

7.4.2 *Kind* — Specifies whether the argument is an ‘in’, ‘out’, or ‘error’ argument for the operation. ‘error’ arguments always function as ‘out’ arguments, but indicate that the operation did not complete successfully. Not all errors possible for a specific operation will be listed, only select typical ones are.

7.4.3 *Form* — Defines the data type of the argument. The terms used to describe data types in this column are defined in the SEMI Compilation of Terms, or are included as part of the specification. Refer to the compilation of terms for the definition of SEMI type name meanings.

8 Background

8.1 In conventional semiconductor equipment communication, such as that defined by SEMI E30, it is assumed that only a single client application can gain access to the equipment for communication. This assumption was valid when the physical transport medium was based on a serial cable connection to the equipment. In such an environment, it is possible to ensure that only a single client application is permitted to interface with the equipment by only providing the serial connection to the machine on which the application runs.

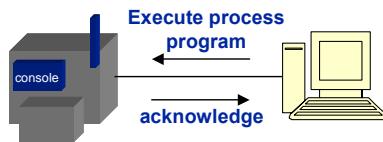


Figure 1
Equipment Communication via Serial Cable

8.2 As equipment communication moves from a serial cable medium to Ethernet, it becomes possible for any client on the network to send requests to the equipment, provided there is a path from the client node to the equipment. Without taking specific measures to secure communication with the equipment, it is not possible to ensure that only authorized clients are permitted to communicate with the equipment.

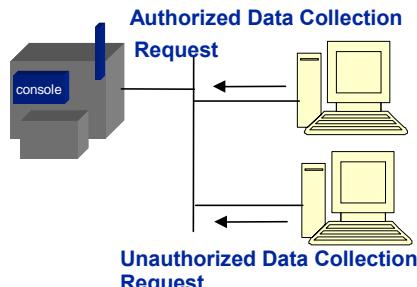


Figure 2
Equipment Communication via Ethernet

8.3 Even if the equipment accepts only a single connection request from the network, it is still vulnerable to a race condition among any clients that attempt to connect. This results in a “first-come, first-served” policy in which the first client to attempt a connection with the equipment is granted access, even if that client is not authorized to communicate with that specific equipment. Such an environment is insufficient for providing adequate protection of equipment resources.

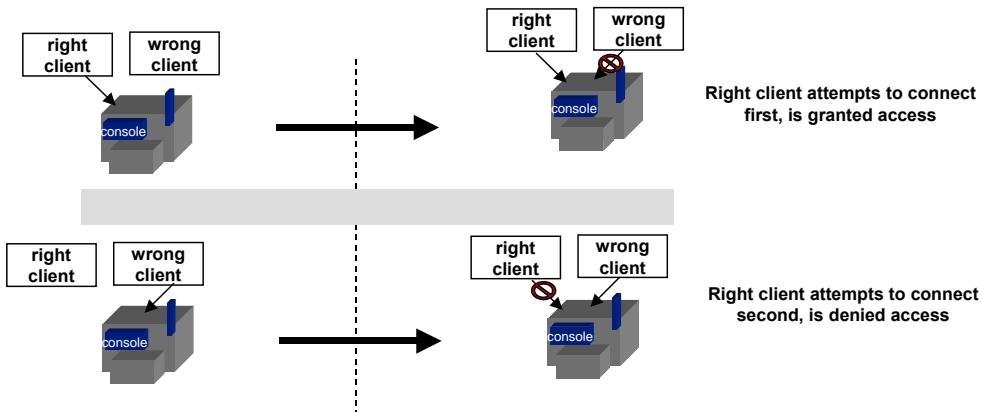


Figure 3
First-come First-served Communication

8.4 A more secure technique to ensure that only authorized clients are allowed to utilize equipment resources is to require clients to first prove their identity to the equipment. If the equipment knows a client's identity, then the equipment may be able to assume that the client is authorized to proceed with further requests.

9 Overview

9.1 This standard defines a mechanism for clients of manufacturing equipment to authenticate to the equipment prior to any subsequent communication. To facilitate this, both the authentication procedure itself and a means for establishing client identities with the equipment beforehand must be defined. This standard also defines a flexible scheme for authorization and mechanisms to support a centralized administration model.

9.2 *Central Administration Model* — A conceptual view of a central security administration model is presented in Figure 4 below.

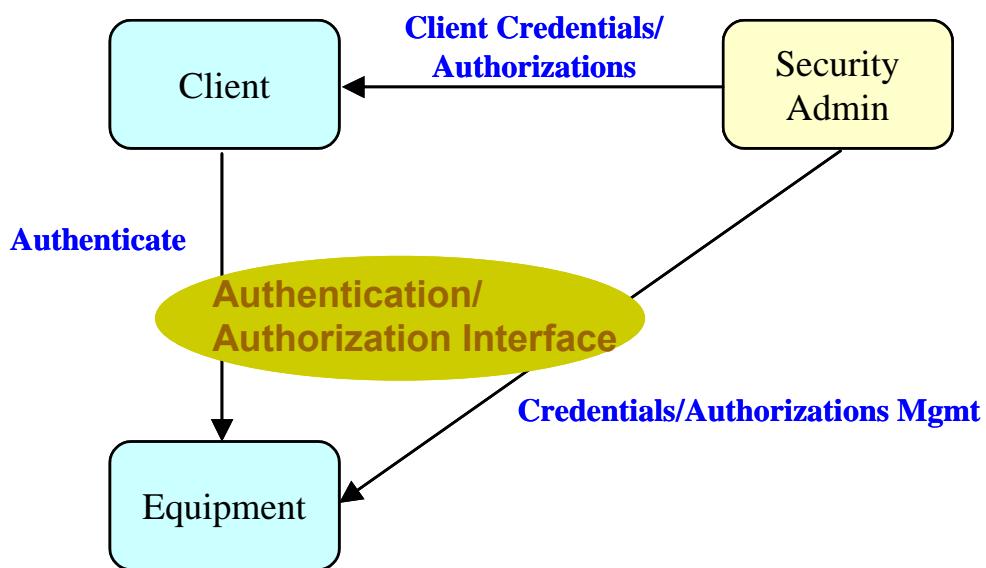


Figure 4
Security Administration Model

9.2.1 *Security Admin* — The security admin issues client credentials, establishes client authorizations and possibly client credentials on the equipment, defines custom roles and their authorizations. Client credentials are identifying information for the client. The security admin manages equipment authorizations and client credentials using an administration interface defined in this specification. The interface between security admin and client to



communicate the credentials will not be defined in this document, and may be as simple as configurations inside the client application.

9.2.2 *Client* — The client holds any issued credentials assigned to it, and authenticates to the equipment to establish an authenticated session.

9.2.3 *Equipment* — The role of equipment in the administration model is to establish the security admin entity for the equipment, hold any defined authorizations, authenticate clients to establish a session, determine client authorization levels and enforce them for the established session.

9.2.4 *Interfaces* — The authentication, administration, and session establishment interfaces will be defined in later sections along with the concepts of establishing client identities and authorizations, though the actual technology and protocols used maybe defined in further detail in a separate adjunct standard.

9.3 *Challenge/Response Authentication*

9.3.1 The authentication process begins when a client first attempts to establish a communication session with the equipment. Upon receiving this request, the equipment challenges the client to prove its identity to the equipment. The client must encrypt some session-generated data using a private or secret key (e.g. a password) that only the claimed client would know. The equipment can verify client's claim using the corresponding public key or shared secret key assigned to the client to decrypt the data, if decrypted data matches plaintext then client is verified.

9.3.2 For communication technologies in which a separate outgoing connection must be established in order for the equipment to send events, the equipment must also verify that the receiving address is valid. This is achieved by sending a ping message to the client endpoint (see Section 15.1.1). This specification currently does not define a mechanism by which a client can transfer an established session to a different endpoint.

9.4 *Authorization*

9.4.1 Any client application wishing to invoke privileged services on the equipment must be authorized to do so. Authorization is implemented by granting access control privileges beforehand to defined clients and later checked against the client's requests. The general operational flow is as follows:

- *Authentication and Credentials Exchange* — Client application authenticates to equipment as to its identity and establishes an authenticated session.
- *Session Established* — Equipment optionally caches client application's privileges and associates these privileges with the newly-created session.
- *Service Request* — Client application makes a service request.
- *Privilege Validation* — Equipment validates the client application's request against its assigned privileges.
- *Session Closed* — Client application closes session, all session state information are reset or cleaned.

9.4.2 Since the client does not authenticate itself for each service request, privilege validation is performed against the authenticated session created for that client. This does not necessarily mean that the implementation of a session requires the use of a stateful or connection-oriented communication protocol, or that authentication must be performed during every protocol-level reconnect. If the implementation protocol is connectionless, the session can be identified by some mutually agreed upon session identifier, negotiated during the authentication process required to establish a session.

NOTE 2: This specification does not define a mechanism for controlling end-user access to equipment services or information. The authentication and authorization scheme defined in this specification applies at the application level only. Management of end-user access to equipment functions and information is strictly the responsibility of applications that make use of the authenticated communication sessions defined in this specification.

10 Authentication

10.1 *Authentication Message Flow* — The authentication message flow during a normal session establishment is presented in Figure 5. Arrows represent the major messages exchanged during the handshake, with side notes clarifying the key data exchanged in each message. Depending on the actual messaging/transport protocols and encryption algorithms used, some additional messages or data parameters may be added, but the concept remains the

same. Specifics of algorithms and protocols will be detailed in a separate technology mapping specification. The terms server and equipment are used interchangeably.

10.1.1 Session Request — This message requests the equipment to initiate establishment of an authenticated session. Upon receiving this message, the equipment begins the authentication handshake as shown in Figure 5. If the request is not accepted, the equipment responds with a protocol specific alert or error message and terminates the connection, see Equipment Challenge.

10.1.2 Equipment Challenge — After the session request is accepted, the equipment responds with a challenge. The equipment selects the algorithms to be used for authentication (if applicable), provides its own identification and challenges the client to authenticate itself. The challenge can be a simple request to client to prove its identity. Note that errors can be returned instead of the normal parameters. Additional errors may be needed as required by the implementation protocol used to support the concepts.

10.1.3 Client Authenticate — The client responds to the server challenge, confirming the algorithms selected (if applicable), equipment's identification and also presents proof of client's own identity. The proof is typically a hash of some random data encrypted with a key that only the client would know and that the server can verify. A secret unique to the authentication protocol may also be generated by the client and sent to equipment in the response. Actual algorithms used for exchange of this secret will be defined by the specific authentication protocol specified in the adjunct standard.

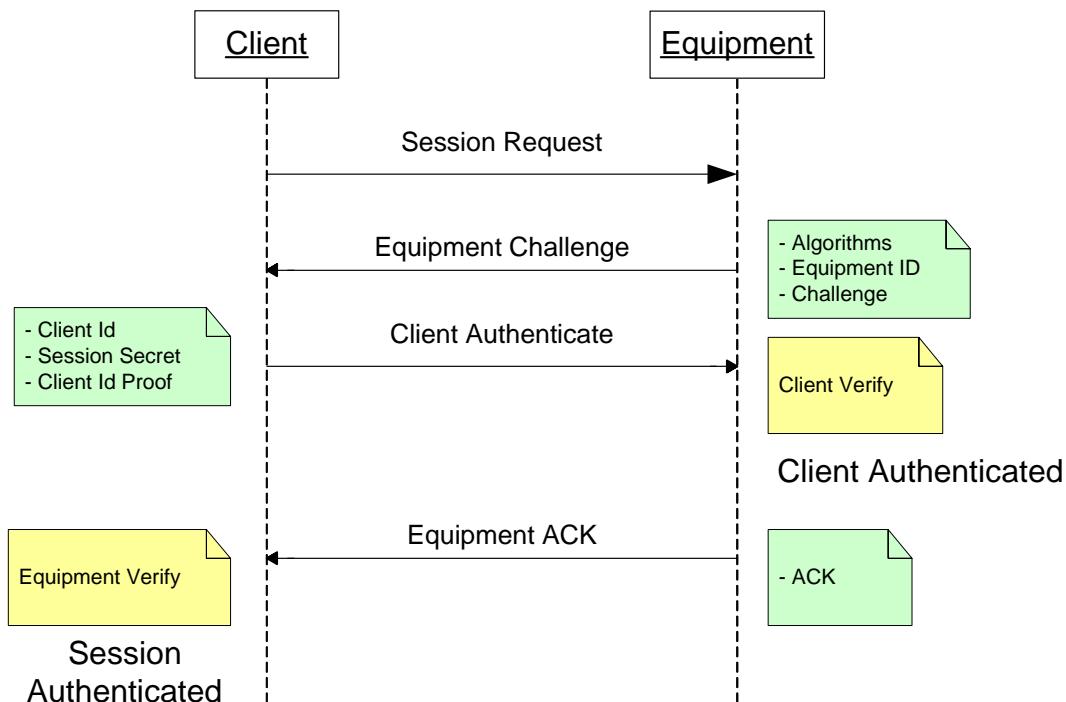


Figure 5
Authentication Message Flow

Table 5 Equipment Challenge Parameter Definitions

Parameter	Description	Kind	Form
Algorithms	Identifies the cryptographic algorithms to be used for key exchange, one-way hash functions and data encryption.	OUT	Text, with delimited fields to identify the algorithms used.
Equipment Id	Equipment identity, exact format as defined by the protocol chosen, and can be as simple as a name identifier such as in a password-based scheme.	OUT	Text.
Challenge	Request for client to authenticate itself, contains random data for	OUT	Text.



	client to sign, preventing replay.		
error	Illegal or unexpected message.	error	Defined by the specific authentication protocol used.

Table 6 Client Authenticate Parameter Definitions

Parameter	Description	Kind	Form
Client Id	Client identity, exact format as defined by the protocol chosen, and can be as simple as a name identifier such as in a password-based scheme.	IN	Text.
Session Secret	Randomly generated data unique to the authentication process. This is encrypted using the equipment's public key or an established shared secret key, depending on the actual protocol used and algorithm chosen for key exchange.	IN	Text, encrypted.
Client Id Proof	Encryption on some random data using a secret that only this client would know, proving client's claim of identity. This serves as a signature. The random data can be the data from equipment's challenge or derived from it.	IN	Text, encrypted.
error	Illegal or unexpected message.	error	Defined by the specific authentication protocol used.

10.1.4 *Equipment ACK* — Equipment verifies client's response and confirms the exchanged session secret in this final authentication protocol message. The acknowledgement contains a parameter that can be a hash of the secret or its derived secret. The derivation algorithm for derived session secret is known both to the equipment and the client, and is part of the protocol specification. This allows client to verify the exchange by verifying the hash. Subsequent communication over the session will make use of the derived secret keys. Note that this standard does not require data be encrypted over the established session, the derived secret is only used to identify the session and hence the authorizations of the authenticated client. Additional errors may be defined as required by the protocol chosen, and technology used by the implementations.

Table 7 Equipment ACK Argument Definitions

Parameter	Description	Kind	Form
ACK	Hash of data derived from the exchanged session secret, this proves equipment's correct decryption of the secret. Hash algorithms as agreed to at the beginning of handshake.	OUT	Text, encrypted.
error	Illegal or unexpected message received.	error	Defined by the specific authentication protocol used.
error	Illegal parameter.	error	Defined by the specific authentication protocol used.
error	Authentication failure.	error	Defined by the specific authentication protocol used.

10.2 Session Identifier

10.2.1 The session id returned from the EstablishSession request (Section 13.2.6) can be used to identify the session for subsequent communications and to enforce authorizations. The identifier will be embedded in special headers in the message transport, and not as formal parameters in the communication messages thus preserving their original structure as defined by the respective interface specifications.

11 Authorization

11.1 Authorization Overview

11.1.1 Any client wishing to invoke privileged services on the equipment must be authorized to do so. Authorization is implemented by granting access control privileges beforehand to defined clients and later checked against the client's requests. The total set of available privileges may include standardized service-level access control, such as privileges to invoke specific services or functions, but may also include supplier-defined finer-grained privileges. A default privilege to allow access to all services available from an authenticated session is defined in this specification, and must be supported by all implementations. A user of the equipment therefore has the option to reduce security enforcement on the equipment to authentication alone, achieved by assigning this all-access privilege to all defined clients.

11.1.2 Figure 6 provides an abstract overview of the relationships among the Role, Principal, and Privilege concepts defined in this specification. A Principal is either assigned to one and only one Role, or to one or more Privileges. A Role is defined as a uniquely-identifiable set of 1 or more Privileges, and cannot be composed from other Roles. Zero or more Principals can be assigned to at most one Role. These relationships are enforced through the management of Access Control List entries, described in Section 11.2.

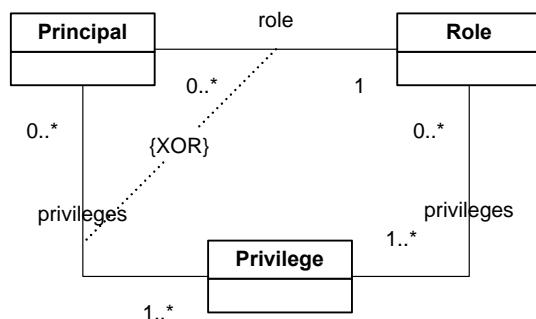


Figure 6
Abstract Model of ACL

11.2 Access Control List

11.2.1 Access control privileges are assigned to defined clients (principals) by using an Access Control List (ACL). An ACL is a collection of ACL entries, each entry granting a subset of the available privileges to a specific subject. A subject can represent a specific principal, a uniquely-named role, or all principals that successfully authenticate with the equipment. A principal can have privileges granted to it through either an explicit list of privileges, or indirectly through assignment to a role. The equipment shall maintain the ACL defined in this specification in non-volatile memory.

