MEF Standard
MEF 118

Zero Trust Framework for MEF Services

October 2022

Disclaimer

The information in this publication is freely available for reproduction and use by any recipient and is believed to be accurate as of its publication date. Such information is subject to change without notice and MEF Forum (MEF) is not responsible for any errors. MEF does not assume responsibility to update or correct any information in this publication. No representation or warranty, expressed or implied, is made by MEF concerning the completeness, accuracy, or applicability of any information contained herein and no liability of any kind shall be assumed by MEF as a result of reliance upon such information.

The information contained herein is intended to be used without modification by the recipient or user of this document. MEF is not responsible or liable for any modifications to this document made by any other party.

The receipt or any use of this document or its contents does not in any way create, by implication or otherwise:

a) any express or implied license or right to or under any patent, copyright, trademark or trade secret rights held or claimed by any MEF member which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor
b) any warranty or representation that any MEF members will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor
c) any form of relationship between any MEF member and the recipient or user of this document.

Implementation or use of specific MEF standards, specifications, or recommendations will be voluntary, and no Member shall be obliged to implement them by virtue of participation in MEF Forum. MEF is a non-profit international organization to enable the development and worldwide adoption of agile, assured and orchestrated network services. MEF does not, expressly or otherwise, endorse or promote any specific products or services.

Table of Contents

List of Figures

List of Tables

1 List of Contributing Members

The following members of the MEF participated in the development of this document and have requested to be included in this list.

- Ciena
- CMC Networks
- Futurewei
- Strata Identity
- Versa Networks

2 Abstract

The increased usage of cloud services, mobile devices, work from home employees, shadow IT, and the Internet of Things (IoT) has dissolved traditional network perimeters. Services must evolve to provide secure User, Device and Application access to the Subscriber's networked resources (referred to as Target Actors in this document) from any location including interactions with third-party organizations, e.g., business partners, contractors, etc.

A Zero Trust cybersecurity approach removes the assumption of trust from these Users, Applications, and Devices (referred to as Actors in this document). It focuses on accessing Target Actors in a secure and authorized manner enforcing rigorous access controls and continually inspecting, monitoring, and logging network activity from the different Actors. This requires data-level protections, a robust identity architecture, and strategic micro-segmentation to create granular trust zones around a Subscriber's digital resources.

Zero Trust evaluates access requests and network traffic behaviors in real time over the length of active Sessions while continually and consistently recalibrating Subject Actor access to Target Actors and associated Policy Actions.

In summary, this standard defines a framework and requirements addressing Identity and Authentication of the different Actors plus Policy-Based Access Control providing enforcement at Policy End Points. The goal of this Zero Trust Framework is for associated Identity, Authentication, Policy Management and Access Control processes to be continuously and properly constituted, protected, and free from vulnerabilities when implemented and deployed. This Zero Trust Framework also defines Service Attributes, which are agreed between Subscriber and Service Provider to enable Service Providers to implement and deliver a broad range of services that comply with Zero Trust principles. This Zero Trust Framework is not intended as a stand-alone, implementable entity, a Zero Trust service, or Zero Trust product.

3 Terminology and Abbreviations

This section defines the terms used in this document. In many cases, the normative definitions to terms are found in other documents. In these cases, the third column is used to provide the reference that is controlling, in other MEF or external documents.

