PLAI Specification

Security Classification:	Protected
Version:	1.0
Revision:	Rev1
Control:	Uncontrolled when printed
Date	08/07/2019

Document History

Version	Date	Author	Description
1.0 r01	July 8, 2019	Kostas Papadopoulos	1. Initial Document

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 to communicate with PLAI compatible systems in a standard way. Systems implementing the PLAI specification will have access to Credential Holders and Credentials from other compatible access control systems ("PACS"). 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 by Physical Logical Access Interoperability (PLAI) Working Group submitted under the PSIA Intellectual Property Policy for defining information exchange and interoperability between different physical access control systems (PACS) using a subset of PSIA's Area Control profile.

In this document we define the PLAI Base profile and the requirements that an Agent or an Adapter (PACS) has to fulfill in order to be PLAI Compliant. PLAI profile is a subset of AreaControl, Common Security Model (CSEC) and Common Metadata and Event Model (CMEM). For more information please refer to each individual document.

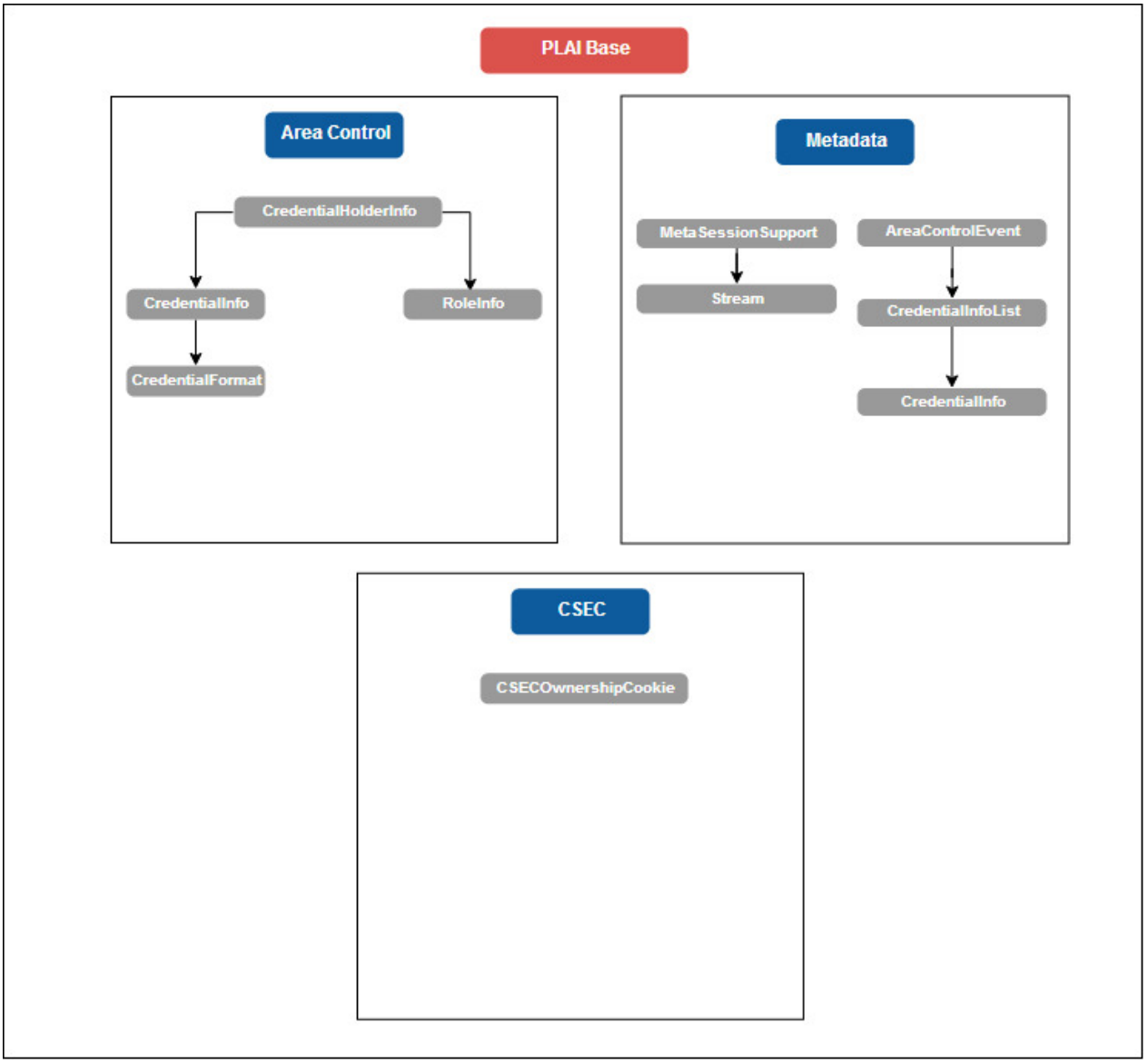


Figure 1: PLAI Base Profile Diagram

Conformance

The PACS 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) and PSIA Common Metadata 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.	
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..

Access Control System:	
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.
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.

Overview of PLAI Implementation

1.2 PLAI Agent Functionality

A PLAI Agent application shall be able to perform the following functionality when it has multiple PACS in the same network:

- Propagate Roles from a central authoritative source to all connected PACS.
- Propagate Credential Holders from a central authoritative source to all connected PACS.
- Propagate Credential Formats from an authoritative source or from each individual PACS to all connected PACS.
- Propagate Credentials either from an authoritative source or from each individual PACS to all connected PACS.