11.2.2 Roles are used to simplify the management of privilege assignments to principals. Each unique role has an ACL entry that specifies the privileges assigned to that role. Those privileges can then be granted to principals through a RoleAssignment ACL entry for each principal that should have those privileges. A principal must either have a RoleAssignment or an explicit list of privileges assigned to it (PrivilegeAssignment); it is not permitted to have both.

11.2.3 The equipment shall refuse session requests from clients that have no ACLEntry, either explicitly through a role or privilege assignment, or implicitly through the existence of an ACLEntry for all authenticated principals.

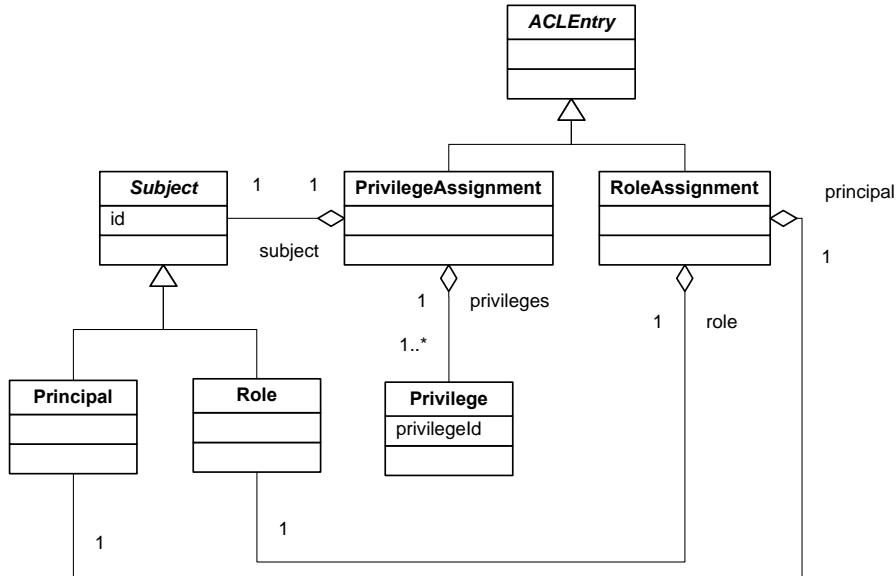


Figure 7
ACL Entry Description

11.2.4 *ACLEntry*

11.2.4.1 Figure 7 shows the classes used for requesting the creation of a new *ACLEntry* on the equipment (see Section 12.3.2.3). The *ACLEntry* is an abstract base class for the two kinds of *ACLEntry*, *PrivilegeAssignment*, and *RoleAssignment*. There can be only one *ACLEntry* for each subject. Attempts to add a second entry for same subject shall generate a duplicate entry error (see Section 12.3.2.3). This class has no attributes or associations.

11.2.5 *PrivilegeAssignment*

11.2.5.1 This class defines an association between a given subject (either a *Principal* or a *Role*) and its set of assigned privileges on the equipment. A *PrivilegeAssignment* for a *Principal* establishes all privileges that are specifically granted to the *Principal*. A given *Principal* must either have a *PrivilegeAssignment* *ACLEntry* or a *RoleAssignment* *ACLEntry* (see Section 11.2.6); it cannot have one (or more) of each. A *PrivilegeAssignment* for a *Role* defines the role itself and the privileges that will be granted to any *Principal* that is assigned to that *Role* (see Section 11.2.6). There can be only one *PrivilegeAssignment* *ACLEntry* for a given *Role*.

Table 8 PrivilegeAssignment Association Definitions

Association Role Name	Definition	Comments
subject	The <i>Principal</i> or <i>Role</i> that is being granted privileges.	One and only one element of any type derived from <i>Subject</i> , described in Section 11.2.7.
privileges	The set of privileges being granted.	List of one or more elements of type <i>Privilege</i> , described in Section 11.2.10.

11.2.6 *RoleAssignment*

11.2.6.1 This class assigns a specific *Principal* to a *Role*. The identified *Role* must have a *PrivilegeAssignment* *ACLEntry* elsewhere in the *ACL*. All privileges assigned to the identified *Role* are granted to the identified *Principal*'s sessions. A given *Principal* must either have a *RoleAssignment* *ACLEntry* or a *PrivilegeAssignment* *ACLEntry* (see Section 11.2.5); it cannot have one (or more) of each.

11.2.6.2 A *PrivilegeAssignment* *ACLEntry* for a *Role* cannot be deleted from the *ACL* until all *Principals* that were assigned to that *Role* have been removed from it by deletion of their *RoleAssignment* *ACLEntry*. Any authenticated sessions that were established for such *Principals* prior to the deletion of their *RoleAssignment* *ACLEntry* shall continue to be operable with that *Role*'s privileges until the session is terminated. Subsequent attempts by such



Principals to request sessions shall be rejected unless there is a default set of privileges assigned to the reserved “anyPrincipal” (see Section 11.2.8). Otherwise, in order for session requests to be accepted, these Principals will have to have specific privileges assigned to them directly through a PrivilegeAssignment ACLEntry, or indirectly through assignment to a new Role.

Table 9 RoleAssignment Association Definitions

Association Role Name	Definition	Comments
principal	The principal that is being assigned to the Role.	One and only one element of type Principal, described in Section 11.2.8.
role	The Role to which the principal is being assigned.	One and only one element of type Role, described in Section 11.2.9. The Subject id for the identified Role must be equal to the Subject id of a PrivilegeAssignment for a Role appearing elsewhere in the ACL.

11.2.7 *Subject*

11.2.7.1 This is an abstract base class used to represent any entity that can be granted privileges in an ACL entry. This can be either a Principal or a Role. The id is a text field that identifies a particular Subject. If the Subject is a Principal, then the id is equal to that Principal’s unique id used during authentication. If the Subject is a Role, then the id is equal to the name of the Role.

Table 10 Subject Attribute Definitions

Attribute Name	Definition	Form
id	Identifies a Principal or Role.	Text.

11.2.8 *Principal*

11.2.8.1 This class represents a specific Principal that can be authenticated and assigned privileges. This specification defines a reserved Principal id that represents any Principal that successfully authenticates to the equipment. The value of the id for this reserved Principal is “urn:semi-org:auth:anyPrincipal”. All implementations shall support this reserved Principal. If an equipment user’s security policy permits the approach, a PrivilegeAssignment or RoleAssignment ACL entry can be created for the reserved Principal to assign a default set of privileges to any client that successfully authenticates, rather than creating an ACL entry for each Principal.

11.2.8.2 Each client that attempts to authenticate to the equipment must have an id that matches the id of one PrivilegeAssignment or RoleAssignment ACLEntry for that Principal, or else there must be one PrivilegeAssignment or RoleAssignment ACLEntry for the reserved “anyPrincipal”. If there are no such entries for the authenticating client, the equipment shall reject the session request. If there is a PrivilegeAssignment or RoleAssignment ACLEntry for both the specific Principal and the reserved “anyPrincipal”, the equipment shall accept the session request, and shall grant only those privileges assigned to the specific Principal to the session. Privileges assigned to the reserved “anyPrincipal” shall not be granted to such sessions.

11.2.9 *Role*

11.2.9.1 This class represents a Role that can be assigned privileges. Roles can simplify the assignment of privileges to Principals. This is achieved by creating a single PrivilegeAssignment ACL entry for the Role containing the appropriate privileges, and then assigning each Principal (or the reserved “anyPrincipal”) to that Role using a RoleAssignment ACL entry.

11.2.10 *Privilege*

11.2.10.1 This class represents a specific privilege that can be granted to subjects as part of an ACL entry. All defined privileges supported by the equipment can be obtained using the GetDefinedPrivileges operation of the SecurityAdmin interface. How each privilege is applied to a subject is determined by the specification or equipment supplier that defines the privilege. The privilege id is a text field that identifies a particular privilege and must be



unique across all privileges defined on the equipment. Permission and privilege is used interchangeably in this specification.

11.2.10.2 This specification defines a reserved Privilege id that represents all Privileges that are supported by the equipment. The value of the id for this reserved Privilege is “urn:semi-org:auth:allPrivileges”. All implementations shall support this reserved Privilege. If an equipment user’s security policy permits the approach, an ACL entry can be created for any Principal or Role that assigns this reserved Privilege to that Principal or Role. Principals or Roles that are assigned this privilege are granted all privileges supported by the equipment. A PrivilegeAssignment ACL entry containing this privilege cannot include any other privileges supported by the equipment in the entry.

Table 11 Privilege Attribute Definitions

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
privilegeId	Identifies a specific privilege.	Text, must be equal to either the reserved value “urn:semi-org:auth:allPrivileges”, or to the ‘id’ attribute of one of the DefinedPrivileges supported by the equipment (see Section Table 21).

11.2.10.3 This specification also defines a reserved Privilege id that represents the Privileges of the SecurityAdmin as described in 12.2. The value of the id for this reserved Privilege is “urn:semi-org.auth:securityAdminPrivileges”. All implementations shall support this reserved Privilege. One and only one Principal can be assigned this privilege, either through a RoleAssignment or PrivilegeAssignment ACLEntry. Once assigned, that Principal is permitted access to all operations defined for the SecurityAdmin interface. Requests to create more than one ACLEntry assigning this privilege to a Principal shall be rejected (see Section 12.3.2.3).

12 Administration

12.1 Administration Overview

12.1.1 As the number of clients and resources to manage across a factory scales up, support for a centralized security administration model becomes a critical capability. This specification will not describe how a factory can implement such a model for their network, but will specify standard mechanisms to enable this capability. This specification describes a security administration interface to allow remote management of client credentials and authorizations, such as from a central location. Only concepts and behavior are described, it is expected that a separate technology adjunct standard will be created describing the specific message structure, protocols and algorithms that will be used.

12.2 Security Admin

12.2.1 The equipment must support one security administration account. This admin agent will have privileges to utilize the SecurityAdmin interface defined in this specification. The initial security admin credentials must be established at the time of equipment install via some out of band means, this can be a local operator interface or provided by equipment vendor as a factory preset. If the admin credentials are factory preset, then there must be some means for end-user to re-establish its credentials. This specification requires that only the security admin can utilize the services defined in the security administration interface. Security admin privileges are granted to one and only one Principal via a reserved privilege id. The equipment shall only support 1 active security admin session at a time.



12.3 Security Administration Interface

12.3.1 Figure 8 summarizes the standard services available through the security admin interface to any client authenticated as the security admin. Equipment manufacturers are free to define additional operations if needed.

«interface» SecurityAdmin
<i>GetDefinedPrivileges()</i> <i>GetACL()</i> <i>AddACLEntry()</i> <i>DeleteACLEntry()</i> <i>GetActiveSessions()</i> <i>SetMaxSessions()</i> <i>GetMaxSessions()</i>

Figure 8
Security Administration Interface

12.3.2 Security Administration Operations

Table 12 Security Administration Operations

<i>Operation</i>	<i>Description</i>	<i>Type</i>	<i>Requestor/ Sender</i>	<i>Responder/ Receiver</i>
<i>GetDefinedPrivileges</i>	Requests for the list of all privileges defined on equipment.	RR	Admin	Equipment
<i>GetACL</i>	Requests for the collection of ACL entries defined on the equipment.	RR	Admin	Equipment
<i>AddACLEntry</i>	Adds an ACL entry on the equipment.	RR	Admin	Equipment
<i>DeleteACLEntry</i>	Deletes an ACL entry on the equipment.	RR	Admin	Equipment
<i>GetActiveSessions</i>	Requests for information on all active sessions.	RR	Admin	Equipment
<i>SetMaxSessions</i>	Sets maximum limit on number of non-admin sessions allowed.	RR	Admin	Equipment
<i>GetMaxSessions</i>	Requests for the maximum limit of non-admin sessions currently set on equipment.	RR	Admin	Equipment

12.3.2.1 *GetDefinedPrivileges* — This operation retrieves the set of privileges supported by the equipment manufacturer. The list shall include the reserved “allPrivileges” Privilege defined in this specification (see Section 11.2.10).

Table 13 GetDefinedPrivileges Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
<i>privileges</i>	Contains description and identifier of all defined privileges.	OUT	Unordered list of type DefinedPrivilege, Section 12.3.3.3.
<i>error</i>	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.
<i>error</i>	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.

12.3.2.2 *GetACL* — This operation requests for the entire ACL currently defined on the equipment.

Table 14 GetACL Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
ACL	Returned ACL of equipment. Null list is returned if no ACL entries are defined.	OUT	Unordered list of types derived from ACLEntry, Section 11.2.4.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.
error	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

12.3.2.3 *AddACLEntry* — This operation adds a new ACL entry to the equipment’s ACL. If the ACLEntry is a PrivilegeAssignment or a RoleAssignment for a Principal or Role that already has a PrivilegeAssignment or RoleAssignment ACLEntry, the equipment shall return an error. If the ACLEntry is a PrivilegeAssignment and refers to privilege id’s that are not supported by the equipment, the equipment shall return an error. If the ACLEntry is a RoleAssignment and refers to a Role that does not have a PrivilegeAssignment ACLEntry defined, the equipment shall return an error. If the ACLEntry is a RoleAssignment or PrivilegeAssignment that would result in more than one Principal having the SecurityAdmin privilege (see Section 11.2.10.3), the equipment shall return an error. Otherwise, the equipment shall store the newly-added ACLEntry together with the rest of the ACL in non-volatile memory.

Table 15 AddACLEntry Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
aclEntry	New ACL entry to be added.	IN	Structured data of type ACLEntry, Section 11.2.4.
error	Duplicate entry found.	error	Text equal to duplicate principal/role name error.
error	Unrecognized role.	error	Text equal to unrecognized role error.
error	Unrecognized privilege.	error	Structured data, of type UnrecognizedPrivilegeError, Section 12.3.3.3.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.
error	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

12.3.2.4 *DeleteACLEntry* — This service removes an ACL entry from equipment’s ACL. The deletion of an ACLEntry that has been applied to one or more active sessions does not affect the privileges granted to that session, nor does it terminate any such session. Attempts to establish new sessions by a Principal whose PrivilegeAssignment or RoleAssignment ACLEntry has been deleted shall be rejected by the equipment until a new ACL entry is created for that Principal, or unless there is an ACLEntry for the reserved “anyPrincipal” (see Section 11.2.8). It is an error to delete a PrivilegeAssignment ACLEntry for a Role when there are still Principals assigned to that Role through a RoleAssignment ACLEntry.

Table 16 DeleteACLEntry Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
subjectId	Identifies the ACL entry to be deleted by the id of subject.	IN	Text
error	Entry not found.	error	Text, equal to ACL entry not found error.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the



			bad session id.
error	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

12.3.2.5 *GetActiveSessions* — This operation allows the security admin to retrieve information for all active non-admin sessions currently established on the equipment.

Table 17 GetActiveSessions Parameter Definitions

Parameter	Description	Kind	Form
activeSessions	List of active non-admin sessions on the equipment. An empty list is returned if there are no active sessions.	OUT	Unordered list, of type ActiveSession, Section 12.3.3.4.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.
error	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

12.3.2.6 *SetMaxSessions* — This operation allows the security admin to set the maximum number of simultaneous non-admin sessions that the equipment will allow. This limit only applies to new session requests not coming from the security admin, current established sessions are not affected. If the number of current active sessions exceeds this limit, any new non-admin session requests shall be rejected by the equipment. New session requests are authenticated first to determine if the client is not the security admin, then the check on the limit is made to determine whether or not to reject the request. Setting this limit to 0 will restrict to security admin sessions only.

Table 18 SetMaxSessions Parameter Definitions

Parameter	Description	Kind	Form
maxSessions	Limit on number of simultaneous non-admin sessions equipment will allow.	IN	Integer.
sessionCount	Total count of all current authenticated non-admin sessions on equipment.	OUT	Integer.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.
error	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

12.3.2.7 *GetMaxSessions* — This operation retrieves the current maximum limit of simultaneous non-admin sessions set on equipment.

Table 19 GetMaxSession Parameter Definitions

Parameter	Description	Kind	Form
maxSessions	Limit on number of simultaneous non-admin sessions equipment will allow.	OUT	Integer
sessionCount	Total count of all current authenticated non-admin sessions on equipment.	OUT	Integer
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.

Parameter	Description	Kind	Form
error	Operation not authorized.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

12.3.3 Type Definitions

12.3.3.1 *UnauthorizedOperationError* — This class describes an authorization error associated with a specific operation.

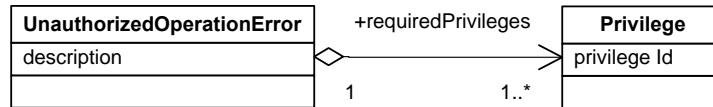


Figure 9
UnauthorizedOperationError Description

Table 20 UnauthorizedOperationError Attribute Definitions

Attribute Name	Definition	Form
description	Description of error.	Text, including name of operation and any important context information associated with original request.