Term	Definition	Reference
Access Control (AC)	A process that utilizes an Access Control Policy to determine whether a request should be Allowed or Blocked	This document
Access Control Policy	A Policy that defines which Subject Actor(s), under what circumstances, may access which information from which Target Actor	This document
Accounting	The set of processes that measure, analyze, and report on information for the purpose of making a decision related to Actors	This document
Actor	A User, Device or Application	This document
Actor ID	An Actor's unique identifier	This document
Allow	The Policy Action which permits one or more operations as specified by an Access Control Policy	This document
Application (App)	A software program including associated Application Programming Interfaces (APIs)	Adapted from NIST SP 800-37 [18]
Application Programming Interface (API)	A software computing interface that exposes Application functionality to external systems	This document
Attribute-based Access Control (ABAC)	An Access Control approach in which access is mediated based on attributes associated with Subject Actors and Target Actors	Adapted from NIST SP 800-162 [24]
Authentication	The process of verifying the Identity of a Subject Actor	Adapted from NIST SP 800-12 [17]
Authorization	The process that results in Allowing or Blocking a Subject Actor from accessing a Target Actor	This document
Block	The Policy Action which denies a one or more operations as specified by an Access Control Policy	This document
Business Awareness	The knowledge and understanding of business concerns and regulations that need to be considered for Access Control decisions	This document
Context	The measured and inferred knowledge that collectively describes the Environmental Conditions in which an Actor exists or has existed	Industry paper [29]
Context Awareness	The knowledge and understanding of how changes in Subject Actor needs, Environmental Conditions, and business goals affect decision-making	ETSI GS ENI 005 [15]
Credential	An object or data structure that authoritatively binds the Identity of an Actor	Adapted from [21] [29] NIST SP 800-63-3
Delegation	The process where an Originating Actor assigns some or all its privileges derived from Policies to one or more Recipient Actors	This document
Delegate	The Actor receiving the assignment of a set of privileges from another Actor that has the authority to assign said privileges and Policies for a given time period	This document
Device	A physical instance of electronic equipment that connects to a network	This document
Direct Delegation	The process where a Recipient Actor receives all privileges derived from Policies from an Originating Actor	This document
Discretionary Access Control (DAC)	A means of restricting access for all Subject Actors based on the access policies defined by the System Administrator of the Target Actor	Adapted from NIST SP 800-192 [26]

Term	Definition	Reference
Event	An asynchronous action or occurrence that initiates further analysis	This document
Identifier String	A string consisting of one or more UTF-8 characters in the range of 32–126 (0x20 to 0x7e).	This document
Identity	The set of characteristics by which a Subject or Target Actor is recognizable and that, within the scope of an IdP's responsibility, is sufficient to uniquely disambiguate an instance of that Actor from an instance of any other Actor	Adapted from NIST SP 800-161 [27]
Identity Management (IdM)	The process that manages the verification, validation, and issuance of Identities	Adapted from NIST SP 800-12 [17]
Identity Provider (IdP)	The organization that manages the Authentication Credentials of an Actor	Adapted from NIST SP 800-63-3 [21]
Indirect Delegation	The process that uses a dedicated mechanism to grant an Originating Actor's unique privileges derived from Policies to one or more Recipient Actors	This document
Intermediary	A component that between a Subject Actor and Target Actor that intercepts the request from the Subject Actor, and forwards the request to the Target Actor, or another Intermediary. It also intercepts the response from the Target Actor and forwards the request to the Subject Actor or another Intermediary	Adapted from NIST SP 800-95 [22]
Legal Entity	An individual, organization, or company that has rights and obligations	This document
Legal Entity Credential	The certificate of the registration authority or the company registration Credential of the certificate authority	This document
Lightweight Directory Access Protocol (LDAP)	The protocol that provides access to distributed directory services that act in accordance with X.500 data and service models	RFC 4511 [3]
Mandatory Access Control (MAC)	A means of restricting access to Target Actors based on the sensitivity of the information contained in the Target Actors and the formal Authorization of Subject Actors to access information of such sensitivity	Adapted from NIST SP 800-53 [19]
Message Authenticity	The outcome of the process that provides assurance of the integrity of messages, documents, or stored data	NIST SP 800-152 [23]
Multi-Factor Authentication (MFA)	An Authentication system that requires more than one distinct Authentication factor.	Adapted from NIST SP 800-63-3 [21]
Open Authorization Protocol (OAuth)	An open-standards protocol that enables secure Authorization in a simple and standard method from web, mobile and desktop Applications	RFC 5849 [5], RFC 6749 [6]
OpenID Connect (OpenIDC)	A simple Identity layer on top of the OAuth 2.0 protocol	OpenID Connect [10]
Originating Actor	The Actor assigning some or all its privileges derived from Policies to one or more Recipient Actors	This document
Policy	A set of rules used to manage and control the changing or maintaining of the state of one or more managed objects	MEF 95 [14]
Policy Action	Definition of what is to be done to enforce a Policy, when the conditions of the Policy are met	Adapted from RFC 3198 [2]
Policy Administration Point (PAP)	An entity that provides an interface for creating, managing, testing, and debugging Policies, and storing these Policies in an appropriate repository	Adapted from NIST SP 800-162 [24]
Policy-based Access Control (PBAC)	An Access Control method that uses Policies to determine the appropriate type of Access Control based on the Subject and Target Actors' Service Attributes and behavior, the current Situation, and applicable business requirements	This document
Policy Decision Point (PDP)	An entity that computes access decisions using a set of Evaluation Policies	Adapted from NIST SP 800-162 [24]

