 2) during office hours (8 AM to 6 PM) but he tried to access area Y through portal 2 at 2 PM and was denied access.

The received event XML will be:


```
<AreaControlEvent version="1.0" xmlns="urn:psialliance-org">
 <MetadataHeader>
  <MetaVersion>1.0</MetaVersion>
  <MetaID>/psialliance.org/AreaControl.Portal/access.denied/2</MetaID>
  <MetaSourceID>{AF5768C8-E695-4315-D06E-AF49E1409654}</MetaSourceID>
  <MetaSourceLocalID>2</MetaSourceLocalID>
  <MetaTime>2011-01-01T14:00:00+05:30</MetaTime>
  <MetaPriority>4</MetaPriority>
 </MetadataHeader>
 <EventData>
  <ValueState>
   <Denied>NotPermittedAtThisTime</Denied>
  </ValueState>
  <CredentialNumberList>
   <CredentialNumber>
    <Number>23456</Number>
   </CredentialNumber>
  </CredentialNumberList>
  <CredentialIDList>
   <CredentialID>
    <ID>4</ID>
   </CredentialID>
```



```
  </CredentialIDList>
  <CredentialHolderIDList>
   <CredentialHolderID>
    <ID>20</ID>
   </CredentialHolderID>
  </CredentialHolderIDList>
  <Info>Entry restricted during office hours</Info>
 </EventData>
</AreaControlEvent>
```

Note that any metadata including detailed credential or credentialholder information must be made via an encrypted transport as documented in CMEM and CSEC. Specifically, while a CredentialID and/or CredentialHolder ID may be transmitted unencrypted, a CredentialNumber (referring to the card's encoded value) or CredentialHolderCoreInfo values may not be transmitted in plain text and should be omitted from metadata streams that are not encrypted.

When using the HostManagedAccessGrant option, an access request may look like the following:

```
<AreaControlEvent version="1.0" xmlns="urn:psialliance-org">
 <MetadataHeader>
  <MetaVersion>1.0</MetaVersion>
  <MetaID>/psialliance.org/AreaControl.Portal/access.requested/2</MetaID>
  <MetaSourceID>{AF5768C8-E695-4315-D06E-AF49E1409654}</MetaSourceID>
  <MetaSourceLocalID>2</MetaSourceLocalID>
  <MetaTime>2011-01-01T14:00:00+05:30</MetaTime>
  <MetaPriority>1</MetaPriority>
 </MetadataHeader>
 <EventData>
  <ValueState>
     <Requested>
       <CredentialInfo version="1.0" xmlns="urn:psialliance-org">
       <ID>1</ID>
       <Name>Credential 1</Name>
       <Description>Assigned to David</Description>
       <AssignedToID><ID>1467</ID></AssignedToID>
       <State>Active</State>
       <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
       <ValidFrom>2011-01-01T00:00:00+05:30</ValidFrom>
       <IdentifierInfoList>
         <IdentifierInfo>
           <Type>Card</Type>
           <Value>12345</Value>
         <ValueEncoding>Decimal</ValueEncoding>
           <Format>
             <ID>1</ID>
             <Name>32 bit</Name>
           </Format>
         </IdentifierInfo>
       </IdentifierInfoList>
       <RawValue>101101110100000</RawValue>
      </CredentialInfo>
    </Requested>
  </ValueState>
  <Info>Credential 1 requests access to portal 2.  Please reply to
/PSIA/AreaControl/PartitionMemebers/Portals/AccessResponse/2 with your answer</Info>
 </EventData>
</AreaControlEvent>
```