Table 21 UnauthorizedOperationError Association Definitions

Attribute Name	Definition	Form
requiredPrivileges	List of one or more privileges that grant authorization for the requested operation.	List of one or more elements of type Privilege , Section 11.2.10.

12.3.3.2 *UnrecognizedPrivilegeError* — This class describes an error where one or more privileges are not recognized by the equipment.



Figure 10
UnrecognizedPrivilegeError Description

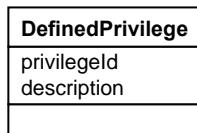
Table 22 UnrecognizedPrivilegeError Attribute Definitions

Attribute Name	Definition	Form
description	Description of error.	Text, including name of operation and any important context information associated with original request.

Table 23 UnrecognizedPrivilegeError Association Definitions

Attribute Name	Definition	Form
requiredPrivileges	List of one or more privileges not recognized by the equipment.	List of one or more elements of type Privilege, Section 11.2.10.

12.3.3.3 *DefinedPrivilege* — This class is used to describe privileges supported by the equipment. The equipment shall maintain the list of supported privileges in non-volatile memory.

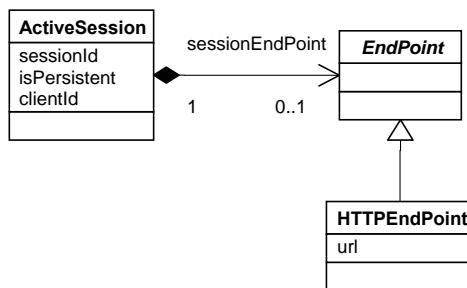


**Figure 11
DefinedPrivilege Description**

Table 24 DefinedPrivileges Attribute Definitions

Attribute Name	Definition	Form
privilegeId	Identifier for the privilege, same as id of privilege in ACL definition. Referred to by the Security Admin in defining ACL entries.	Text, unique across all Privileges.
description	A human-readable description of the privilege clearly describing the consequences of granting this privilege to a Principal.	Text.

12.3.3.4 *ActiveSession* — This class describes an active session on the equipment. Not all communication protocols require the specification of a separate endpoint. Any implementation of this specification must define requirements for the presence or absence of the endpoint information provided for sessions (see Section 13.2.6).



**Figure 12
ActiveSession Description**

Table 25 ActiveSession Attribute Definitions

Attribute Name	Definition	Form
sessionId	Unique session identifier established between the client and the equipment.	Text.
isPersistent	Whether session is persistent or not, see Section 13.2.3 regarding making sessions persistent.	Boolean.
clientId	Client identifier as authenticated by the equipment.	Text.

Table 26 ActiveSession Association Definitions

Association Role Name	Definition	Comments
sessionEndPoint	Addressing information for sending notifications to the authenticated Principal.	Structured data, of any type derived from EndPoint, defined in Section 12.3.3.4.1.

12.3.3.4.1 *EndPoint* — An abstract data type used to describe a communication end point. This specification only defines end points using the HTTP protocol, although others are possible depending on the communication protocol. Any implementation of this specification must define requirements for the support of separate endpoint addresses, and any additional EndPoint subclasses required for the communication protocol used. This class has no attributes or associations.

12.3.3.4.2 *HTTPEndPoint* — This class is used to describe addressing information that the equipment is to use for sending notifications to clients where the communication protocol in use is based on HTTP.

Table 27 HTTPEndPoint Attribute Definitions

Attribute Name	Definition	Form
url	HTTP URL of communication end point.	Text, formatted according to the IETF RFC 2396 specification for URL syntax.

12.4 Equipment Console Interfaces

12.4.1 The equipment manufacturer shall provide access to the SecurityAdmin operations from the equipment console.

13 Session Communications

13.1 *Session Message Flow* — Figure 13 shows an example of a typical message flow for an authenticated communication session.

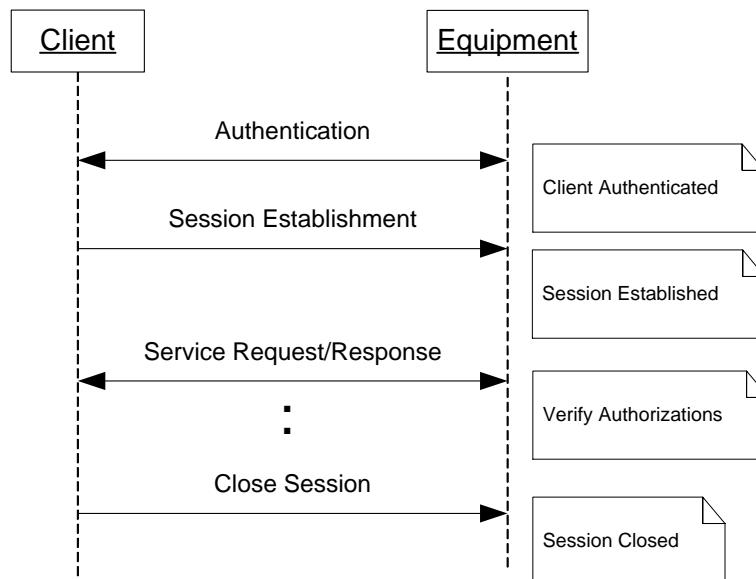


Figure 13
Session Message Flow

13.1.1 *Authentication* — Authentication message flow as described in Section 10 in which both client and equipment mutually verify each other's identity.

13.1.2 Session Establishment — Session establishment in this specification is achieved by sending a EstablishSession request following authentication. EstablishSession includes the client's end point (if necessary) for equipment notifications (see Section 13.2.6). Upon receiving the request, the equipment determines if the authenticated client has a corresponding ACLEntry. If so, a new session is created for the client if possible (see Section 13.2.6), and generates a unique session identifier for it. The session identifier is returned to the client in the EstablishSession response for use in subsequent requests. Every session established on the equipment has associated with it a unique session identifier, the client's id, and end point address. Once established, the session is granted the Privileges assigned to the client based on the corresponding ACLEntry for that client (Principal). Changes to the ACL do not affect the Privileges of sessions established prior to the change..

13.1.3 Service Request and Response — Service requests, as defined by any interface specification that requires authenticated communications and authorization, are made. The unique session identifier required for authorization is carried in reserved session headers within the implementation protocol (see Section 10.2).

13.1.4 Close Session — Request by a client to terminate an authenticated session. Upon receiving this request, the equipment clears all context information related to that session, including closing any existing, active connection, and terminating any activities initiated by the client during the session.

13.2 SessionManager Interface

13.2.1 The equipment shall implement the session management operations defined in this section. These operations are accessible to any client after an authenticated session is established, including the security admin.

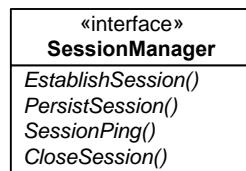


Figure 14
Generic Session Services

13.2.2 SessionManager Operations

Table 28 SessionManager Operations

<i>Operation</i>	<i>Description</i>	<i>Type</i>	<i>Requestor/ Sender</i>	<i>Responder/ Receiver</i>
PersistSession	Requests equipment to persist current session.	RR	Client	Equipment
SessionPing	Checks if remote entity is still active.	RR	Client	Equipment
CloseSession	Requests to close and terminate current or specified session.	RR	Client	Equipment
EstablishSession	Request to establish a new authenticated session and to set client end point.	RR	Client	Equipment

13.2.3 PersistSession — Upon receiving this request, if the client is requesting that the session be persisted, the equipment shall store the client's current session context including the session identifier, any client endpoint addressing information, and session privileges to non-volatile memory. Once the session has been persisted, it remains active across equipment shutdowns, reactivating as soon as the equipment is in a state where it can support external communications. Persistent sessions can still be terminated if the equipment determines that the client is unreachable. For further details, see the Session Monitoring state of Session State Model, defined in Section 14.3.2.7. See session state transitions (Table 30) into and out of the Frozen state for required behavior on freezing and resuming persistent sessions. If the client uses this operation to remove persistence from the session state, the equipment can remove the client's session context from non-volatile memory, and shall not re-activate this session automatically at equipment startup.



Table 29 PersistSession Parameter Definitions

Parameter	Description	Kind	Form
persist	Whether or not the session should be persisted	IN	Boolean. If the value of this parameter would not change the session state (e.g., persist='true' when the current session is already persistent), the equipment shall silently ignore the request.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.

13.2.4 *SessionPing* — Upon receiving this request, the equipment shall reply with its id. Ids are preconfigured and must be the same one as used during the authentication phase. If equipment ids are not used during authentication, then this is an identifier that shall be configurable at the equipment. See the Session Monitoring state defined in Section 14.3.2.7, regarding behavior of established authenticated sessions in the event of ping timeouts.

Table 30 SessionPing Parameter Definitions

Parameter	Description	Kind	Form
equipmentId	Id of the equipment responding to the ping message.	OUT	Text, equal to the equipment Id used in authentication phase.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.

13.2.5 *CloseSession* — Upon receiving this request, the equipment shall send the SessionClosed notification (see Section 15.1.3) to the client whose session is being closed, delete all information related to the client's session, and close any active connection to the client if one exists.

13.2.5.1 The established equipment security admin has the privilege to terminate or close any active session via the CloseSession request. This admin function can be used to cleanup sessions where clients are no longer active or to force termination of any session even when a client is still active.

Table 31 CloseSession Parameter Definitions

Parameter	Description	Kind	Form
sessionId	Identifier of the session to terminate. Only the security admin has the privilege to terminate sessions other than its own.	IN	Text, equal to the id of the session to be terminated.
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.
error	Operation not authorized. Occurs only when the client has specified a session id other than its own, and the client is not the security admin.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

13.2.6 *EstablishSession* — This operation creates a new authenticated session, sets the end point address to be used for sending notifications to the client and establishes a session identifier. This must be the first request sent after authentication is complete in order to create a new session, any other request is rejected with operation cannot be performed error. The session identifier must be unique for each active session and is valid as long as the session is not closed. The session identifier must be included in all subsequent requests in a special header defined by the protocol specification. A client must have an ACLEntry defined in order to establish sessions, else request is

rejected. The client's assigned Privileges from its ACLEntry are read at the time the session is established and are assigned to the session. Subsequent requests using the established session will use the session's Privileges for access verification even if the client's Privileges have changed since then.

Table 32 EstablishSession Parameter Definitions

Parameter	Description	Kind	Form
endpoint	End point of client where the equipment will send all notifications.	IN	Structured data, of any type derived from EndPoint, Section 12.3.3.4.1.
sessionId	Unique session identifier generated by equipment after authentication, this identifier must be used in all subsequent requests.	OUT	Text.
error	Max session limit exceeded	error	Text, equal to session limit exceeded error.
error	Operation not authorized. For example client might not have any ACLEntry defined.	error	Structured data, of type UnauthorizedOperationError, Section 12.3.3.1.

13.3 Encryption of Data

13.3.1 This standard does not require data transmitted over an established session to be encrypted, encryption is only required as specified by the authentication protocol.

14 State Model

14.1 This section defines the client's view of the equipment's behavior in supporting authentication, session establishment, session management and communications over an established session. The state models represent client's view of the equipment's state, not necessarily the actual equipment's internal operations. No transitions in these state models are to be made available for event reporting.

14.2 *Session Establishment State Model* — The session establishment state model defines the behavior of the equipment for establishment of authenticated sessions over this interface. It defines the equipment's management of authenticated sessions overall during startup and shutdown. For definition of individual session behaviors, see Section 14.3.

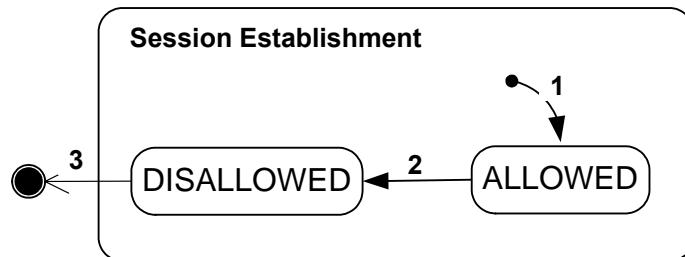


Figure 15
Session Establishment State Diagram

14.2.1 Session Establishment States

14.2.1.1 *DISALLOWED* — This state is active when the equipment is shutting down, either as part of an orderly shutdown or an emergency power-off. While in this state, any requests for new authenticated sessions (if it is possible to receive them) are rejected and all active sessions have been closed or FROZEN (Section 14.3.2.9). In the case of an emergency shutdown, all active sessions may be dropped abruptly and notifications to clients may not be possible. In this case, persistent sessions shall still be resumed on startup since their state is saved to non-volatile storage.

14.2.1.2 *ALLOWED* — Establishment of authenticated sessions is allowed. Upon entry to this state, all FROZEN persistent sessions are resumed per the session state model. While in this state, requests for establishment of new authenticated sessions are accepted, see Section 14.3 Session State Model for behavior on establishment of an authenticated session.

14.2.2 Session Establishment State Transitions

Table 33 Session Establishment State Transitions Table

#	Current State	Trigger	New State	Actions	Comments
1	(no state)	Completion of system initializations or the equipment is in a state where it can support authenticated session communications.	ALLOWED	The equipment shall wait for requests to establish authenticated sessions. All FROZEN persistent sessions are resumed by automatic re-entry to the ESTABLISHED state.	Any connectivity issues encountered in attempting to re-establish persistent sessions shall be managed by each session's Authenticated Session state model defined in Section 14.3 (see, for example, Section 14.3.2.7).
2	ALLOWED	The equipment is powering down or emergency down.	DISALLOWED	If possible, the equipment shall close all active non-persistent sessions, notify all clients with persistent sessions that their session is being FROZEN, and reject any requests for session establishment.	
3	DISALLOWED	Equipment shutdown complete.	(no state)		

14.3 Session State Model

14.3.1 The session state model defines behavior of equipment related to establishing an authenticated session, supporting the services and message exchanges over the established session and to terminating the session.

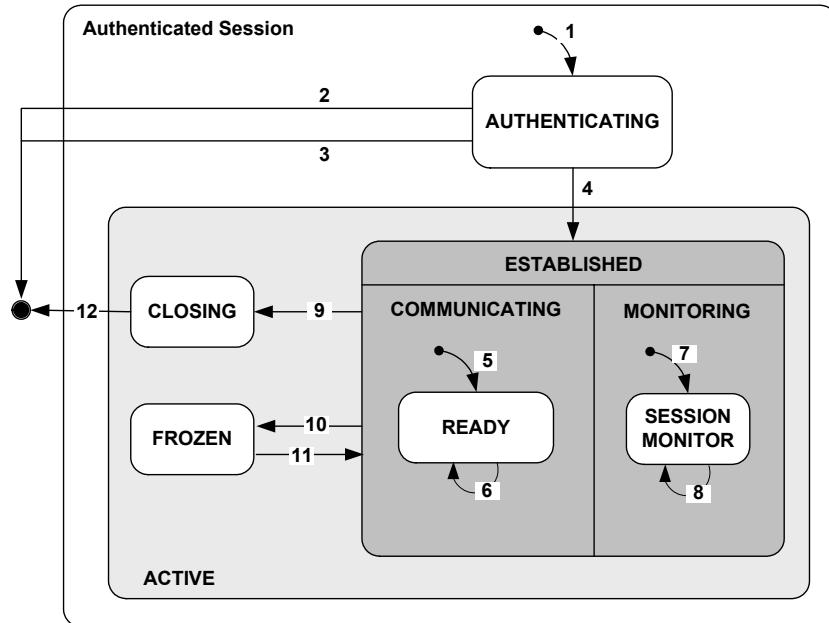


Figure 16
Authenticated Session State Diagram



14.3.2 Session States

14.3.2.1 *ACTIVE* — Parent state, covers entire lifecycle of an authenticated session. Session is active beginning with a successful request to establish a new session, and is active until the session is closed.

14.3.2.2 *AUTHENTICATING* — This state is where the client and the equipment carry out the authentication process and establish new authenticated sessions. Privileges are granted to the session once the client is authenticated and the session successfully established. See Section 10 for the authentication message flow, and Section 13.2.6 for session establishment.

14.3.2.3 *ESTABLISHED* — This state is the parent of all those sub-states that relate to service invocation and communications subsequent to successful authentication and session establishment. It has two simultaneous sub-states, *COMMUNICATING* and *MONITORING*, both of which are active whenever the session is *ESTABLISHED*.

14.3.2.4 *COMMUNICATING (ESTABLISHED Sub-State)* — This is the *ESTABLISHED* sub-state where all session communications are handled.

14.3.2.5 *MONITORING (ESTABLISHED Sub-State)* — This state provides a watchdog mechanism to monitor status of the established session.

14.3.2.6 *READY (COMMUNICATING Sub-State)* — Initial state upon session established and *COMMUNICATING* state becomes active. Both the client and the equipment have been successfully authenticated without errors. The equipment is awaiting service requests in this state and can report relevant events to authenticated clients. Except for the services defined in this specification, the services allowed or events reported are dependent on the specifications that defined them.