Term	Definition	Reference
Policy End Point (Policy EP)	A location where one or more Policy-related functions are placed	This document
Policy Enforcement Point (PEP)	An entity that implements Policy decisions that were made by the PDP	This document
Policy Information Point (PIP)	A repository of the data required for Policy evaluation needed by the PDP to make a decision	Adapted from NIST SP 800-162 [24]
Recipient Actor	The Actor receiving some or all its privileges derived from Policies from one or more Originating Actors	This document
Risk	A measure of the extent to which threats by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence.	Adapted from CNSSI No. 4009 [28]
Risk Awareness	The knowledge, understanding or recognition of the potential hazards that could be caused by or to a Subject or Target Actor that need to be considered for Access Control decisions	This document
Role	A set of responsibilities assigned to an Actor	Adapted from MEF 78.1 [13]
Role-based Access Control (RBAC)	A collection of access Authorizations a Subject or Target Actor receives based on a given set of Roles	Adapted from NIST SP 800-53 [19]
Role Object	A class instance that defines computing responsibilities using a set of attributes, methods, relationships, or constraints	Adapted from MEF 78.1 [13]
Security Assertion Markup Language (SAML)	The set of specifications describing security assertions that are encoded in XML, profiles for attaching the assertions to various protocols and frameworks, the request/response protocol used to obtain the assertions, and bindings of this protocol to various transfer protocols such as SOAP or HTTP	OASIS SAML [11]
Security Label	The means used to associate a set of security attributes with a specific information object as part of the data structure for that object	NIST SP 800-53 [19]
Separation of Duty (SoD)	The security principle that mandates that no User should be given enough privileges to be able to misuse or compromise a system on their own	Adapted from NIST SP 800-192 [26]
Service	A MEF-defined service	This document
Service Provider (SP)	An organization that provides Services to Subscribers. In this document, “Service Provider” refers to an organization that delivers services that comply with this Zero Trust Framework	This document
Session	An ephemeral flow of IP packets between a specific Subject and Target Actor pair	This document
Situation	The set of circumstances and conditions at a given time that may influence decision-making.	ETSI ENI [15]
Situation Awareness	The perception of data and behavior that pertain to the relevant circumstances or conditions of a system or process, the comprehension of the meaning and significance of these data and behaviors, and how processes, actions, and new situations inferred from these data and processes are likely to evolve in the near future	MEF W95 [14]
Subject Actor	An Actor requesting access to a Target Actor	This document
Subscriber	The organization that purchases or uses a Service that uses this Zero Trust Framework	This document
System Administrator	The User Actor, assigned the Role that is authorized to create and maintain Policies	This document
Target Actor	An Actor that a Subject Actor wants to access	This document

Term	Definition	Reference
Trust Domain	A security domain that implements a security Policy and is administered by a single authority	Adapted from CNSSI No. 4009 [28]
User	A person	This document
Single Sign-On (SSO)	A process that allows a User to be authenticated across multiple administrative domains with one set of login Credentials	This document
Uniform Resource Identifier (URI)	A federated and extensible naming system wherein each scheme's specification may further restrict the syntax and semantics of identifiers using that scheme	RFC 8820 [8]
Zero Trust Framework (ZTF)	A cybersecurity architecture where Subject Actors are authenticated, authorized, and continuously validated before being granted access to, or maintain access to, or perform operations on Target Actors	This document

Table 1 - Terminology and Abbreviations

4 Compliance Levels

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 (RFC 2119 [1], RFC 8174 [7]) when, and only when, they appear in all capitals, as shown here. All key words must be in bold text.