Credential assignments to CredentialHolders will be passed back using events like the following. Note the accompanying CredentialInfo is required.


```
<AreaControlEvent version="1.0" xmlns="urn:psialliance-org">
 <MetadataHeader>
  <MetaVersion>1.0</MetaVersion>
  <MetaID>/psialliance.org/AreaControl.Credential/assigned.assigned/{e53c9f38-3211-41c8-bd07-61b110e08a7b}</MetaID>
  <MetaSourceID>{AF5768C8-E695-4315-D06E-AF49E1409654}</MetaSourceID>
  <MetaSourceLocalID>168</MetaSourceLocalID>
  <MetaTime>2011-01-01T14:00:00+05:30</MetaTime>
  <MetaPriority>1</MetaPriority>
 </MetadataHeader>
 <EventData>
  <CredentialHolderID>
        <ID>1014</ID>
        <UID>{9d725bb7-5df8-4443-9518-d35b4a2efe72}</UID>
        <Name>Credential {e53c9f38-3211-41c8-bd07-61b110e08a7b}, localid 168, is being assigned to Credentialholder
{9d725bb7-5df8-4443-9518-d35b4a2efe72}, localid 1014</Name>
  </CredentialHolderID>
        <CredentialInfo version="1.0" xmlns="urn:psialliance-org">
        <ID>168</ID>
        <Name>Credential 168</Name>
        <Description>Assigned to David</Description>
        <AssignedToID><ID>1014</ID><UID>{9d725bb7-5df8-4443-9518-d35b4a2efe72}</UID></AssignedToID>
        <State>Active</State>
        <LastModifiedDate>2011-02-16T00:00:00+05:30</LastModifiedDate>
        <ValidFrom>2011-01-01T00:00:00+05:30</ValidFrom>
        <IdentifierInfoList>
         <IdentifierInfo>
          <Type>Card</Type>
          <Value>12345</Value>
         <ValueEncoding>Decimal</ValueEncoding>
          <Format>
           <ID>1</ID>
           <Name>32 bit</Name>
          </Format>
         </IdentifierInfo>
        </IdentifierInfoList>
        <RawValue>101101110100000</RawValue>
       </CredentialInfo>
  <Info>Credential 1 requests access to portal 2. Please reply to
/PSIA/AreaControl/PartitionMemebers/Portals/AccessResponse/2 with your answer</Info>
 </EventData>
</AreaControlEvent>
```


XSD Schemas

The PSIA XSD schemas are published at: http://www.psialliance.org/schemas/

Additional References

1.28 REST

http://en.wikipedia.org/wiki/Representational_State_Transfer

http://www.xml.com/pub/a/2004/12/01/restful-web.html

1.29 HTTP

http://www.w3.org/Protocols/rfc2616/rfc2616.html

http://www.ietf.org/rfc/rfc2617.txt

http://www.ietf.org/rfc/rfc3986.txt

1.30 MIME

http://tools.ietf.org/html/rfc2045

http://tools.ietf.org/html/rfc2046

1.31 XML

http://www.w3.org/TR/xml

http://www.w3.org/TR/xlink

1.32 Data Encodings

http://www.ietf.org/rfc/rfc3548.txt

1.33 Time Format

http://www.ietf.org/rfc/rfc3339.txt

1.34 SHA-1 Hashing Algorithm

http://www.ietf.org/rfc/rfc3174.txt

1.35 ZLIB Compression

http://www.ietf.org/rfc/rfc1950.txt

1.36 DNS-SD

http://www.dns-sd.org/

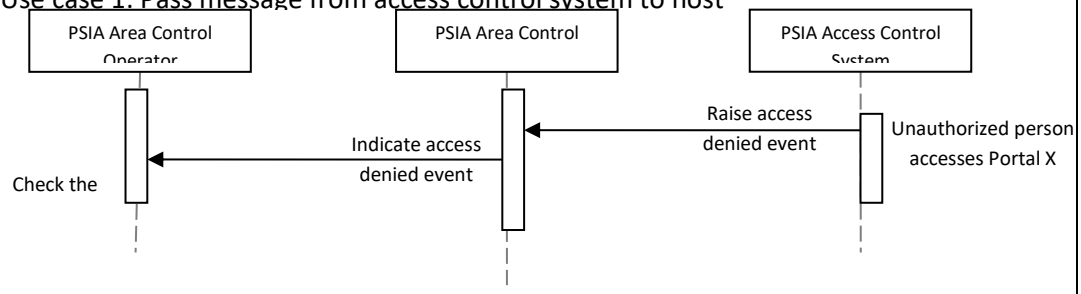
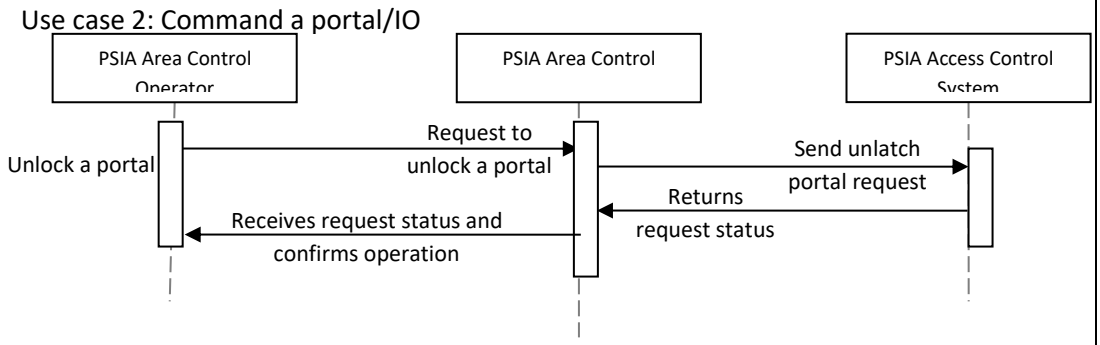
http://developer.apple.com/networking/bonjour/