14.3.2.7 *SESSION MONITOR (MONITORING Sub-State)* — Initial state upon entry to *ESTABLISHED* state and when *MONITORING* becomes active. The equipment shall periodically check the active status of the session by sending ping messages to the client. See Section 15.1.1 for details of the *SessionPing* message. A ping message is sent upon first entry to the state. When resuming persistent sessions during equipment startup, the ping message tells client of each persistent session that the equipment is now online. The interval at which the next ping message is sent, and the number of attempts to be made before closing a session shall be configurable, and must include a way to disable periodic pings (such as by using a delay interval of zero). If the client does not respond after the configured number of pings, the equipment shall terminate the session automatically. The default number of ping attempts shall be set to 3.

14.3.2.8 *CLOSING* — Session is being terminated either at request of client, security admin, or equipment initiated due to errors. Session context are cleared including established session identifier. A *SessionClosed* (section 15.1.3) notification is sent to client (if possible) on transition to this state.

14.3.2.9 *FROZEN* — A persistent session is put on hold as the equipment is powered down or in a state where it is not able to communicate. A *SessionFrozen* (section 15.1.2) notification is sent to client (if possible) on transition to this state. When resuming a persistent session, behavior is governed by re-entry to the *ESTABLISHED* state and its sub-states.

14.3.3 Session State Transitions

Table 34 Session State Transitions Table

#	Current State	Trigger	New State	Actions	Comments
1	(no state)	Client request to start an authenticated session	AUTHENTICATING	Equipment and client begins authentication and session establishment message flow (see Sections 10 and 14.3.2.2).	

#	<i>Current State</i>	<i>Trigger</i>	<i>New State</i>	<i>Actions</i>	<i>Comments</i>
2	AUTHENTICATING	Client or equipment failed to authenticate, an error occurred during the process, or the client has no corresponding ACLEntry.	(no state)	Error reported and connection is terminated.	
3	AUTHENTICATING	Request to establish new session received but max session limit would be exceeded, or request is for a new admin session and there is already an active admin session.	(no state)	Error reported and connection is terminated.	
4	AUTHENTICATING	Client and equipment successfully authenticated with no errors and the request to establish new session was successful.	ESTABLISHED	Equipment enters the COMMUNICATING and MONITORING sub-states for the session. Active session count is incremented if non-admin session.	
5	ESTABLISHED	Entry to COMMUNICATING, client authenticated and a session is established or resumed.	READY	Equipment is waiting for service requests for this session.	COMMUNICATING and MONITORING are concurrent sub-states of ESTABLISHED.
6	READY	Equipment receives a client request for service.	READY	Equipment verifies session id and permission for the client to execute requested service based on granted privileges.	
7	ESTABLISHED	Entry to MONITORING, client authenticated and a session is established or resumed.	SESSION MONITOR	Equipment sends a ping message to client to indicate session is established. When resuming persistent sessions during startup, indicates equipment is now online.	Concurrent with COMMUNICATING state.
8	SESSION MONITOR	Ping delay timer reached	SESSION MONITOR	Equipment sends ping message to client. If ping is not replied within configurable timeout, or after the configured number of retries, the session will be closed by equipment.	See Section 14.3.2.7.
9	ESTABLISHED	Client or SecurityAdmin request to terminate session, equipment is shutting down, or communication errors have been detected by the equipment.	CLOSING	Equipment properly closes session and clears any state information. SessionClosed notification is sent to client (if possible).	See Sections 14.3.2.3 and 13.1.4 for discussion of equipment response to communication errors.

#	<i>Current State</i>	<i>Trigger</i>	<i>New State</i>	<i>Actions</i>	<i>Comments</i>
10	ESTABLISHED	Equipment is shutting down or is in emergency down, and session is persistent	FROZEN	SessionFrozen notification is sent to client (if possible).	Applies to sessions that are persistent only, see PersistSession (Section 13.2.3).
11	FROZEN	Equipment has completed startup or has otherwise entered a state where it is able to resume authenticated session communications.	ESTABLISHED	Session context including client endpoint, client id, session id etc is restored from non-volatile memory. MONITORING and COMMUNICATING sub-states will be entered, see transition 7 for notifications to client on online status.	In attempting to resume a frozen session, the client may not actually be reachable by the equipment. In such cases, the session will eventually be closed according to the SessionPing messages sent from the SessionMonitor sub-state (see transitions 8 and 9).
12	CLOSING	Session terminated	(no state)	Session is deleted and active session count decremented if non-admin session.	

15 Client Interface

15.1 This section specifies the interface that the equipment shall use to communicate security related information and requests/events to authenticated clients with established sessions.

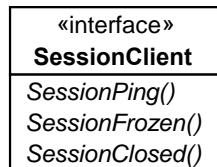


Figure 17
Session Client Interface

Table 35 Session Client Operations

<i>Operation</i>	<i>Description</i>	<i>Type</i>	<i>Requestor/ Sender</i>	<i>Responder/ Receiver</i>
SessionPing	Used by equipment to check if client is still active.	RR	Equipment	Client
SessionFrozen	Notification to client that current session will be frozen.	FF	Equipment	Client
SessionClosed	Used by equipment to close an active session.	RR	Equipment	Client

15.1.1 *SessionPing* — Upon receiving this request, the client shall reply with its id. The client id returned must be the same one used by the client during the authentication phase. See the Session Monitoring state defined in Section 14.3.2.7, regarding behavior of established authenticated sessions in the event of ping timeouts.

Table 36 SessionPing Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
clientId	Id of the client responding to the ping message.	OUT	Text, equal to the client Id used in authentication phase.



Parameter	Description	Kind	Form
error	Unrecognized session.	error	Text, equal to unrecognized session error, including the bad session id.

15.1.2 *SessionFrozen* — This is a notification message sent by the equipment to notify clients with persistent sessions that communications will be put on hold, either due to communications disabled or equipment shutdown. From the client's perspective, this means session pings will not be answered by the equipment and no notifications will be sent by the equipment until a ping message is received from the equipment indicating that the session is again resumed. The equipment shall send one notification for each persistent session. See session state transitions (Table 31) into and out of frozen state for respective behavior on freezing and resuming sessions.

15.1.2.1 There are no parameters associated with this service.

15.1.3 *SessionClosed* — This is a notification message sent by the equipment to notify the client that this session is being terminated, and is sent whenever a session is closed.

Table 37 SessionClosed Parameter Definitions

Parameter	Description	Kind	Form
sessionId	Identifier of the session being terminated.	IN	Text, equal to the id of the session being terminated.

16 Compliance

16.1 Fundamental Requirements

16.1.1 This specification defines the standard services and mechanisms to support authentication of clients to the equipment for session establishment and access control and to support credential and privilege/policy administrations. Not all capabilities need to be supported for a particular implementation. The following constitutes fundamental security requirements that all compliant implementations must support:

- Authentication services and state model, Section 10
- Authorization, Section 11
- Session and ACL administration, Section 11.2.10.3
- Authenticated session management, Sections 13 and 15
- All communication and authenticated session state models, Section 14

16.2 Optional Capabilities

16.2.1 The following capabilities defined in this standard can optionally be implemented depending on the needs of the equipment:

- Equipment console administration interface, Section 12.4

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature, respecting any materials or equipment mentioned herein. These standards are subject to change without notice.

By publication of this standard, Semiconductor Equipment and Materials International (SEMI) takes no position respecting the validity of any patent rights or copyrights asserted in connection with any items mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights are entirely their own responsibility.



SEMI E132.1-0305

PROVISIONAL SPECIFICATION FOR SOAP BINDING FOR EQUIPMENT CLIENT AUTHENTICATION AND AUTHORIZATION (ECA)

This provisional specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on December 10, 2005. Initially available at www.semi.org February 2005; to be published March 2005.

Table of Contents

1 Purpose	4
1.1 Specification Purpose	4
2 Scope	4
2.1 Specification Scope	4
3 Limitations.....	4
3.1 Provisional Specification.....	4
4 Referenced Standards	4
4.1 SEMI Standards	4
4.2 OMG Standards	4
4.3 W3C Standards	5
4.4 Other Standards	5
5 Terminology	5
5.1 Abbreviations and Acronyms	5
5.2 Definitions	5
6 Conventions	6
6.1 Translating UML to XML Schema	6
6.2 Documenting XML Schema and WSDL Files	7
6.3 Documenting XML Schema with Diagrams.....	7
6.4 XML Schema Sample	8
6.5 Translating UML to WSDL.....	9
7 SEMI E132 Authentication Mapping to SSL	10
7.1 SSL Overview.....	10
7.2 Credentials and Digital Certificates	11
7.3 Enabling/Disabling SSL Authentication	13
8 Mapping of SEMI E132 UML to XML Schema and WSDL	13
8.1 WSDL Organization.....	13
8.2 SecurityAdmin Interface	14
8.3 SessionManager Interface	33
8.4 SessionClient Interface	41

List of Figures

Figure 1 XML Schema Example Diagram	8
Figure 2 XML for Sample	9
Figure 3 SSL Basic Handshake Flow	11
Figure 4 SEMI E132.1 Security Credentials.....	12
Figure 5 XML Schema and WSDL File Organization	14
Figure 6 XML Schema and WSDL files for SEMI E132	14
Figure 7 Mapping the SecurityAdmin interface to a WSDL portType.....	15
Figure 8 Sample SecurityAdmin WSDL Service definition	16
Figure 9 UnauthorizedOperationType XML Schema Type	17
Figure 10 E132Header XML Schema Element	18
Figure 11 GetDefinedPrivilegesRequest XML Schema Element	19
Figure 12 GetDefinedPrivilegesResponse XML Schema Element	19
Figure 13 DefinedPrivilegesType XML Schema Type	19



Figure 14 GetDefinedPrivilegesErrorType XML Schema Type	20
Figure 15 GetACLRequest XML Schema Element.....	21
Figure 16 GetACLResponse XML Schema Element	21
Figure 17 ACLEntrySelectorType XML Schema Type	22
Figure 18 PrivilegeAssignment XML Schema Type.....	22
Figure 19 RoleAssignment XML Schema Type.....	23
Figure 20 AddACLEntryRequest XML Schema Element.....	24
Figure 21 AddACLEntryResponse XML Schema Element	25
Figure 22 UnrecognizedPrivilegeType XML Schema Type	25
Figure 23 DeleteACLEntryRequest XML Schema Element	26
Figure 24 DeleteACLEntryResponse XML Schema Element.....	27
Figure 25 GetActiveSessionsRequest XML Schema Element	28
Figure 26 GetActiveSessionsResponse XML Schema Element.....	28
Figure 27 ActiveSession XML Schema Type	29
Figure 28 SetMaxSessionsRequest XML Schema Element	30
Figure 29 SetMaxSessionsResponse XML Schema Element.....	31
Figure 30 GetMaxSessionsRequest XML Schema Element.....	32
Figure 31 GetMaxSessionsResponse XML Schema Element	32
Figure 32 Sample SessionManager WSDL Service Definition	35
Figure 33 EstablishSessionRequest XML Schema Element.....	35
Figure 34 EstablishSessionResponse XML Schema Element	36
Figure 35 PersistSessionRequest XML Schema Element.....	37
Figure 36 PersistSessionResponse XML Schema Element	37
Figure 37 SessionPingRequest XML Schema Element	38
Figure 38 SessionPingResponse XML Schema Element	39
Figure 39 CloseSessionRequest XML Schema Type	40
Figure 40 CloseSessionResponse XML Schema Element.....	40
Figure 41 E132HashHeader Schema Element.....	41
Figure 42 Sample SessionClient WSDL Service definition	43
Figure 43 SessionFrozenNotification XML Schema Element.....	45
Figure 44 SessionClosedNotification XML Schema Element.....	46

List of Tables

Table 1 Example Translation Table.....	6
Table 2 Example Translation Table for Operation Input/Output Arguments	7
Table 3 Example Schema/WSDL Document Description Table	7
Table 4 Altova XMLSPY Schema Diagram Symbols.....	7
Table 5 Example Interface WSDL Port Type Table.....	9
Table 6 Example Interface WSDL Binding Table.....	9
Table 7 Example Operation Binding Table	10
Table 8 Example PortType Operation Table	10
Table 9 XML Schema.....	14
Table 10 SecurityAdmin PortType Definitions	15
Table 11 SecurityAdmin Binding Definitions	15
Table 12 SecurityAdmin Port Type	16
Table 13 SecurityAdmin Binding	16
Table 14 SEMI E132 Error Codes	17
Table 15 Translation Table for UnauthorizedOperationError Class.....	17
Table 16 Translation Table for GetDefinedPrivileges Output Parameters	19
Table 17 Translation Table for DefinedPrivilege Class.....	19
Table 18 GetDefinedPrivileges Messages	20
Table 19 GetDefinedPrivileges PortType Operation	20
Table 20 GetDefinedPrivileges Operation Binding	20
Table 21 Translation Table for GetACL Output Parameters	21
Table 22 Translation Table for PrivilegeAssignment Class	22



Table 23 Translation Table for RoleAssignment Class	23
Table 24 Translation Table for Subject Class.....	23
Table 25 Translation Table for Privilege Class	23
Table 26 GetACL Messages	24
Table 27 GetACL PortType Operation.....	24
Table 28 GetACL Operation Binding.....	24
Table 29 Translation Table for AddACLEntry Input Parameters.....	24
Table 30 Translation Table for AddACLEntry Output Parameters	25
Table 31 Translation Table for UnrecognizedPrivilegeError Class.....	25
Table 32 AddACLEntry Messages.....	26
Table 33 AddACLEntry PortType Operation.....	26
Table 34 AddACLEntry Operation Binding.....	26
Table 35 Translation Table for DeleteACLEntry Input Parameters	26
Table 36 Translation Table for DeleteACLEntry Output Parameters.....	27
Table 37 DeleteACLEntry Messages	27
Table 38 DeleteACLEntry PortType Operation	27
Table 39 DeleteACLEntry Operation Binding	27
Table 40 Translation Table for GetActiveSessions Output Parameters.....	28
Table 41 Translation Table for Subject Class.....	29
Table 42 Translation Table for HTTPEndPoint Class	29
Table 43 GetActiveSessions Messages.....	29
Table 44 GetActiveSessions PortType Operation	30
Table 45 GetActiveSessions Operation Binding	30
Table 46 Translation Table for SetMaxSessions Input Parameters	30
Table 47 Translation Table for SetMaxSessions Output Parameters.....	30
Table 48 SetMaxSessions Messages.....	31
Table 49 SetMaxSessions PortType Operation	31
Table 50 SetMaxSessions Operation Binding	31
Table 51 Translation Table for GetMaxSessions Output Parameters	32
Table 52 GetMaxSessions Messages	32
Table 53 GetMaxSessions PortType Operation	33
Table 54 GetMaxSessions Operation Binding	33
Table 55 XML Schema.....	33
Table 56 SessionManager PortType Definitions	33
Table 57 SessionManager Binding Definitions	34
Table 58 SessionManager Port Type	34
Table 59 SessionManager Binding	34
Table 60 Translation Table for EstablishSession Input Parameters	35
Table 61 Translation Table for EstablishSession Output Parameters	36
Table 62 EstablishSession Messages.....	36
Table 63 EstablishSession PortType Operation.....	36
Table 64 EstablishSession Operation Binding	36
Table 65 Translation Table for PersistSession Input Parameters.....	37
Table 66 Translation Table for PersistSession Output Parameters	37
Table 67 PersistSession Messages	38
Table 68 PersistSession PortType Operation.....	38
Table 69 PersistSession Operation Binding	38
Table 70 Translation Table for SessionPing Output Parameters	39
Table 71 SessionPing Messages	39
Table 72 SessionPing PortType Operation	39
Table 73 SessionPing Operation Binding	39
Table 74 Translation Table for CloseSession Input Parameters	40
Table 75 Translation Table for CloseSession Output Parameters	40
Table 76 CloseSession Messages	40
Table 77 CloseSession PortType Operation	41
Table 78 CloseSession Operation Binding	41



Table 79 XML Schema.....	41
Table 80 SessionClient PortType Definitions.....	42
Table 81 SessionClient Binding Definitions.....	42
Table 82 SessionClient Port Type	42
Table 83 SessionClient Binding	43
Table 84 Translation Table for SessionPing Output Parameters	43
Table 85 SessionPing Messages	44
Table 86 SessionPing PortType Operation	44
Table 87 SessionPing Operation Binding	44
Table 88 SessionFrozen Messages	45
Table 89 SessionFrozen PortType Operation	45
Table 90 SessionFrozen Operation Binding	45
Table 91 Translation Table for SessionClosed Input Parameters	46
Table 92 SessionClosed Messages	46
Table 93 SessionClosed PortType Operation	46
Table 94 SessionClosed Operation Binding	46

1 Purpose

1.1 Specification Purpose

1.1.1 The purpose of this specification is to provide an authentication implementation using SSL and a WSDL/SOAP implementation and XML Schema that corresponds to the UML model for Equipment Client Authentication and Authorization as defined by SEMI E132.