Items that are **REQUIRED** (contain the words **MUST** or **MUST NOT**) are labeled as **[Rx]** for required. Items that are **RECOMMENDED** (contain the words **SHOULD** or **SHOULD NOT**) are labeled as **[Dx]** for desirable. Items that are **OPTIONAL** (contain the words **MAY** or **OPTIONAL**) are labeled as **[Ox]** for optional**.**

A paragraph preceded by **[CRa]<** specifies a conditional mandatory requirement that **MUST** be followed if the condition(s) following the "<" have been met. For example, "**[CR1]<**[D38]" indicates that Conditional Mandatory Requirement 1 must be followed if Desirable Requirement 38 has been met. A paragraph preceded by **[CDb]<** specifies a Conditional Desirable Requirement that **SHOULD** be followed if the condition(s) following the "<" have been met. A paragraph preceded by **[COc]<** specifies a Conditional Optional Requirement that **MAY** be followed if the condition(s) following the "<" have been met.

5 Introduction

A Zero Trust Framework (ZTF) is a cybersecurity architecture where Subject Actors are authenticated, authorized, and continuously validated before being granted access to, or maintain access to, or perform operations on Target Actors. A Service Provider would use this ZTF in conjunction with a MEF Service, subsequently referred to as Service, that promises to ensure that the Subscriber's Subject Actors cannot access Target Actors unless explicitly permitted by the Subscriber. In this ZTF, an Identity Provider (IdP) is used to identify and authenticate Subject and Target Actors. However, Actor identification may vary depending upon the Service capabilities. For example, an IP Service may only be able to identify Devices based on their IP address and not individual Users or Applications. In this case, the Service would use the ZTF to protect Devices and not Users or Applications. How Users, Devices, and Applications are identified would be determined by the Service using this ZTF and thus their identification is beyond the scope of this document.

A Service complying with this ZTF must provide "least privilege" access by Subject Actors only to Target Actors for which the Subject Actor is authorized to access and nothing more. The Service using this ZTF continuously monitors, evaluates their access privileges based on their behavior, and authenticates Users, Devices and Application Subject Actors that require access to Target Actors and provides access only to Target Actors that the authenticated Subject Actor requires with the minimum privilege necessary.

Cloud adoption and digital transformation are happening at an accelerated pace and organizations have been forced to rethink their security architecture accordingly. Organizations now have more Users, Devices (including IoT Devices) and Applications distributed across their network than ever before, which in turn makes their current centralized security architecture less effective. With many people working from home, often on personal (not corporate controlled) Devices, the attack surface for organizations has dramatically increased and the security posture of such Devices has decreased.

The goal of this Zero Trust Framework is for associated Identity, Authentication, Policy Management and Access Control processes to be continuously and properly constituted, protected, and free from vulnerabilities when implemented and deployed.

The standard is organized as follows:

- Section 6: Overview of how MEF Services are defined using Service Attributes and how they relate to this ZTF standard
- Section 7: Summary of components of a ZTF, further described throughout the document
- Section 8: Identity Management, Identity Management protocols, and Identity Provider definitions, requirements, and Service Attributes
- Section 9: Actor types, definitions, requirements, and Service Attributes for Users, Devices, and Applications
- Section 10: Actor relationships among Identity, Roles, Context, and Capabilities
- Section 11: Concepts of Roles, principle of least privilege, and principal of Separation of Duty
- Section 12: Definitions and requirements for Access Control methods (MAC, DAC, RBAC, and ABAC)
- Section 13: Definition of Policy functional entities, Policy Actions, and Policy End Points
- Section 14: Policy-Based Access Control (PBAC) definition and requirements
- Section 15: Concepts and requirements for Actor Delegation and revocation

- Section 16: Lifecycle management of Zero Trust Policies based on Situation, Risk, and Business Awareness
- Section 17: Continuous Monitoring Method Service Attribute definition, methods, and requirements
- Section 18: References and Resources
- Appendix A: Example use cases for Policy End Point placement
- Appendix B: Use Case Example for Actor Delegation (Informative)

6 MEF Services using this Zero Trust Framework

MEF Services are defined using Service Attributes. A Service Attribute captures specific information that describes some aspect of the Service behavior. Service Attributes are agreed between the Service Provider, an organization that provides Services to Subscribers, and a Subscriber, which is the organization that purchases or uses a Service that uses this Zero Trust Framework. How such an agreement is reached, and the specific values agreed upon, might have an impact on the price of the Service or on other business or commercial aspects of the relationship between the Subscriber and the Service Provider; this is outside the scope of this document. Some examples of how agreement could be reached are:

- The Service Provider mandates a particular value
- The Subscriber selects from a set of options specified by the Service Provider
- The Subscriber requests a particular value, and the Service Provider accepts it
- The Subscriber and the Service Provider negotiate to reach a mutually acceptable value

Service Attributes describe the externally visible behavior of the Service as experienced by the Subscriber as well as the rules and Policies associated with a MEF Service. However, they do not constrain how the Service is implemented by the Service Provider. The initial value for each Service Attribute is agreed upon by the Subscriber and the Service Provider in advance of the Service deployment. At any time, the Subscriber and the Service Provider may subsequently agree on changes to the values of certain Service Attributes. This document does not constrain how such agreement is reached, e.g., a Service Provider may allow the Subscriber to select an initial value from a pre-determined set of values or allow them to change their selection at any time during the lifetime of the service.

This standard defines the components of a ZTF, their Service Attributes, and associated requirements that would be used when creating a broad set of Services, e.g., Secure Access Service Edge (SASE) services, Security Service Edge (SSE) services, Software-defined Wide Area Networking (SD-WAN) services, cybersecurity services, etc., that incorporate Zero Trust. This ZTF does not define a Service but defines the requisite properties and associated Service Attributes for Service Providers to use when creating such Services to deliver to their Subscribers in order to comply with this ZTF.

7 Zero Trust Framework

A Zero Trust Framework consists of the following key elements, as illustrated in Figure 1. Note that the figure provides a conceptual, macro-level view of some of the interactions between the key elements. These ZTF elements would be applied to a MEF service.

- Subject Actor
- Target Actor
- Identity Management
- Access Control
- Policy Management
- Policy End Points
- Continuous Monitoring

The Service using this ZTF uses Policies for both Subject and Target Actors to determine if the Subject Actor is permitted to access the Target Actor and if the Target Actor is willing to accept access from the Subject Actor. In a ZTF, all interactions between a Subject Actor and Target Actor are assumed to be malicious. Therefore, all interactions are Blocked, unless a Policy authorizes the interaction. Furthermore, if no Policy exists for a Subject Actor, then the Actor's access will be Blocked. The Service enforces Policies at the Policy End Points. The Service Provider places Policy End Points, where appropriate for a given Service. For example, they could be placed within the Subject Actor Device, the Target Actor (when under control of the Subscriber), or on another networking Device such as an SD-WAN Edge (MEF 70.1 [12]), Ethernet switch, or Wi-Fi access point. Use case examples of Policy End Point placement are described in section Appendix A:

Subject Actors can perform operations on a set of Target Actors if Allowed by an Access Control Policy defined by the Subscriber and permitted by the Service. An Access Control Policy is a Policy that defines which Subject Actor(s), under what circumstances, may access which information from which Target Actor. An Access Control Policy encompasses the human-readable Policy in addition to the machine-readable Policy assertions. Identity Management (IdM) may be provided by one or more IdM systems that may be provided by the Subscriber or a third-party Identity Provider (IdP). The IdM systems manage the Actor's and its Delegate's identities. Policies use Policy-based Access Control (PBAC) that may select different Access Control (AC) mechanisms (described in section 12.1) while performing Authentication, Authorization, Accounting, and auditing (AAAA) of the Actor and its Delegates. Access Control is a process that utilizes an Access Control Policy to determine whether a request should be Allowed or Blocked. To simplify notation, this document uses the term Actor to refer to both an Actor and its Delegate, except where differences need to be discussed for each.

The Service continuously monitors the authenticated Actors for compliance with their Policies. When found to be non-compliant, the Service reevaluates Policies for Actors that may result in a different Policy Action, e.g., Block Subject Actor from accessing a Target Actor. The Service reevaluates a Policy based on different types of triggers, including time-based, e.g., automatically Block a Subject Actor after 30 minutes, event-based, e.g., a Target Actor has been found to be compromised by malware so Block the Subject Actor from accessing the Target Actor, or data-driven, e.g., through machine learning, an anomalous pattern of behavior has been determined for the Subject Actor and thus the Subject Actor is Blocked from accessing the Target

Actor. Throughout this document, the term "Policy" refers to a Policy that uses PBAC unless explicitly stated otherwise.