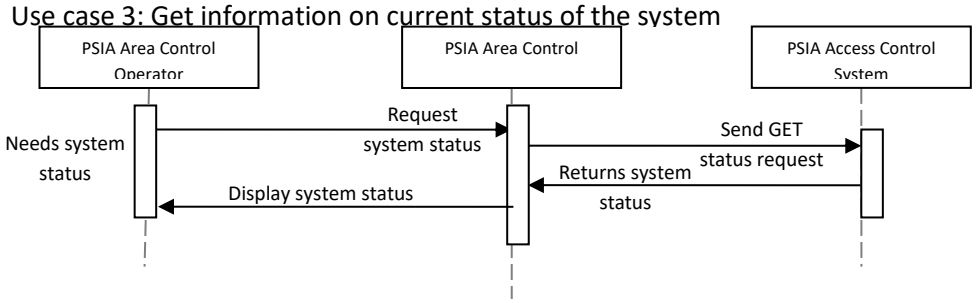
http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt

http://www.dns-sd.org/ServiceTypes.html

Appendix A – Area Control Profiles

1.37 *Baseline Access Control Profile*

Functional Profile Name	Baseline Access Control (BaselineAccessControl)
Functional Profile Version	1.0
Base Specification	Area Control 2.0
Submission Date	June 10, 2013
Functional Profile Description	The Access Control profile provides functionality to perform basic access control tasks.
Justify the need for this functional profile	This profile is the base profile for access control. It provides a platform for which to perform basic access control functionality. It also serves as a firm foundation for more advanced access control functionality delivered via the profile's options.
Sample Use Case(s)	Use case 1: Pass message from access control system to host  Use case 2: Command a portal/IO 

	Use case 3: Get information on current status of the system 
PSIA Topology (for Core Spec Requirements) and CMEM Profile	Peer – Peer LAN CMEM Core Metadata 1.0

Required Operations	Operation	Exception?
	/PSIA/AreaControl/configuration	
	/PSIA/AreaControl/status	
	/PSIA/AreaControl/PartitionMembers/Portals/info	
	/PSIA/AreaControl/PartitionMembers/Portals/status	
	/PSIA/AreaControl/PartitionMembers/Portals/instanceDescription/<portalID>	
	/PSIA/AreaControl/PartitionMembers/Portals/status/<portalID>	
	/PSIA/AreaControl/PartitionMembers/Portals/latchState/<portalID>	
	/PSIA/AreaControl/PartitionMembers/Portals/forcedMaskState/<portalID>	
	/PSIA/AreaControl/PartitionMembers/Portals/heldMaskState/<portalID>	
	/PSIA/AreaControl/Devices/Inputs/info	
	/PSIA/AreaControl/Devices/Inputs/status	
	/PSIA/AreaControl/Devices/Inputs/instanceDescription/<inputID>	
	/PSIA/AreaControl/Devices/Inputs/status/<inputID>	
	/PSIA/AreaControl/Devices/Inputs/maskState/<inputID>	
	/PSIA/AreaControl/Devices/Outputs/info	
	/PSIA/AreaControl/Devices/Outputs/status	
	/PSIA/AreaControl/Devices/Outputs/instanceDescription/<outputID>	
	/PSIA/AreaControl/Devices/Outputs/status/<outputID>	
	/PSIA/AreaControl/Devices/Outputs/state/<outputID>	
	/PSIA/AreaControl/Permissions/info	
	/PSIA/AreaControl/Credentials/info	
	/PSIA/AreaControl/Credentials/state/<credentialID>	

Metadata:

- AreaControl.System/*

	• AreaControl.Portal/latchState.*, AreaControl.Portal/portalOpenState.*, AreaControl.Portal/portalForcedState.*, AreaControl.Portal/portalHeldState.*, AreaControl.Portal/forcedMaskState.*, AreaControl.Portal/heldMaskState.* AreaControl.Portal/tamperState.*, AreaControl.Portal/access.granted, AreaControl.Portal/access.denied • AreaControl.Input/* • AreaControl.Output/*