2 Scope

2.1 Specification Scope

2.1.1 The scope of this document is the faithful representation of the ECA model in an XML schema and corresponding WSDL services. It will not add new domain information or concepts to the model. The only additions made are those needed to render useful WSDL or XML Schema.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard to establish appropriate safety and health practices and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 Provisional Specification

3.1.1 This specification is provisional pending the approval of SEMI E138. This specification relies on XML Schema types defined in Core Components, and cannot be fully implemented without these types. Finalization of that specification is a condition for the removal of the provisional status of this document.

4 Referenced Standards

4.1 SEMI Standards

SEMI E121 — Guide for Style & Usage of XML for Semiconductor Manufacturing Applications

SEMI E132 — Specification for Equipment Client Authentication and Authorization

4.2 OMG Standards¹

Unified Modeling Language (UML) Specification, Version 1.4, OMG Specification 01-09-67, (http://www.omg.org/technology/documents/modeling_spec_catalog.htm).

¹ Object Management Group, Inc., 250 First Ave. Suite 100, Needham, MA 02494, U.S.A., Phone: 781.444.0404, Fax: 781.444.0320, website: www.omg.org.



4.3 W3C Standards²

Extensible Markup Language (XML) 1.0 (Second Edition) — W3C, 6 October 2000 (<http://www.w3.org/TR/2000/REC-xml-20001006/>).

Namespaces in XML — W3C, 14 January 1999 (<http://www.w3.org/TR/1999/REC-xml-names-19990114/>).

XML Schema Part 0: Primer — W3C, 2 May 2001 (<http://www.w3.org/TR/xmlschema-0/>).

XML Schema Part 1: Structures — W3C, 2 May 2001 (<http://www.w3.org/TR/xmlschema-1/>).

XML Schema Part 2: Datatypes — W3C, 2 May 2001 (<http://www.w3.org/TR/xmlschema-2/>).

XML Path Language (Xpath) — W3C, 16 November 1999 (<http://www.w3.org/TR/xpath/>).

Web Services Description Language (WSDL) 1.1 — W3C Note, (<http://www.w3.org/TR/wsdl>)

Simple Object Access Protocol (SOAP) 1.1 — W3C Note, (<http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>)

4.4 Other Standards

Secure Sockets Layer (SSL) Protocol 3.0 (available from <http://wp.netscape.com/eng/ssl3/draft302.txt>)

NIST FIPS PUB 180-1 — Secure Hash Algorithm (SHA-1) (<http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.pdf>)

Web Services Interoperability (WS-I): Basic Profile Version 1.0a (<http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html>)

Internet Engineering Task Force (IETF) RFC 2459 — Internet X.509 Public Key Infrastructure Certificate and CRL Profile (<http://www.ietf.org/rfc/rfc2459.txt>)

PKCS #12: Personal Information Exchange Syntax Standard, version 1.0 (available from <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/>)

Universal Unique Identifier (UUID) — ISO/IEC 11578:1996 Information technology – Open Systems Interconnection – Remote Procedure Call (<http://www.iso.ch/cate/d2229.html>)

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 Abbreviations and Acronyms

5.1.1 *HTTP* — Hyper Text Transport Protocol

5.1.2 *SHA* — Secure Hash Algorithm

5.1.3 *SOAP* — Simple Object Access Protocol

5.1.4 *SSL* — Secure Sockets Layer

5.1.5 *UML* — Unified Modeling Language

5.1.6 *W3C* — World Wide Web Consortium

5.1.7 *WSDL* — Web Service Description Language

5.1.8 *XML* — eXtensible Markup Language

5.2 Definitions

5.2.1 *UML (Unified Modeling Language)* — a notation for representing object-oriented designs and views created by Booch, Rumbaugh, and Jacobson in order to merge their three popular notations plus aspects of other existing notations into a single object-oriented notation intended to be usable by all.

² World Wide Web Consortium, Massachusetts Institute of Technology (MIT), Computer Science and Artificial Intelligence Laboratory (CSAIL), 32 Vassar Street Room 32-G515, Cambridge, MA 02139, USA, Telephone: 617.253.2613, Fax: 617.258.5999, website: www.w3.org.



5.2.2 *XML (eXtensible Markup Language)* — a markup language used for representing data rich with context and content in documents and in communications. XML is an extension of SGML, a document-oriented markup language. It was created by W3C for use on the Internet.

6 Conventions

6.1 Translating UML to XML Schema

6.1.1 The reader is expected to have a working knowledge of the UML, XML, and Schema specifications (See ¶¶4.2 and 4.3). This document does not provide tutorial information on these subjects.

6.1.2 This document follows the guidelines for XML as outlined in SEMI E121.

6.1.3 Some of the key guidelines followed in this document are summarized here:

- Attributes of UML classes are generally represented as XML attributes, to improve efficiency in transferring metadata descriptions from the equipment. Exceptions to this convention are UML attributes that correspond to data structures or other constructs that cannot be represented as XML attributes, or may include text values that may need to include non-parsed character data (CDATA).
- Composition associations are mapped to a single XML element (or multiple such elements if the specified multiplicity is greater than 1).
- Aggregations are modeled as contained elements or arrays of references to one or more unique attributes of the target type.
- Inheritance is modeled using complexType extension. In cases where a reference to an abstract base class occurs in the UML model, a choice compactor for each of the possible derived types is used in order to restrict instance documents from including arbitrary extension through type substitution.

6.1.4 The translation of a UML class to XML Schema types is documented using a table format illustrated by Table 1. Operations defined for a UML class are translated into WSDL operations.

6.1.5 Translation Table Column Header Description

- *Attribute or Role Name* — If an attribute, the name of the attribute is placed here. If an association (including aggregation or composition), the role name from the UML diagram is placed here. Compositions are often not assigned roll names. In that case “none” is placed here.
- *UML Name/Type* — If an attribute, the data type of the UML attribute is place here. If an association, the type of association is placed here. The possible types are “Composition”, “Aggregation”, or the basic “Association”. UML defines these three types of association.
- *XML Element or Attribute* — Lists the type of XML construct used to represent the UML attribute or association.
- *XML Name/Type* — Provides the name and data type of the resulting XML construct. The type may be a built-in type (for example, xs:string), or a named type defined within the XML Schema.

Table 1 Example Translation Table

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
friend	association	element	Friend: HumanReferenceArray
employees	aggregation	element	Employees: HumanArray
family	composition	element	Family: HumanArray
name	string	element	Name: xs:string

6.1.6 Translating Operations to XML Schema

6.1.6.1 The translation of an operation defined for UML interface classes to XML Schema types is documented using a table format illustrated in Table 2. One row is provided for each possible input/output argument for the operation being described.

Table 2 Example Translation Table for Operation Input/Output Arguments

Argument Name	Format	XML Element or Attribute	XML Name/Type
<argument name from abstract specification>	<argument format from abstract specification>	<whether the argument is modeled as an Element or Attribute>	<namespace-qualified name of the attribute/element>

6.2 Documenting XML Schema and WSDL Files

6.2.1 Associated with this document are XML Schema and WSDL files that are core components of the specification. Each such document is described using a table format illustrated by Table 3.

Table 3 Example Schema/WSDL Document Description Table

<i>File Name</i>	<.xsd or .wsdl file name>
<i>Target Namespace</i>	<target namespace defined in the schema or WSDL file>
<i>Imported/Referenced Namespaces</i>	<namespaces imported or used by the schema or WSDL file>
<i>Description</i>	<brief description of the purpose/function of the schema or WSDL file>

6.3 Documenting XML Schema with Diagrams³

6.3.1 This document provides graphical representations of the included XML. Although no standard graphical notation for XML could be found, various XML tools have their own notation. This document will use the notation provided by XMLSpy from Altova Corporation. Figure 1 shows a sample XML diagram that will be used to provide a basis for explanation of the XML graphical notation used in the rest of the document.

6.3.2 In the diagram, rectangular boxes represent XML elements. Ownership or containment is read from right to left in the diagrams. In the sample diagram, ParentType contains Child1, Child2, Child 3, and Child 4. In turn, Child2 contains Child2a, Child2b, and Child2c. The additional symbols (8-sided boxes) represent sequences or choices. See Table 4 for an explanation of these symbols.

Table 4 Altova XMLSPY Schema Diagram Symbols

	Denotes a required ordered sequence of the right hand elements with a cardinality of one for each element (sequence).
	Denotes an optional ordered sequence of the right hand elements with a cardinality of one for each element (sequence).
	Denotes a required, but unordered, sequence of the right hand elements with a cardinality of one for each element (all).
	Denotes a required choice of the right hand elements. Exactly one or two of the right hand elements must be present (choice).
	Denotes an element that contains parsed character data.
	Denotes a reference to a group or element defined elsewhere.

6.3.3 A graphic using a solid line is a required element; using a dashed line represents an optional element. Numbers or ranges in the lower right hand corner represent cardinality. The default cardinality is one.

³ All images/graphics were created using Altova's XMLSPY®. Copyright 2003 Altova GmbH and reprinted with permission of Altova.

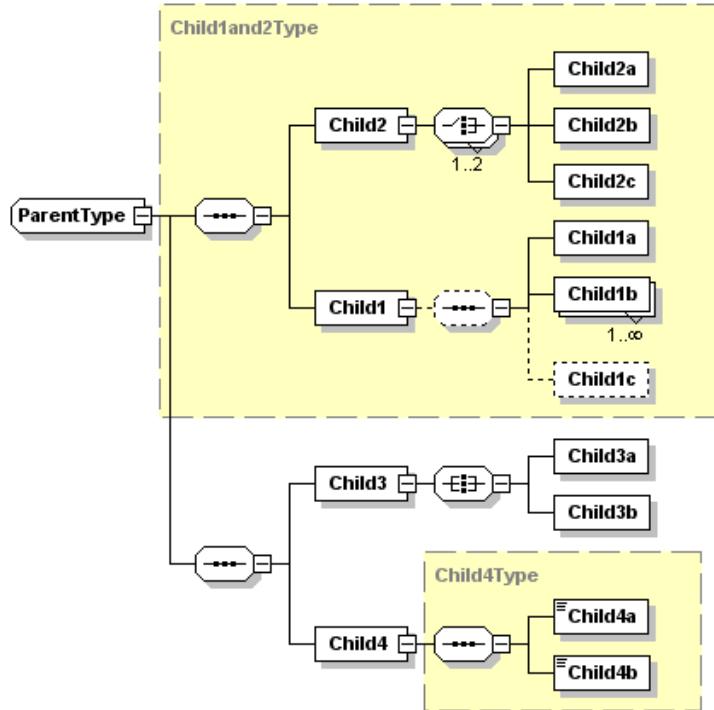


Figure 1
XML Schema Example Diagram

6.3.4 To simplify a diagram or help focus on a particular aspect, detail may be hidden. The 8-sided symbols have a small square on the right end. If a minus sign “-” is in the box, then all detail is shown. If the box contains a plus sign “+”, then all detail to the right of that symbol is hidden. The example has no hidden detail.

6.3.5 The yellow (or grey if printed in monochrome) boxes indicate the use of other defined types. So, **Child4** is of type “**Child4Type**”. **Child4Type** defines **Child4a** and **Child4b**. This detail may be hidden in the diagram. Object oriented inheritance is typically represented in XML as type extension. In Figure 1, **ParentType** extends **Child1and2Type** by adding a sequence that includes **Child3** and **Child4**.

6.3.6 Reading Figure 1 would yield the following additional information:

1. **Child1and2Type** is an ordered sequence of two items: **Child2** and **Child1**.
2. **Child1** contains an optional ordered sequence of **Child1a**, one or more **Child1b**, and (optionally) **Child1c**.
3. **Child2** contains a choice of one or two of the following: **Child2a**, **Child2b**, and **Child2c**.
4. **Child3** contains an unordered sequence of **Child3a** and **Child3b**.