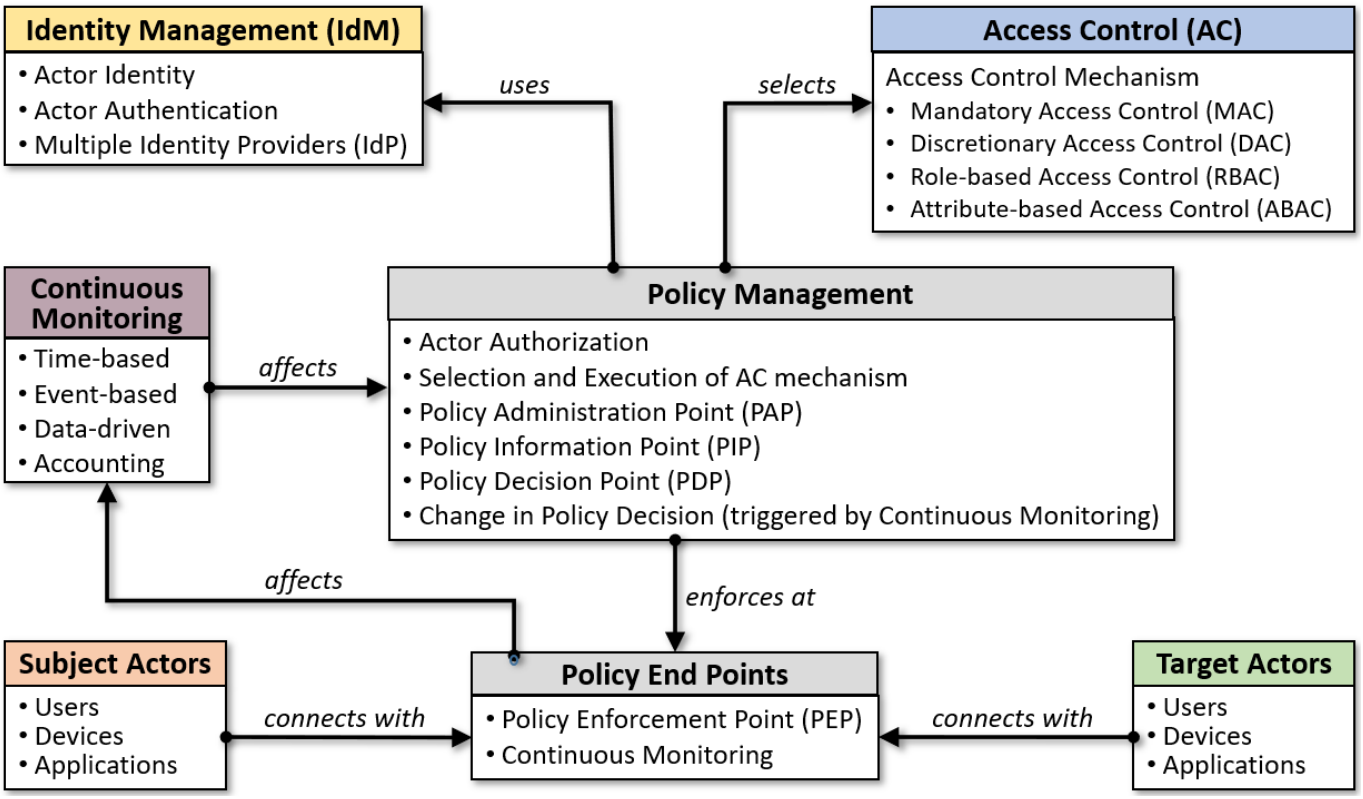

Figure 1 - Key Elements of a Zero Trust Framework

The Zero Trust Framework Service Attributes defined in this document can be used by network services, cybersecurity services, or a collection of services, e.g., Secure Access Service Edge (SASE) services.

[R1] To comply with this ZTF, the Service Provider **MUST** encrypt all Service Attribute parameters and their respective values defined in this standard, both at rest and in transit, using common cryptographic suites.

For example, once a User logs into a portal that uses encrypted Credentials, they can then view and make changes, if permitted, to the Service Attribute parameter values. When the User saves their changes or reviews the information, the information is again encrypted. The method provided by the Service Provider to modify and review encrypted Service Attribute parameters and values is beyond the scope of this document.