A PLAI Agent can support different authoritative sources for its individual component. With the above scenario in mind one or more PACS can act as an authoritative source. An external source can also be used (ex. LDAP).

1.3 PLAI Adapters Functionality

A PLAI adapter shall be able to perform the following functionality when it is connected to a PLAI Agent:

- Receive Roles from the agent.
- Receive credential holders from the agent.
- Receive credential formats and process them following the rules defined in the credentials format section (1.5.1 and also refer to section 1.21 of the PSIA Area Control Specification).
- Receive credentials from the agent and allow seamless operations between own credentials and those received from the agent.

General Considerations

1.4 Global and Local IDs

Global ID and a Local ID is a required element for all resources.

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.

Resource Requirements

1.5 /PSIA/AreaControl

1.5.1 /PSIA/AreaControl/CredentialFormats

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

1.5.2 /PSIA/AreaControl/Credentials

Command	GET	PUT	POST	DEL
Credentials/info	☑		☑	
Credentials/info/<credentialID>	☑	☑		☑
Credentials/count	☑			

1.5.3 /PSIA/AreaControl/CredentialHolders

Command	GET	PUT	POST	DEL
CredentialHolders/info	☑		☑	
CredentialHolders/info/<credentialHolderID>	☑	☑		☑
CredentialHolders/count	☑			

1.6 /PSIA/Metadata

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

Command	GET	PUT	POST	DEL
Metadata/sessionSupport	☑			
Metadata/stream	☑	☑	☑	☑

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 Access Control

1.7 /PSIA/AreaControl/CredentialFormats

1.7.1 /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>
<…>
```

1.7.2 /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

1.8 /PSIA/AreaControl/Credentials

1.8.1 /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>
  <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.

1.8.2 /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>
 <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.

1.8.3 /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.

1.9 /PSIA/AreaControl/CredentialHolders

1.9.1 /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>.

1.9.2 /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>
```

1.9.3 /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.