6.4 XML Schema Sample

6.4.1 The sample XML Schema for the example shown in Figure 1, is presented below. Refer to the XML documentation referenced in ¶4.3 for a complete description of the syntax and semantics of XML Schema.

```

<xs:element name="Parent">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Child1">
        <xs:complexType>
          <xs:sequence minOccurs="0">
            <xs:element name="Child1a"/>
            <xs:element name="Child1b" maxOccurs="unbounded"/>
            <xs:element name="Child1c" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Child2">
        <xs:complexType>
          <xs:choice maxOccurs="2">
            <xs:element name="Child2a"/>
            <xs:element name="Child2b"/>
            <xs:element name="Child2c"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <xs:element name="Child3">
        <xs:complexType>
          <xs:all>
            <xs:element name="Child3a"/>
            <xs:element name="Child3b"/>
          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Figure 2
XML for Sample

6.5 Translating UML to WSDL

6.5.1 In general, UML interface classes defined in the abstract specification are translated into WSDL as portType definitions, and each portType definition has a corresponding WSDL binding definition. The following tables show the convention used for documenting the WSDL port type and binding definitions for a given UML interface class.

Table 5 Example Interface WSDL Port Type Table	
<i>Class Name</i>	<UML interface name from abstract specification>
<i>WSDL Port Type Name</i>	<port type name used in WSDL>
<i>SEMI E125 Operation → WSDL Operation</i>	<operation name from UML interface> → <WSDL port type operation name>

Table 6 Example Interface WSDL Binding Table

<i>SEMI E125 Class Name</i>	<UML interface name from abstract specification>
<i>WSDL Binding Name</i>	<binding name used in WSDL>
<i>SOAP Binding Style</i>	<RPC or document>
<i>SOAP Transport</i>	<transport identifying URI>

6.5.2 Each operation defined for a given UML interface class has a corresponding operation definition and binding. Operations are described using a table format illustrated by Table 7 and Table 8.



Table 7 Example Operation Binding Table

<i>SOAPAction</i>	<the SOAPAction HTTP header value to be used for this operation>
<i>Input Headers (WSDL Message, Required)</i>	<the name of the WSDL message providing input headers, and whether or not the headers are required>
<i>Output Headers (WSDL Message, Required)</i>	<the name of the WSDL message providing output headers, and whether or not the headers are required>

Table 8 Example PortType Operation Table

<i>Input Message Name</i>	<WSDL message name used as input for the WSDL port type operation>
<i>Output Message Name</i>	<WSDL message name used as output for the WSDL port type operation>

7 SEMI E132 Authentication Mapping to SSL

7.1 SSL Overview

7.1.1 This section describes the use of SSL security protocol and supporting technology to implement the authentication scheme described in SEMI E132. SSL is application protocol independent and transparently supports any higher layered transport protocols such as SOAP over HTTP.

7.1.2 Figure 3 presents the basic message flow (handshake) of SSL to establish a secure connection and how it maps to the SEMI E132 authentication messages, please refer to the SSL documentation referenced in ¶4.4 for complete specification on SSL. This specification defines several restrictions on the capabilities of SSL to the minimum necessary to support SEMI E132 authentication. These restrictions are described in more detail in the following section.

7.1.3 SEMI E132 Restrictions on SSL

7.1.3.1 X.509 v3 public key certificates as defined by RFC 2459 must be used to establish equipment and client identities, see X.509 documentation referenced in ¶4.4 for complete details of X.509 structure and content. The certificate is sent when required during the SSL handshake.

7.1.3.2 Mutual authentication is required. Equipment shall always send its certificate and request for the client's certificate as part of the SSL handshake. Correspondingly, client will send its own certificate and the certificate verify message to equipment to allow client authentication.

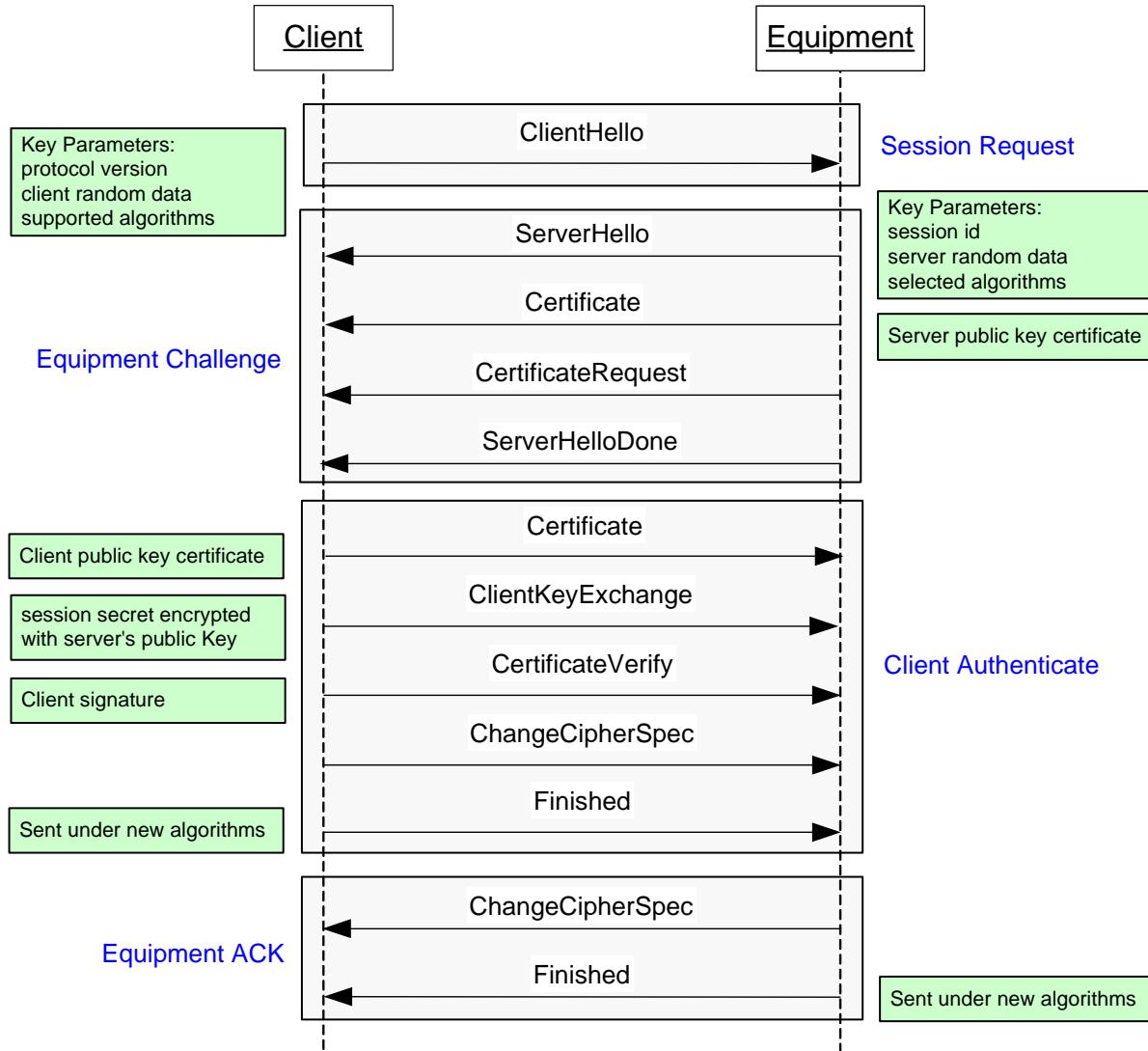
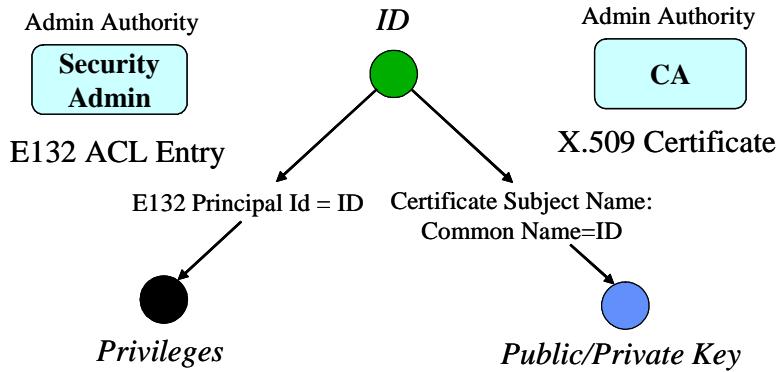


Figure 3
SSL Basic Handshake Flow

7.2 Credentials and Digital Certificates

7.2.1 Overview

7.2.1.1 X.509 certificates shall be used to establish identities for equipment and application clients. The certificate is used by SSL for authentication and to establish secure connections between equipment and clients. An illustration of the SEMI E132.1 security credentials showing role of certificates and ACL is presented in Figure 4. As shown, an X.509 certificate binds an ID to a public key, managed by an agent called the CA described in following sections. An ACL Entry binds the ID to a set of authorizations, managed by the Security Admin. Authorization management, ACL Entry and Security Admin are as defined in SEMI E132, see Authorizations section.



**Figure 4
SEMI E132.1 Security Credentials**

7.2.1.2 CA is the Certification Authority that issues certificates to equipment and application clients wishing to communicate over SSL. The authenticity of the certificate is verified by the CA signature within the certificate, in effect the CA is certifying the validity of the information contained in the certificate, more importantly it certifies the binding between the entity identified in the certificate and its public key used for authentication purposes in SSL. For more information on certificate validation and signature verification, please see referenced X.509 v3 documentation in ¶4.4. The CA may or may not be the same as the Security Admin as defined in SEMI E132, which manages access control permissions for application clients. Implementation and maintenance of both the CA and the Security Admin are the responsibility of the equipment user, including certificate issuance and security credentials management. Equipment suppliers shall not be required to provide CA or Security Admin implementations.

7.2.1.3 In this specification, only the common name (CN) attribute of the certificate's subject distinguished name (DN) is required, this is represented as ID in Figure 4. The same ID must be used by the Security Admin when binding authorizations to a principal id in an ACLEntry. In summary, SEMI E132 principals are identified by the common name (CN) attribute of the subject distinguished name field of the certificate issued to the principal.

7.2.1.4 Certificates issued by the CA for SEMI E132 shall be encoded in PKCS#12 format along with corresponding private keys. Equipment shall support PKCS#12 to decode certificate and any associated private key issued by the CA. There are no restrictions on the local encoding format of certificates. See PKCS#12 documentation referenced in ¶4.4 for complete details. The CA certificate will also be delivered in same format but without the private key. The CA certificate is used by SSL for certificate verification.

7.2.1.5 Certificates must minimally support signing capability if the KeyUsage extension field is present in the certificate, more specifically the digitalSignature bit must be asserted. Please see the X.509 v3 specification for more information on certificate profile.

7.2.2 Establishing Equipment Credentials

7.2.2.1 Equipment must install both the CA's public certificate and its own certificate along with the corresponding private key, the CA certificate to verify client certificates are from the same trusted CA. Equipment is responsible for decoding the PKCS#12 encoded certificates and private key from CA to a format supported by its implementation.

7.2.2.2 An ACLEntry defining the Security Admin for the equipment must be added using the local equipment interface. This shall be done at time of equipment install to allow remote administration of access permissions for the equipment.

7.2.3 Establishing Application Credentials

7.2.3.1 Application clients wishing to establish SEMI E132 sessions with any equipment must first possess a valid certificate issued by the CA prior to any communications. Equipment shall ensure SSL handshake terminates for any client that does not present a valid certificate issued by a trusted CA. Similar to equipment certificates, all client certificates and corresponding private key from CA will be encoded in PKCS#12 format. The CA certificate must also be installed by the application client to verify equipment certificate chain.



7.2.3.2 Each client must also have an ACLEntry defined on the equipment granting the appropriate privileges to support its functions prior to establishing SEMI E132 sessions with the equipment. This can be done via the Security Admin or via the equipment console.

7.2.3.3 Equipment shall reject all SEMI E132 establish session requests if client does not have an ACLEntry defined, even if client possesses a valid certificate issued by trusted CA.

7.3 Enabling/Disabling SSL Authentication

7.3.1 There may be some integration, development, and/or test environments in which it is not practical or feasible to install equipment and/or client certificates in order to communicate with the equipment. As a practical consideration for such environments only, the equipment shall support the ability of the user to enable or disable SSL authentication at the equipment.

7.3.2 If enabled, all SEMI E132-based communication with the equipment shall take place via mutually-authenticated SSL as described in ¶7.1.

7.3.3 If disabled, all SEMI E132-based communication with the equipment shall take place via the normal, unauthenticated HTTP transport. While disabled, requests shall still follow the normal authorization verifications using the client ID provided in the FROM field of the SEMI E132 header; however, because requests are not authenticated there can be no guarantee that the requesting application is authentic. Users should be aware that disabling SSL authentication provides unrestricted access to all SEMI E132 equipment operations for any application (factory authorized or not) with network access to the equipment.

7.3.4 Equipment shall provide a configuration option to allow switching between SSL and non-SSL communications. Only one such communication port shall be active at any time. During a configuration change, it is the responsibility of the equipment user to close existing sessions prior to the change, as the active port will no longer be available for requests following the change, and clients will not be notified of the change by the equipment.

7.3.5 After a configuration change, any pre-existing persistent sessions shall remain active as defined in the SEMI E132 Session State Model. Users should be aware that all requests made by those sessions after the change must take place on the new port. The equipment is not required to retain non-persistent sessions in the active state following a configuration change. If an equipment reset is required to change this configuration, then persistent sessions shall be resumed, and non-persistent sessions shall be closed as for any equipment shutdown as defined in the SEMI E132 Session State Model.

8 Mapping of SEMI E132 UML to XML Schema and WSDL

8.1 WSDL Organization

8.1.1 Each interface definition in SEMI E132 is mapped to a WSDL portType and binding definition. WSDL portType definitions are named after the interface and its operations as they appear in SEMI E132. WSDL binding definitions for each portType are used to specify the SOAP 1.1 envelope contents for each operation, and to define the corresponding XML encoding styles and HTTP header usage. All SEMI E132 WSDL interfaces use document/literal encoding, with the complete SOAP header and body contents defined in XML Schema file(s) via global element definitions.

8.1.2 Figure 5 shows the relationship between the WSDL binding and portType definitions and the XML Schema types used for each interface. The portType definition imports XML Schema type definitions used in each operation via the XML Schema “import” statement. The WSDL binding definition imports the portType definition via the WSDL “import” statement.

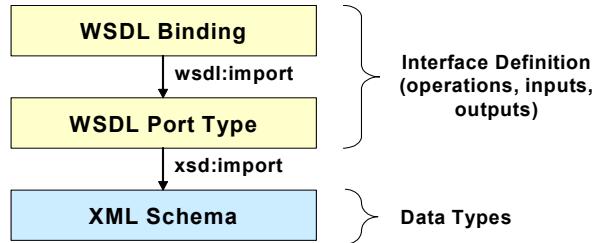


Figure 5
XML Schema and WSDL File Organization

8.1.3 Figure 6 shows the specific XML Schema and WSDL files defined for SEMI E132.

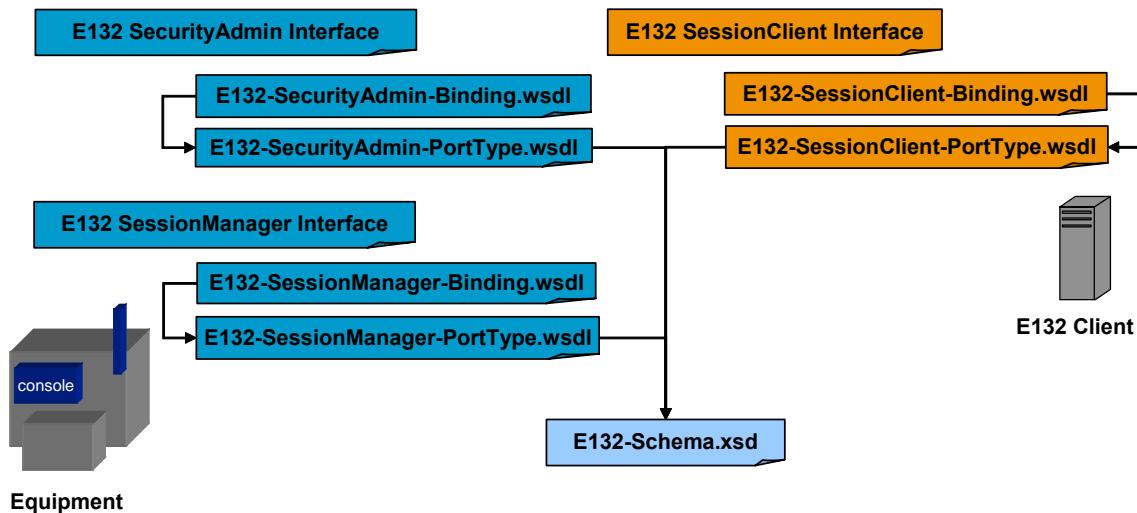


Figure 6
XML Schema and WSDL files for SEMI E132

8.2 SecurityAdmin Interface

8.2.1 Establishing an ACLEntry for the Security Admin

8.2.1.1 An ACLEntry defining the Security Admin shall be added via the local equipment interface at time of equipment install. The subject id of the ACLEntry must match the common name (CN) attribute of the subject distinguished name of the certificate issued to the Security Admin, see ¶7.2.1.3. The Security Admin ACLEntry must grant the reserved privilege “urn:semi-org.auth:securityAdminPrivileges” as defined by SEMI E132.

8.2.1.2 Exactly one ACLEntry shall exist for the Security Admin.

8.2.2 XML Schema and WSDL Files

8.2.2.1 The XML Schema and WSDL defined by this specification for the SecurityAdmin interface is contained in the following documents:

Table 9 XML Schema

<i>File Name</i>	E132-1-V0305-Schema.xsd
<i>Target Namespace</i>	urn:semi-org:xsd.E132-1.V0305.auth
<i>Imported/Referenced Namespaces</i>	http://www.w3.org/2001/XMLSchema urn:semi-org:xsd.CommonComponents.V0305.ccs
<i>Description</i>	This file defines all of the SEMI E132 data types and elements used by the SecurityAdmin, SessionManager and SessionClient interfaces.

Table 10 SecurityAdmin PortType Definitions

<i>File Name</i>	E132-1-V0305-SecurityAdmin-PortType.wsdl
<i>Target Namespace</i>	urn:semi-org:ws.E132-1.V0305.secAdmin-portType
<i>Imported/Referenced Namespaces</i>	urn:semi-org:xsd.E132-1.V0305.auth http://schemas.xmlsoap.org/wsdl/ http://www.w3.org/2001/XMLSchema
<i>Description</i>	This file defines all of the input/output messages and operations for the SecurityAdmin interface, based on the data types defined in the SEMI E132 XML Schema.

Table 11 SecurityAdmin Binding Definitions

<i>File Name</i>	E132-1-V0305-SecurityAdmin-Binding.wsdl
<i>Target Namespace</i>	urn:semi-org:ws.E132-1.V0305.secAdmin-binding
<i>Imported/Referenced Namespaces</i>	urn:semi-org:ws.E132-1.V0305.secAdmin-portType http://schemas.xmlsoap.org/wsdl/soap/ http://schemas.xmlsoap.org/wsdl/
<i>Description</i>	This file binds the abstract portType definition to HTTP and SOAP, specifying required SOAP and HTTP headers for each operation and the XML encoding style for the SecurityAdmin interface.

8.2.2.2 These documents are a part of this specification and should accompany this document. The contents of these documents constitute the core part of this specification.

8.2.3 WSDL Port Type Overview

8.2.3.1 The SecurityAdmin WSDL portType definition organizes the SEMI E132.1 XML Schema data types into a collection of named operations with inputs and outputs that correspond to the UML operations that are defined for the SecurityAdmin interface in SEMI E132. The WSDL portType itself is named after the SEMI E132 interface, and each WSDL portType operation is named after the corresponding UML operation defined for the interface. Each WSDL operation consists of two WSDL message definitions corresponding to the input and output (request and response) for the UML operation defined in SEMI E132. Figure 7 illustrates the relationship between the SEMI E132 UML interface and the WSDL portType definition; Table 12 describes this in tabular form. The WSDL message and operation definitions are described in ¶8.2.9 through 8.2.15.

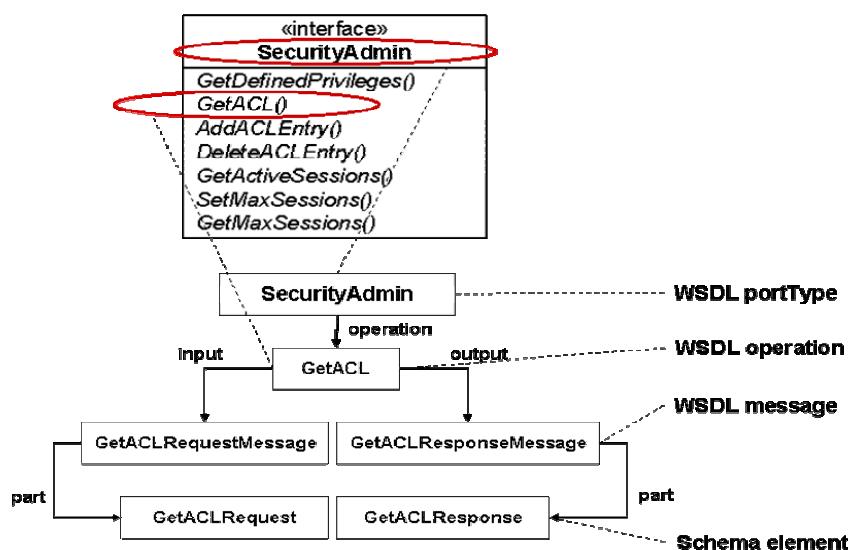


Figure 7
Mapping the SecurityAdmin interface to a WSDL portType



Table 12 SecurityAdmin Port Type

<i>SEMI E132 Class Name</i>	SecurityAdmin
<i>WSDL Port Type Name</i>	SecurityAdmin
<i>SEMI E132 Operation → WSDL Operation</i>	GetDefinedPrivileges → GetDefinedPrivileges GetACL → GetACL AddACLEntry → AddACLEntry DeleteACLEntry → DeleteACLEntry GetActiveSessions → GetActiveSessions SetMaxSessions → SetMaxSessions GetMaxSessions → GetMaxSessions

8.2.4 WSDL Binding Overview

8.2.4.1 The WSDL SecurityAdmin binding definition specifies a message and transport protocol to use for a given portType (SOAP and HTTP for SEMI E132.1), the interface style used (document style for SEMI E132.1). These settings are shown in Table 13. For each WSDL portType operation, the binding defines any SOAP headers used and whether or not they are required, the HTTP SOAPAction header value to use for that operation, and the XML encoding to use (literal for SEMI E132.1). The binding definitions for each portType operation are described in ¶¶8.2.9 through 8.2.15.

Table 13 SecurityAdmin Binding

<i>SEMI E132 Class Name</i>	SecurityAdmin
<i>WSDL Binding Name</i>	SecurityAdminBinding
<i>SOAP Binding Style</i>	document
<i>SOAP Transport</i>	http://schemas.xmlsoap.org/soap/http

8.2.5 WSDL Service Overview

8.2.5.1 This specification does not provide a WSDL service definition. WSDL service definitions provide one way to define and locate the endpoint(s) at which a given interface can be accessed by a client. Such information cannot be determined until the equipment has been installed on a factory network, and therefore is out of scope for this specification. Users are responsible for defining the mechanisms by which clients locate SEMI E132 equipment services in the factory. A sample SecurityAdmin service definition is provided in Figure 8 for reference only.

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SecurityAdmin"
  xmlns="urn:semi-org:ws.E132-1.V0305.secAdmin-service"
  targetNamespace="urn:semi-org:ws.E132-1.V0305.secAdmin-service"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SABind="urn:semi-org:ws.E132-1.V0305.secAdmin-binding">

  <wsdl:import namespace="urn:semi-org:ws:E132-1.V0305.secAdmin-binding"
    location="E132-1-V0305-SecurityAdmin-Binding.wsdl"/>

  <wsdl:service name="SecurityAdmin">
    <wsdl:port name="SecurityAdminPort" binding="SABind:SecurityAdminBinding" >
      <soap:address location="https://www.someplace/SecurityAdmin"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

Figure 8
Sample SecurityAdmin WSDL Service definition

8.2.6 SEMI E132 Operation Errors

8.2.6.1 Common Error

8.2.6.1.1 All errors returned by SEMI E132 operations uses the Error type defined in SEMI E138. The common error is in addition to any custom error construct that may be defined by each operation to provide additional context relevant to the operation and a specific error. The common error type defines 2 attributes, “source” and “code”; see Common Components Specification for more detailed information. All E132 defined errors shall set the “source” attribute to “urn:semi-org:E132”. The error codes for SEMI E132 defined errors are provided in Table 14. Description is a textual description of the error. The Extension element allows equipment suppliers to provide more detailed information regarding the error using XML content defined by the supplier.

8.2.6.1.2 Table 14 below provides a mapping of the SEMI E132 defined errors to its common error code. The SEMI E132 Error column matches the error description of each defined error in the SEMI E132 specification. Errors during the Authentication message exchange are as defined by the SSL authentication protocol. The SEMI E132 mapped error codes below are in addition to any common error codes defined in the Common Components Specification.

Table 14 SEMI E132 Error Codes

SEMI E132 Error	Common Error Code
Operation not authorized	6000
Duplicate ACL entry found	6001
Unrecognized role	6002
Unrecognized privilege	6003
ACL entry not found	6004
Unrecognized session	6005
Max session limit exceeded	6006

8.2.6.2 Unauthorized Operation Error

8.2.6.2.1.1 Many operations may return an operation not authorized error (common error code 6000). For this error, SEMI E132 defines a specific UnauthorizedOperationError structure to provide additional context information such as list of all the privileges that grant authorization for the requested operation.

8.2.6.2.1.2 The UnauthorizedOperationError class is mapped to the XML UnauthorizedOperationType. Table 15 shows how the attributes and associations of UnauthorizedOperationError class are mapped to XML. Figure 9 shows the XML Schema diagram.

Table 15 Translation Table for UnauthorizedOperationError Class

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
description	Text	Element	Description: xsd:string
requiredPrivileges	Association	Element	RequiredPrivilege: PrivilegeType (one or more)

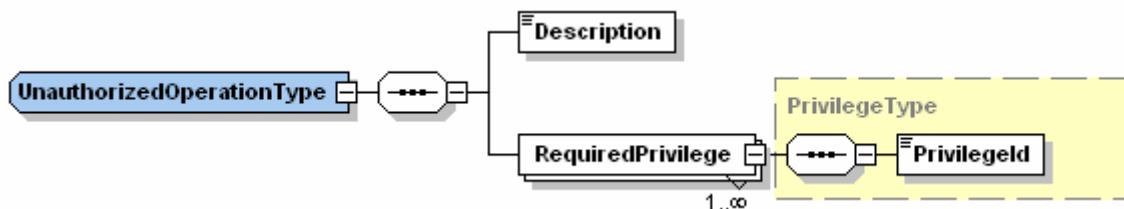


Figure 9
UnauthorizedOperationType XML Schema Type

8.2.7 Session Identifier

8.2.7.1 The session identifier is required to be present in all requests to the equipment over an SEMI E132 authenticated session. The session identifier is generated by the equipment during the Establish Session operation, and must be a UUID that is guaranteed unique across space and time, see documentation referenced in ¶4.4 for specific details on syntax and format of a UUID.

8.2.7.2 In this specification, a header type is defined in the XML Schema file to carry the SEMI E132 session identifier within the SOAP header for SOAP messages defined for this interface. This provides the most transparent way of sending the session identifier without affecting the input/output parameters of the individual operations. Since SEMI E132 requires all requests to the equipment to be sent over SSL, the session identifier need not be encrypted and shall be included in the header unmodified. Figure 10 shows the XML schema diagram for the E132Header schema element. The session identifier is modeled as the SessionID of XML type xsd:string in the E132Header.

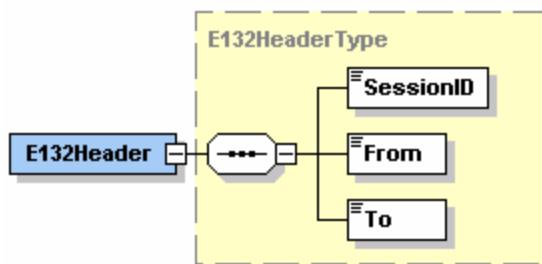


Figure 10
E132Header XML Schema Element

8.2.7.3 The WSDL message “E132HeaderMessage” defines the message used in the SOAP header. It maps to the schema element E132Header as described above, see definitions of “E132HeaderMessage” in SEMI E132-SecurityAdmin-PortType.wsdl.

8.2.7.4 Since SEMI E132 does not require outgoing equipment messages to be sent over SSL, the session identifier must be masked to avoid sending in plaintext. See ¶8.4 of SessionClient interface regarding handling and encoding of session identifier for outgoing equipment messages.

8.2.8 *From/To Header Fields*

8.2.8.1 Each message over SEMI E132 must also provide the originating source and intended destination using the From/To fields of the E132Header, as shown in Figure 10. The “From” element identifies the originator of the message, and the “To” element identifies the intended recipient of the message, both of type xsd:string. In this specification, the ID of the principals that established the SEMI E132 session shall be used for purposes of identifying message source and destination. Although this information can be extracted based on the session identifier, it is desired that the equipment and client be identified explicitly in each message.

8.2.9 *GetDefinedPrivileges Operation*

8.2.9.1 *XML Schema Types*

8.2.9.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the GetDefinedPrivileges operation. Per WSDL convention described, a request and response message is defined for the operation corresponding to the input and output messages. A corresponding schema element and type is defined for the request and response message containing the input and output parameters of the defined operation respectively.

8.2.9.1.2 *GetDefinedPrivileges Request*

8.2.9.1.2.1 The service request is mapped to the XML Schema element GetDefinedPrivilegesRequest which is of type GetDefinedPrivilegesRequestType. There are no input parameters for this operation, thus the request schema element has no content.

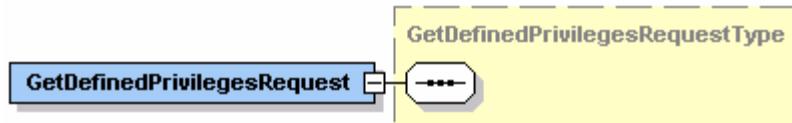


Figure 11
GetDefinedPrivilegesRequest XML Schema Element

8.2.9.1.3 *GetDefinedPrivileges Response*

8.2.9.1.3.1 The service response is mapped to the XML Schema element GetDefinedPrivilegesResponseType of type GetDefinedPrivilegesResponseType. Table 16 shows how the output parameters are mapped to XML. Errors for this operation are returned using the XML schema element Error of type GetDefinedPrivilegesErrorType described in ¶8.2.9.1.5. The XML Schema diagram for the response element is shown in Figure 12.

Table 16 Translation Table for GetDefinedPrivileges Output Parameters

Argument Name	Format	XML Element or Attribute	XML Name/Type
privileges	Unordered list	Element	Privilege: DefinedPrivilegeType (one or more)
error	Structured data	Element	Error: GetDefinedPrivilegesErrorType

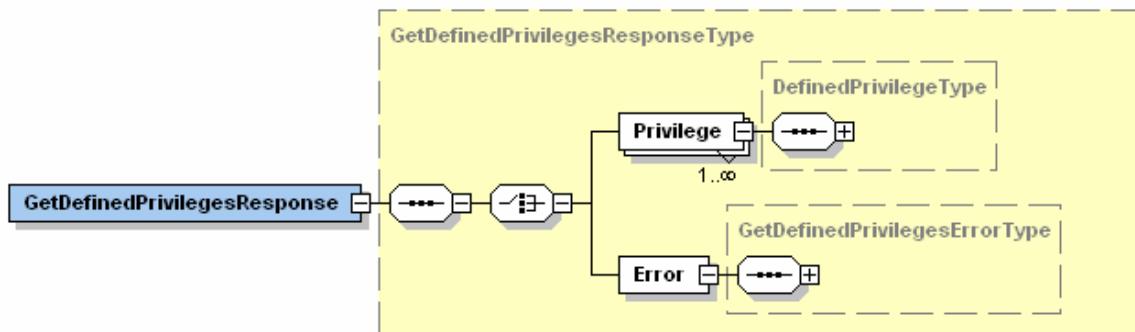


Figure 12
GetDefinedPrivilegesResponse XML Schema Element

8.2.9.1.4 *DefinedPrivilege Class*

8.2.9.1.4.1 The DefinedPrivilege class is mapped to the XML DefinedPrivilegesType. Table 17 shows how the attributes and associations of DefinedPrivilege class are mapped to XML. Figure 13 shows the XML schema diagram for the type.

Table 17 Translation Table for DefinedPrivilege Class

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
privilegeId	Text	Element	PrivilegeID: xsd:string
description	Text	Element	Description: xsd:string

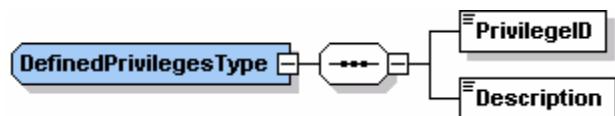


Figure 13
DefinedPrivilegesType XML Schema Type

8.2.9.1.5 *GetDefinedPrivileges Errors*

8.2.9.1.5.1 Errors for this operation are mapped to the XML Schema element Error of type GetDefinedPrivilegesErrorType. Figure 14 shows the XML Schema diagram for the error type. As shown, any error returned must contain the common error element as described in ¶8.2.6.1. In addition, if the error is an Operation not authorized error, the UnauthorizedOperationError must also be included providing additional context information as defined in SEMI E132. The mapping for the UnauthorizedOperationError class is as described in ¶8.2.6.2.

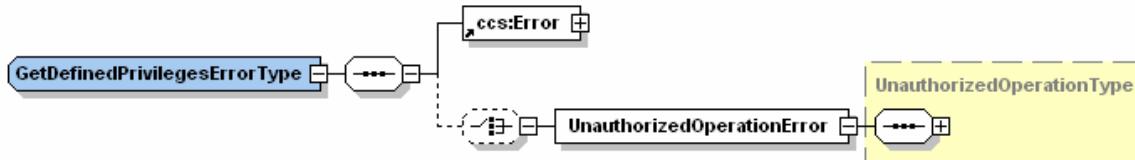


Figure 14
GetDefinedPrivilegesErrorType XML Schema Type

8.2.9.2 WSDL Message(s)

8.2.9.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “GetDefinedPrivilegesRequest Message”, and “GetDefinedPrivilegesResponseMessage” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 18.

Table 18 GetDefinedPrivileges Messages

<i>Message Name → Schema Element Name</i>	E132HeaderMessage → auth:E132Header GetDefinedPrivilegesRequestMessage → auth:GetDefinedPrivilegesRequest GetDefinedPrivilegesResponseMessage → auth:GetDefinedPrivilegesResponse