[D1] This standard **RECOMMENDS** that the cryptographic suites in [R1] use NIST cryptographic algorithms defined in NIST SP 800-175B [25]

Per [R1], the expectation is that encryption is applied all the time. However, some decryption and subsequent re-encryption may occur in certain "trusted" places where the information must be processed. Such places may be inside a Service Provider secure data center where the information may need to be processed by other security functions, e.g., to identify any malware threats, malicious URLs or domain names, or protocol exploits.

8 Identity Management

Identity Management (IdM) plays a key role in a Zero Trust Framework. IdM is used by a Service that adopts this Zero Trust Framework to identify and Authenticate the Subject Actor (and possibly the Target Actor) for which Policies are applied.

The Identity Provider (IdP) is the organization or system used to establish the Identity of an Actor for a MEF Service. The Zero Trust framework described in this document is defined to support any type of IdP. IdM, delivered by the IdP, may be provided by the Subscriber, by a third-party IdP, or by the Service Provider through which the Subscriber has contracted for their IdM services. IdM protocols and their requirements are discussed in section 8.3. There are two Identity Service Attributes defined in this section: Identity Provider Service Attribute and Identity Provider Subscriber Service Attribute. These two Service Attributes are necessary to provide IdM for a Service using this ZTF.

The Service Provider uses the IdPID and IdPSubscriberID parameters (as described in sections 8.1 and 8.2, respectively) provided by the Subscriber to associate the Subscriber's identity with a given IdP. For example, Subscribers commonly use an IdP for all their employees to authenticate their desktop or laptop computers before providing initial access to the corporate network. This commonly manifests itself via a login screen that appears after the computer powers up. The Subscriber's User then enters their login Credentials, e.g., username and password at a minimum, to get authenticated. This Authentication is performed by the IdP. An IdP in the context of this ZTF standard for a Service is the same IdP used by the Subscriber as described in the aforementioned example.

8.1 Identity Provider Service Attribute

A Service Provider that complies with this Zero Trust Framework needs to connect to an IdP to obtain information about an Actor's Identity and to obtain Actor Authentication. To connect to an IdP, the Subscriber needs to provide the identity of the IdP they use to the Service Provider when the Service Provider is not the IdP.

The Identity Provider Service Attribute is used to identify an IdP. The Identity Provider Service Attribute has parameters with example values as described in Table 2 for the IdP used with a Service. Note that a MEF Service could define additional parameters and values for this Service Attribute beyond those defined in Table 2.

Note that the Subscriber provides the IdPID parameter value (in Table 2) from the Identity Provider Service Attribute to the Service Provider if the Service Provider is not the IdP.

This document specifies requirements for three parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
IdPID	Unique identifier of the Identity Provider for a given Subscriber	https://subscribername.myidpname.com
IdPCommonName	Human-readable name of the Subscriber's Identity Provider associated with the IdPID	IdP Corporation
IdPAuthProtocol	Protocol used to authenticate the Actors	OAuth 2.0, SAML 2.0

Table 2 - Identity Provider Service Attribute Parameters

The IdPID is the URI [8] assigned by the IdP that uniquely identifies the IdP for a given Subscriber, e.g., https://subscribername.myidpname.com.

[R2] Each IdP **MUST** use an IdPID value that is unique among all Subscribers to whom the IdP is providing Identity Management services

[R3] If the IdP is provided by the Subscriber, the IdPID **MUST** be reachable via its IP address

[D2] If the IdP is provided by the Subscriber, the IdPID **SHOULD** be reachable via its domain name

[R4] If the IdP is external to the Subscriber, the IdPID **MUST** be reachable and publicly resolvable

Publicly resolvable means that the IdP must be reachable via a public IP address or internet domain name, e.g., myIdP.com.

The IdPCommonName is used to convey the name of the IdP, e.g., "IdP Corporation", associated with the IdPID, using a human-readable format.

[R5] The IdPCommonName **MUST** use an Identifier String that can consist of up to 128 characters

An Identifier String is a string consisting of one or more UTF-8 characters in the range of 32–126 (0x20 to 0x7e).