Required Types/Fields	Field	Exception?
Only fill out if there are exceptions. If no exceptions, say NO EXCEPTIONS	Type -	
	Type -	

NO EXCEPTIONS

1.37.1 Baseline Access Control Profile – Flexible Authentication Option

Option Name	Flexible Authentication Option (BaselineAccessControl-FlexibleAuthenticationOption)
Option Version	1.0
Option Description	The Flexible Authentication Option allows a host to change what identifiers should be required at runtime (i.e. card only, card+pin, card or pin, etc)
Justify the need for this option	There are scenarios where a customer may wish to routinely change which identifiers be required at a portal.
Sample Option Use Case(s)	1) A customer wishes for only cards to be required during business hours, but for card+pin to be required after business hours.
Required Operations (in addition to what is included from base profile)	Operation: /PSIA/AreaControl/PartitionMembers/Portals/requiredCredentialTypes/<portalID> — Exception?: Metadata: none
Required Types/Fields (in addition to what is included from base profile). *Only fill out if there are exceptions. If*	NO EXCEPTIONS

no exceptions, say NO EXCEPTIONS	

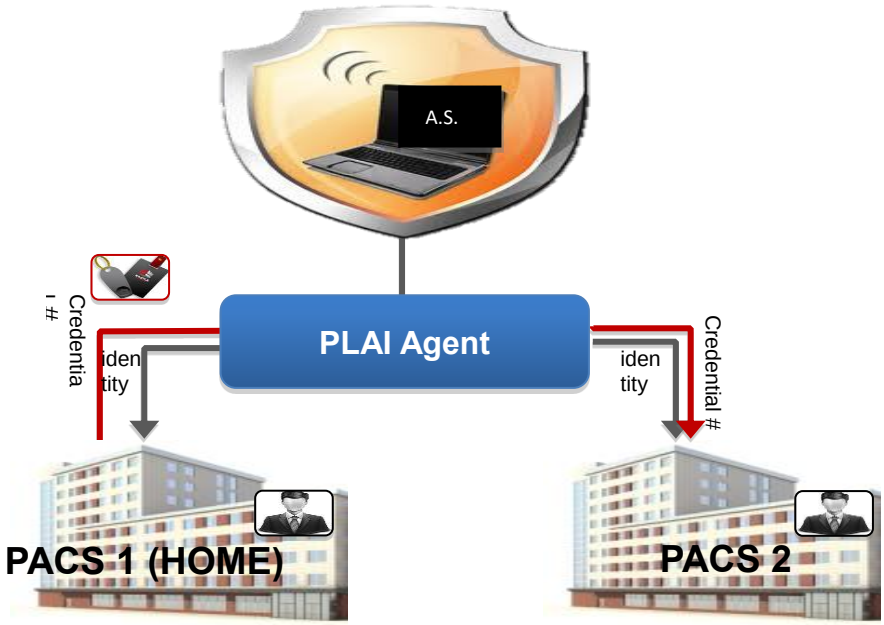
1.37.2 Baseline Access Control Profile – Host Managed Access Grant Option

Option Name	Host Managed Access Grant Option (BaselineAccessControl-HostManagedAccessGrantOption)
Option Version	1.0
Option Description	The Host Managed Access Grant Option allows real-time decision making about access control to be done at the host for added functionality for users with multiple access control devices (i.e. medium to large enterprises).
Justify the need for this option	The requirements of the Area Control specification entail having credentials and credential holders being loaded onto the device which would not be applicable in a scenario where decisions are being made on the host.
Sample Option Use Case(s)	2) A garage owner with multiple garage facilities has made an agreement where n number of users from a given company can use their garages per day. The garage owner wants the access control company (host) to enforce that and ensure that the company is not issuing additional cards across different locations. This profile allows the host to give realtime approval for cards within an organization and if a threshold has been reached it can stop granting access to the cards without updating a whole company's cards on the access controllers multiple times (both to revoke access and to re-grant access the next day). 3) A customer wants to employ anti-passback across access panels. In this way, the host can track which zone a user is in and make such decisions. This situation is applicable in a campus setting (multiple buildings grouped together) or an enterprise environment where a corporation has multiple offices across a city.
Required Operations (in addition to what is included from base profile)	Operation / Exception? /PSIA/AreaControl/PartitionMembers/Portals/accessResponse/<portalID> / Metadata: • AreaControl.Portal/access.requested ○ Permissions are not required to be sent in CredentialInfo.
PSIA Topology (for Core Spec Requirements) and CMEM Profile	Peer – Peer LAN CMEM Low Latency Profile Option 1.0