1.10 /PSIA/AreaControl/Roles

1.10.1 /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:

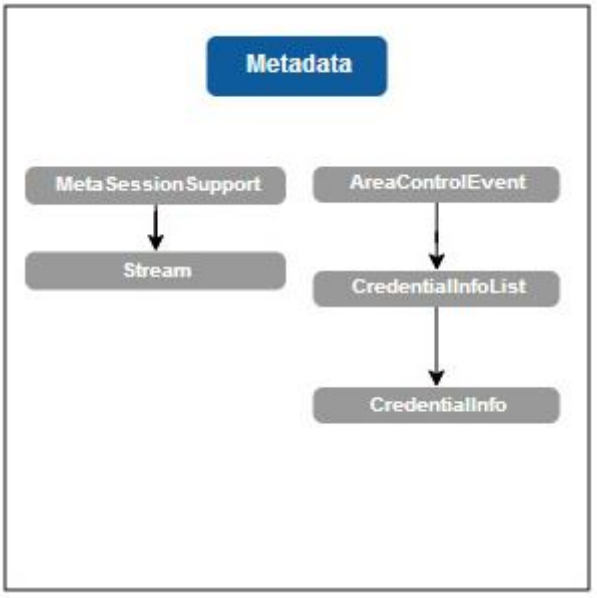
```
<RoleInfo version="1.0" xmlns="urn:psialliance-org">
  <ID>1</ID>
  <UID>{05FE374C-C98D-4D81-B936-D4BF698BCF27}</UID>
  <Name>Role {05FE374C-C98D-4D81-B936-D4BF698BCF27}</Name>  <PermissionIDList>
   <PermissionID>
    <ID>1</ID>
   </PermissionID>
  </PermissionIDList>
</RoleInfo>
```

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

1.11 Services and Resources Overview



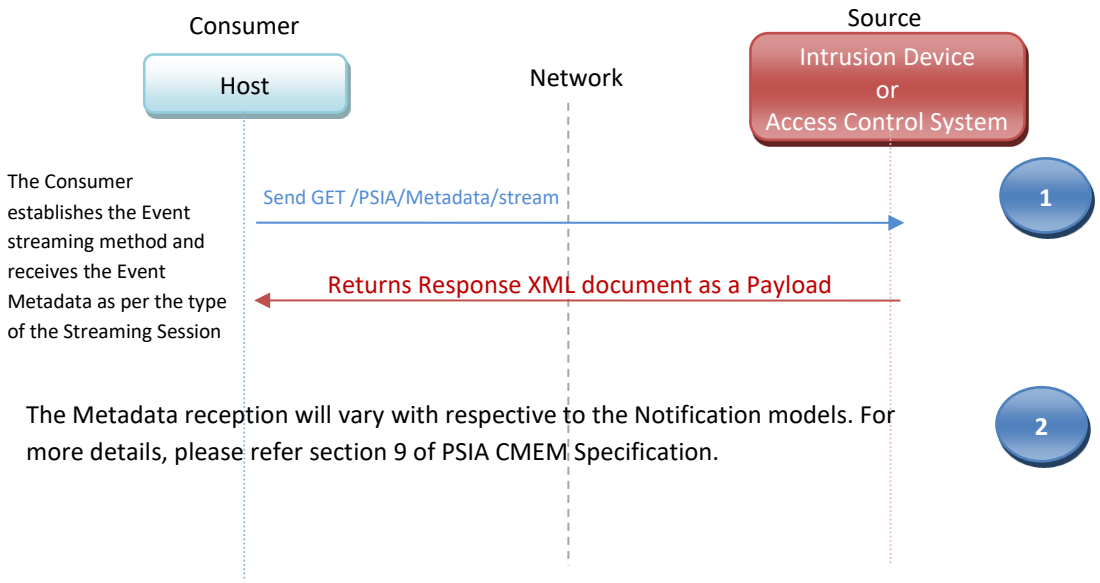
Some, but not all, of the information within the PSIA CMEM specification is repeated here in this document for the sake of completeness.

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

Name	Type	Description
sessionSupport	resource	Metadata resource that defines all of the transport, format and session parameters offered by a device for transferring metadata information.
stream	resource	Metadata resource that acts as the access point for creating metadata/event data streams.