---	---

8.2.9.3 WSDL Operation

8.2.9.3.1 See the portType operation definition for “GetDefinedPrivileges” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 19.

Table 19 GetDefinedPrivileges PortType Operation

<i>Input Message Name</i>	GetDefinedPrivilegesRequestMessage
<i>Output Message Name</i>	GetDefinedPrivilegesResponseMessage

8.2.9.4 WSDL Operation Binding

8.2.9.4.1 See the operation binding for “GetDefinedPrivileges” in E132-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 20.

Table 20 GetDefinedPrivileges Operation Binding

<i>SOAPAction</i>	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:GetDefinedPrivileges
<i>Input Headers (WSDL Message, Required)</i>	E132HeaderMessage, required
<i>Output Headers (WSDL Message, Required)</i>	E132HeaderMessage, required

8.2.10 GetACL Operation

8.2.10.1 XML Schema Types

8.2.10.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the GetACL operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.2.10.1.2 GetACL Request

8.2.10.1.2.1 The service request is mapped to the XML Schema element GetACLRequest which is of type GetACLRequestType. There are no input parameters for this operation, thus the XML type is just an empty construct.

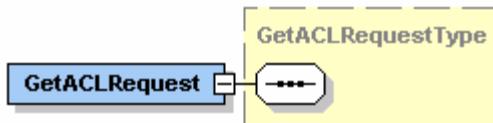


Figure 15
GetACLRequest XML Schema Element

8.2.10.1.3 *GetACL Response*

8.2.10.1.3.1 The service response is mapped to the XML Schema element GetACLResponse of type GetACLResponseType. Table 21 shows how the output parameters are mapped to XML. The XML Schema diagram for GetACLResponse is shown in Figure 16. The ACL is defined as a sequence of one or more elements of type ACLEntrySelectorType corresponding to the SEMI E132 UML class ACLEntry. Mapping for ACLEntry class is as described ¶8.2.10.1.4. Errors for this operation are returned using the XML Schema element Error of type GetACLErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 21 Translation Table for GetACL Output Parameters

Argument Name	Format	XML Element or Attribute	XML Name/Type
ACL	Unordered list	Element	ACL: ACLEntrySelectorType (one or more)
error	Structured data	Element	Error: GetACLErrorType

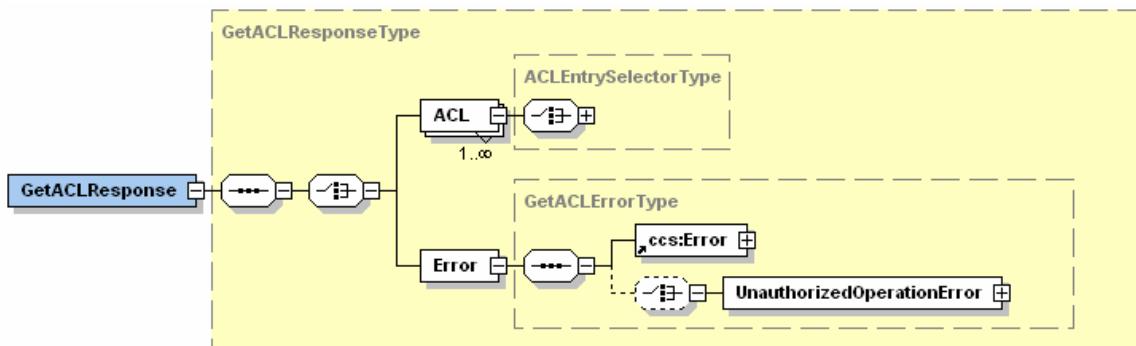


Figure 16
GetACLResponse XML Schema Element

8.2.10.1.4 *ACLEntry Class*

8.2.10.1.4.1 The ACLEntry class is mapped to the XML Schema type ACLEntryType. ACLEntry is an abstract base class and there are no attributes or roles defined for this base class, the XML Schema type for ACLEntry is just an empty sequence. Since there are only two derived classes of ACLEntry defined in SEMI E132, and this specification prohibits unrecognized extensions of the class, an ACLEntrySelector XML Schema type is defined which uses choice to select between the XML Schema types for the only 2 derived classes, PrivilegeAssignment and RoleAssignment. The XML Schema diagram for ACLEntrySelectorType is shown in Figure 17. Schema type mappings for PrivilegeAssignment and RoleAssignment classes are further described in subsequent sections.

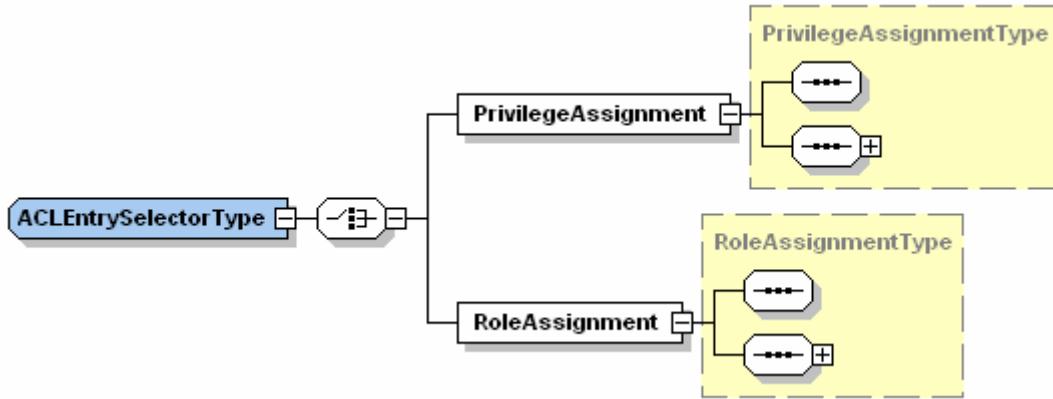


Figure 17
ACLEntrySelectorType XML Schema Type

8.2.10.1.5 *PrivilegeAssignment Class*

8.2.10.1.5.1 The *PrivilegeAssignment* class is mapped to the XML *PrivilegeAssignmentType*. Table 22 shows how the attributes and associations of *PrivilegeAssignment* class are mapped to XML. As with the XML definition for the *ACLEntry* class, the XML *SubjectSelectorType* provides a choice between the XML types of the only 2 allowed derived classes of the *Subject* class, *Principal* and *Role*. The XML mapping of *Principal* and *Role* classes are described further by the *RoleAssignment* class. The XML Schema diagram for *PrivilegeAssignmentType* is shown in Figure 18. *PrivilegeAssignmentType* extends *ACLEntryType*.

Table 22 Translation Table for *PrivilegeAssignment Class*

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
subject	Association	Element	Subject: SubjectSelectorType
privileges	Association	Element	Privilege: PrivilegeType (one or more)

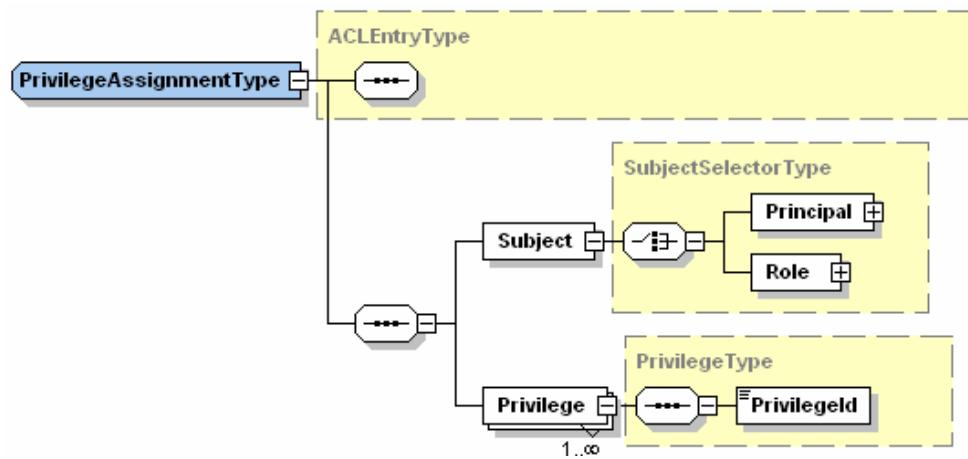


Figure 18
PrivilegeAssignment XML Schema Type

8.2.10.1.6 *RoleAssignment Class*

8.2.10.1.6.1 The *RoleAssignment* class is mapped to the XML *RoleAssignmentType*. Table 23 shows how the attributes and associations of *RoleAssignment* class are mapped to XML. The XML Schema diagram for *RoleAssignmentType* is shown in Figure 19. *RoleAssignmentType* extends *ACLEntryType*.

Table 23 Translation Table for RoleAssignment Class

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
principal	Association	Element	Principal: PrincipalType
Role	Association	Element	Role: RoleType

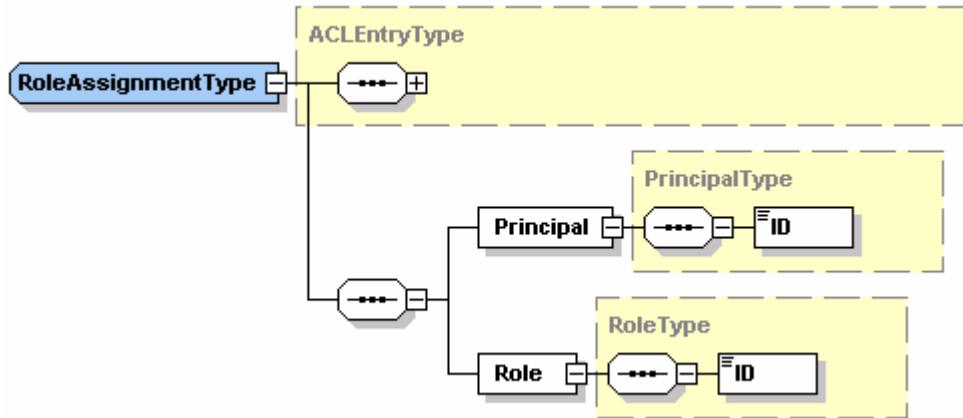


Figure 19
RoleAssignment XML Schema Type

8.2.10.1.7 Subject Class

8.2.10.1.7.1 The Subject Class is another abstract base class and is mapped to the XML SubjectType.

Table 24 Translation Table for Subject Class

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
id	Text	Element	ID: xsd:string

8.2.10.1.8 Role Class

8.2.10.1.8.1 The Role class is mapped to the XML RoleType. RoleType extends SubjectType, with no additional attributes or roles defined.

8.2.10.1.9 Principal Class

8.2.10.1.9.1 The Principal class is mapped to the XML PrincipalType. PrincipalType extends SubjectType, with no additional attributes or roles defined.

8.2.10.1.10 Privilege Class

8.2.10.1.10.1 The Privilege class is mapped to the XML PrivilegesType.

Table 25 Translation Table for Privilege Class

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
privilegeId	Text	Element	PrivilegeId: xsd:string

8.2.10.2 WSDL Message(s)

8.2.10.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “GetACLRequestMessage”, and “GetACLResponseMessage” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 26.

Table 26 GetACL Messages

<i>Message Name → Schema Element Name</i>	E132HeaderMessage → auth:E132Header GetACLRequestMessage → auth:GetACLRequest GetACLResponseMessage → auth:GetACLResponse
---	---

8.2.10.3 WSDL Operation

8.2.10.3.1 See the portType operation definition for “GetACL” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 27.

Table 27 GetACL PortType Operation

<i>Input Message Name</i>	GetACLRequestMessage
<i>Output Message Name</i>	GetACLResponseMessage

8.2.10.4 WSDL Operation Binding

8.2.10.4.1 See the operation binding for “GetACL” in E132-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 28.

Table 28 GetACL Operation Binding

<i>SOAPAction</i>	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:GetACL
<i>Input Headers (WSDL Message, Required)</i>	E132HeaderMessage, required
<i>Output Headers (WSDL Message, Required)</i>	E132HeaderMessage, required

8.2.11 AddACLEntry Operation

8.2.11.1 XML Schema Types

8.2.11.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the AddACLEntry operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.2.11.1.2 AddACLEntry Request

8.2.11.1.2.1 The service request is mapped to the XML schema element AddACLEntryRequest which is of type AddACLEntryRequestType. Table 29 shows how the input parameters are mapped to XML. The XML ACLEntryType is as described in ¶8.2.10.1.4, under Schema Types for GetACL operation.

Table 29 Translation Table for AddACLEntry Input Parameters

<i>Argument Name</i>	<i>Format</i>	<i>XML Element or Attribute</i>	<i>XML Name/Type</i>
aclEntry	Structured type	Element	ACLEntry: ACLEntryType

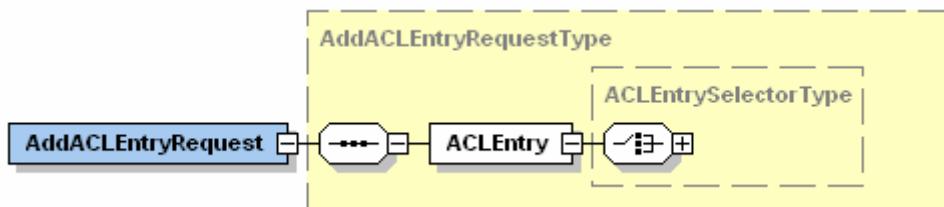


Figure 20
AddACLEntryRequest XML Schema Element

8.2.11.1.3 AddACLEntry Response

8.2.11.1.3.1 The service response is mapped to the XML Schema element AddACLEntryResponse of type AddACLEntryResponseType. Table 30 shows how the output parameters are mapped to XML. The XML Schema diagram for AddACLEntryResponse is shown in Figure 21. Errors for this operation are returned using the XML Schema element Error of type AddACLEntryErrorType, containing the common error element described in ¶8.2.6.1. In addition, if the error is an operation not authorized error, the UnauthorizedOperationError element must also be provided, of type UnauthorizedOperationType corresponding to the UnauthorizedOperationError class defined in SEMI E132, UML mapping is as described in ¶8.2.6.2. If the error is an unrecognized privilege error, the UnrecognizedPrivilegeError element must also be provided, of type UnrecognizedPrivilegeType corresponding to the UnrecognizedPrivilegeError class, UML mapping for this class is described in ¶8.2.11.1.4.

Table 30 Translation Table for AddACLEntry Output Parameters

Argument Name	Format	XML Element or Attribute	XML Name/Type
error	Structured data	Element	Error: AddACLEntryErrorType

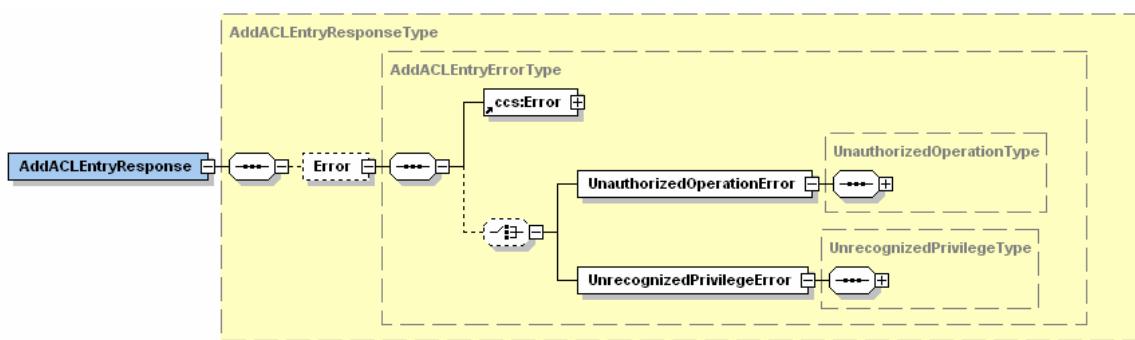


Figure 21
AddACLEntryResponse XML Schema Element

8.2.11.1.4 UnrecognizedPrivilegeError Class

8.2.11.1.4.1 The UnrecognizedPrivilegeError class is mapped to the XML UnrecognizedPrivilegeType. Table 31 shows how the attributes and associations of UnrecognizedPrivilegeError class are mapped to XML. The privileges provided represent the list of privileges in input arguments that are not recognized by the equipment. Figure 22 shows the XML Schema diagram.

Table 31 Translation Table for UnrecognizedPrivilegeError Class

Attribute or Role Name	UML Name/Type	XML Element or Attribute	XML Name/Type
description	Text	Element	Description: xsd:string
unrecognizedPrivileges	Association	Element	UnrecognizedPrivilege: PrivilegeType (one or more)

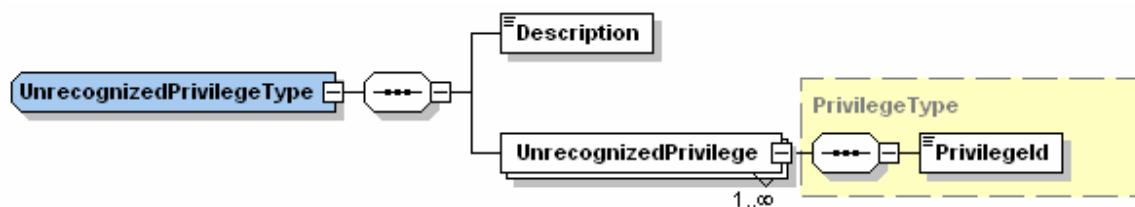


Figure 22
UnrecognizedPrivilegeType XML Schema Type

8.2.11.2 WSDL Message(s)

8.2.11.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “AddACLEntryRequestMessage”, and “AddACLEntryResponseMessage” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 32.

Table 32 AddACLEntry Messages

<i>Message Name → Schema Element Name</i>	E132HeaderMessage → auth:E132Header AddACLEntryRequestMessage → auth:AddACLEntryRequest AddACLEntryResponseMessage → auth:AddACLEntryResponse
---	---

8.2.11.3 WSDL Operation

8.2.11.3.1 See the portType operation definition for “AddACLEntry” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 33.

Table 33 AddACLEntry PortType Operation

<i>Input Message Name</i>	AddACLEntryRequestMessage
<i>Output Message Name</i>	AddACLEntryResponseMessage

8.2.11.4 WSDL Operation Binding

8.2.11.4.1 See the operation binding for “AddACLEntry” in E132-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 34.

Table 34 AddACLEntry Operation Binding

<i>SOAPAction</i>	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:AddACLEntry
<i>Input Headers (WSDL Message, Required)</i>	E132HeaderMessage, required
<i>Output Headers (WSDL Message, Required)</i>	E132HeaderMessage, required

8.2.12 DeleteACLEntry Operation

8.2.12.1 XML Schema Types

8.2.12.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the DeleteACLEntry operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.2.12.1.2 DeleteACLEntry Request

8.2.12.1.2.1 The service request is mapped to the XML Schema element DeleteACLEntryRequest which is of type DeleteACLEntryRequestType. Table 35 shows how the input parameters are mapped to XML.

Table 35 Translation Table for DeleteACLEntry Input Parameters

<i>Argument Name</i>	<i>Format</i>	<i>XML Element or Attribute</i>	<i>XML Name/Type</i>
subjectId	Structured type	Element	SubjectID: xsd:string

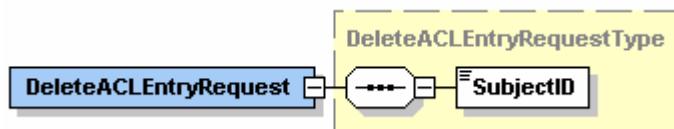


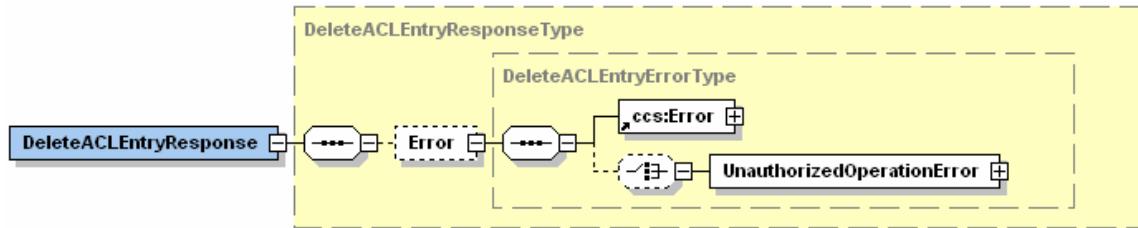
Figure 23
DeleteACLEntryRequest XML Schema Element

8.2.12.1.3 DeleteACLEntry Response

8.2.12.1.3.1 The service response is mapped to the XML Schema element DeleteACLEntryResponse of type DeleteACLEntryResponse. Table 36 shows how the output parameters are mapped to XML. The XML Schema diagram for DeleteACLEntryResponse is shown in Figure 24. Errors for this operation are returned using the XML Schema element Error of type GetACLErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 36 Translation Table for DeleteACLEntry Output Parameters

Argument Name	Format	XML Element or Attribute	XML Name/Type
error	Structured data	Element	Error: DeleteACLEntryErrorType



**Figure 24
DeleteACLEntryResponse XML Schema Element**

8.2.12.2 WSDL Message(s)

8.2.12.2.1 Three messages are defined for this operation corresponding to the E132Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “DeleteACLEntryRequestMessage”, and “DeleteACLEntryResponseMessage” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 37.

Table 37 DeleteACLEntry Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header DeleteACLEntryRequestMessage → auth: DeleteACLEntryRequest DeleteACLEntryResponseMessage → auth: DeleteACLEntryResponse
------------------------------------	---

8.2.12.3 WSDL Operation

8.2.12.3.1 See the portType operation definition for “DeleteACLEntry” in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 38.

Table 38 DeleteACLEntry PortType Operation

Input Message Name	AddACLEntryRequestMessage
Output Message Name	AddACLEntryResponseMessage

8.2.12.4 WSDL Operation Binding

8.2.12.4.1 See the operation binding for “DeleteACLEntry” in E132-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 39.

Table 39 DeleteACLEntry Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:DeleteACLEntry
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required