Required Types/Fields (in addition to what is included from base profile).	NO EXCEPTIONS

1.38 Physical Logical Access Interoperabiliy (PLAI) Profile

1.38.1 PLAIBase

Profile Name	*Physical Logical Access Interoperability Base Profile ()*
Profile Version	*1.0*
Base Specification	Area Control 3.0
Submission Date	April 22, 2014
Profile Description	The PLAI profile provides basic commands for Physical-Logical Access Interoperability for Identities and Credentialholders.
Justify the need for this profile	This profile is the base profile for Physical-Logical Access Interoperability with the aim to unify physical and logical access control systems and have LDAP as single source (Authoritative Source) for identities.
Sample Profile Use Case(s)	

	1. Authoritative Source (AS) pushes the identity to multiple PACS via PLAI Agent (CredentialHolders). 2. PACS 1 provides the credential number back to PLAI Agent. 3. PLAI Agent pushes the credential to PACS 2 (and any additional PACS that may be registered).
PSIA Topology (for Core Spec Requirements) and CMEM Profile	Peer – Peer Internet CMEM Core Metadata 1.0
Required Operations	

Operation	Exception?
/PSIA/AreaControl/Credentials/info	Y
/PSIA/AreaControl/Credentials/info	
/PSIA/AreaControl/Credentials/info/<credentialUID>	
/PSIA/AreaControl/Credentials/state/<credentialUID>	
/PSIA/AreaControl/CredentialHolders/info	Y
/PSIA/AreaControl/CredentialHolders/info/<credentialHolderUID>	
/PSIA/Metadata/Stream	
/PSIA/profile	
/PSIA/CSEC/deviceOwnership	

Metadata:

- AreaControl.Credential/assigned.*
  - Every Credential/assigned event requires the inclusion of CreadentialInfo, however the Permissions do not need to be conveyed in the CredentialInfo as they are PACS-specific.

Exceptions:

All exceptions refer to the fact that POST is not supported in PLAI. POST requires that the PACS assign the identifier, which in this case is a global UUID from LDAP

	(for credentialholders) or the source PACS (for credentials). Per HTTP standards, PUT supports creation if it does not exist or update if it does exist and allows for specification of a UUID during creation.

Required Types/Fields	Field	Exception?
Only fill out if there are exceptions. If no exceptions, say NO EXCEPTIONS		

1.38.2 PLAI Functional Roles Option

Option Name must have the word "Option" at the end	Functional Roles Option (*PLAI-FunctionalRolesOption*)
Option Version	1.0
Option Description	The Role Management Option allows Authoritative Source (AS) to manage functional roles in addition to identities.
Justify the need for this option	The Baseline Access Interoperability profile unifies identities across physical and logical access control systems. Functional Roles Option takes it forward with AS acting as the single source for functional roles therefore unifying functional roles across physical and logical access control systems

Sample Option Use Case(s)	1. Authoritative Source pushes the identity and functional role to PACS via PLAI Agent. 2. PACS 1 provides the credential number back to PLAI Agent. 3. PLAI Agent pushes the identity, functional role and credentials number to PACS 2. 

Required Operations in addition to what is included from base profile	Operation	Exception?
	/PSIA/AreaControl/Roles/info/<roleUID>	

Required Types/Fields in addition to what is included from base profile Only fill out if there are exceptions. If no exceptions, say NO EXCEPTIONS	Field	Exception?

1.38.3 PLAI Raw Location Option

Option Name must have the word "Option" at the end	Raw Location Option (*PLAI-RawLocationOption*)
Option Version	1.0
Option Description	The Raw Location Option allows the PLAI Agent to receive raw location updates in realtime from various.
Justify the need for this option	The Raw Location Option allows the PLAI Agent to collect access information from multiple PACS to pass on to a connected building management system.
Sample Option Use Case(s)	The Raw Location Option allows the PLAI Agent to collect access information from multiple PACS to pass on to a connected building management system which would be able to perform actions based on occupancy.
Required Operations in addition to what is included from base profile	<table><tr><th>Operation</th><th>Exception?</th></tr><tr><td>None</td><td></td></tr></table> Metadata – AreaControl.Portal/access.granted AreaControl.Portal/access.denied
Required Types/Fields in addition to what is included from base profile Only fill out if there are exceptions. If no exceptions, say NO EXCEPTIONS	None