1.11.1 Establishing the Streaming

The following diagram illustrates the sequence of operations involved in establishing event streaming with an adapter or an agent:



1.11.1.1 Step 1: Starting Streaming session

An adapter or an agent 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>

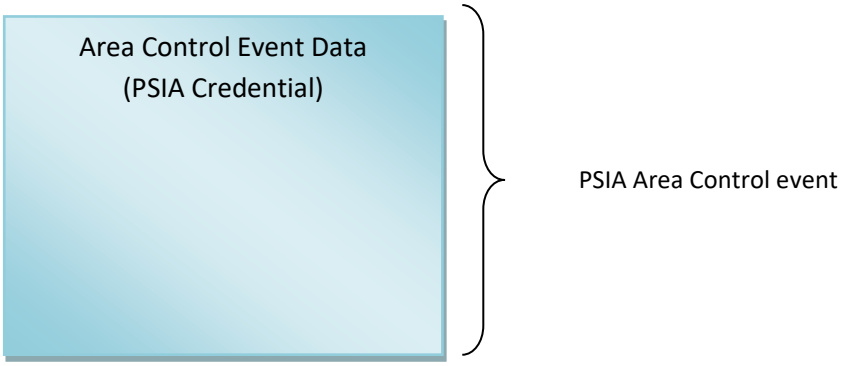
DELETE	None	None	Delete is only used in cases: A) A consumer wants to terminate the current synchronous session after setting parameters for non-synchronous session B) A consumer wants to delete the session resource instance for an asynchronous HTTP/REST session. In this case, the original session ID 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.		

An agent should support credential streaming. An example metasession to enable credential streaming can be found below:

```
<MetaSessionParms version="2.0">
 <MetaXportParms>
  <metaSessionID>0</metaSessionID>
  <metaFormat>xml-psia</metaFormat>
  <metaSessionType>RESTSyncSessionTargetSend</metaSessionType>
  <metaSessionFlowType>dataStream</metaSessionFlowType>
 </MetaXportParms>
</MetaSessionParms>
```

1.11.1.2 Step 2: Sending a credential Event

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

<table>
<tr><td>Metadata ID String (MIDS)</td><td>Domain (format)/Class/Type of the corresponding Metadata/Event (see below) Also known as “metaID”

For Area Control, the metaID is:

<table>
<tr><td>Domain</td><td>psialliance.org</td></tr>
</table>

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 parts of the class/types used by the PLAI Specification. For a full list please consult PSIA Common MetadataEvent Specification.

<table>
<tr><td colspan="3">Domain: psialliance.org</td></tr>
<tr><td>Type in MIDS or GMCH</td><td>Corresponding XML enum type in Status</td><td>Corresponding XML enum values in Status, Details</td></tr>
<tr><td colspan="3">Class: AreaControl.Credential</td></tr>
<tr><td>state.inactive
state.active
state.expired</td><td>StateOfCredential</td><td>Inactive
Active
Expired</td></tr>
<tr><td>assigned.assigned</td><td>ReferenceID</td><td>{ReferenceID of CredentialHolder being assigned. ID=0 means unassigned}</td></tr>
<tr><td colspan="3">Class: AreaControl.CredentialHolder</td></tr>
<tr><td>state.active
state.inactive</td><td>StateOfCredentialHolder</td><td>Active
Inactive</td></tr>
</table>
</td></tr>
<tr><td>Source ID</td><td>UUID/GUID of the metadata/event source (ISO/IEC 9834-8, ITU X.667)</td></tr>
<tr><td>Source’s Local ID (LID)</td><td>Local ID of the Partition, Portal, Zone, Input, or Output</td></tr>
<tr><td>Time</td><td>Absolute time of the generation of the metadata/event info in “xs:dateTime” format</td></tr>
<tr><td>Priority</td><td>Needed for differentiating events in a multi-domain environment</td></tr>
<tr><td>Link</td><td>Multi-event correlation/reference/linkage ID (UUID/GUID)</td></tr>
</table>

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
CredentialInfo	(List of) CredentialInfo (Optional)
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.

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>
        <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

1.12 REST

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

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

1.13 HTTP

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

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

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

1.14 MIME

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

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

1.15 XML

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

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

1.16 Data Encodings

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

1.17 Time Format

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

1.18 SHA-1 Hashing Algorithm

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

1.19 ZLIB Compression

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

1.20 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

Additional information