The IdPAuthProtocol conveys the Authentication protocol used by the IdP for identity management that the Subscriber (Subject Actor) needs to use. The Authentication protocol to be used is commonly negotiated between the Service Provider's identity management system software and a Subscriber's (Subject Actor) Application, e.g., web browser client or Zero Trust Network Access (ZTNA) client for a SASE service.

[R6] The Service Provider supporting this ZTF **MUST** support OAuth 2.0 as defined in RFC 6749 [6]

[D3] The Service Provider supporting this ZTF **SHOULD** support SAML 2.0 as defined by OASIS [11]

[O1] The Service Provider supporting this ZTF **MAY** support other identity management protocols in addition to OAuth 2.0 and SAML 2.0.

Refer to section 8.3 for more details on OAuth and SAML Authentication protocols.

An example of the Identity Provider Service Attribute is described below using a JSON (JavaScript Object Notation) format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"IdentityProviderServiceAttribute":[
  { "IdPID":"https://MyCommpany.MyFavoriteIdP.com" },
  { "IdPCommonName":"MyFavoriteIdP" },
  { "IdPAuthProtocol":"OAuth 2.0" }
]}
```

8.2 Identity Provider Subscriber Service Attribute

The Identity Provider Subscriber Service Attribute is used to identify the Subscriber. The Identity Provider Subscriber Service Attribute has parameters with example values as described in Table 3.

For a Service Provider to deliver a Service using this ZTF, when the Service Provider is not the IdP, the Subscriber needs to provide to the Service Provider, their Credential method for their IdP account for Subscriber identification of all Subscriber Actors.

This document specifies requirements for three parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
IdPSubscriberID	Unique identifier of the Subscriber within the Identity Provider	123456-fe93
IdPSubscriberCommonName	Human-readable name for the Subscriber associated with the IdPSubscriberID	ACME Anvil Corporation
IdPSubscriberCredentialsType	Type of Credential used by the Subscriber to log into the specific IdP	username/password, bearer token, certificate

Table 3 - Identity Provider Subscriber Service Attribute Parameters

The IdPSubscriberID is the unique identifier, e.g., 123456-fe93, to be processed by Applications using a machine-readable format. Note that the IdPSubscriberID could be created from the hash of the validated Credentials.

[R7] The IdPSubscriberID value **MUST** be unique across all Subscribers who use Identity Management services from a given IdP, identified via their IdPID

The IdPSubscriberCommonName is used to convey the name of the Subscriber, e.g., "ACME Anvil Corporation", associated with the IdPSubscriberID, using a human readable format.

[R8] The IdPSubscriberCommonName value **MUST** be either of the following:

- *Null* (meaning that no IdPSubscriberCommonName value is provided)
- String of the name of the IdP Subscriber

The IdPSubscriberCredentialsType conveys the type of Credential used by the Subscriber to log into a specific IdP, e.g., username/password or bearer token.

An example of the Identity Provider Subscriber Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"IdentityProviderSubscriberServiceAttribute":[
  { "IdPSubscriberID":"123456-fe93},
  { "IdPSubscriberCommonName":"ACME Corporation" },
  { "IdPSubscriberCredentialsType":"bearer token" }
]}
```

8.3 Identity Management Protocols

There are different identity management (IdM) protocols that can be used to identify and authorize Actors. This document only specifies requirements for Open Authentication (OAuth) and Security Assertion Markup Language (SAML) - two of the most popular IdM protocols.

8.3.1 Open Authentication/Authorization (OAuth) Protocol

OAuth 2.0, defined in RFC 6749 [6], is a protocol that provides token-based Authentication and Authorization for Internet Applications. OAuth eliminates the need for Users to provide passwords to each third-party Application by acting as an intermediary between Users and the Applications that require access to the third-party web Applications. It is the underlying technology used for website Authentication that

allows Users to consolidate their Credentials and streamline their login process. OAuth communication is secured using the Transport Layer Security (TLS) protocols used by the Hypertext Transfer Protocol Secure (HTTPS) protocol and access tokens issued to client Applications (Subject Actors) have a limited validity period which is typically per Session. A Session is an ephemeral flow of IP packets between a specific Subject and Target Actor pair.

[R9] When the Subscriber and Service Provider agree to use OAuth as the means of Authentication, then OAuth 2.0 **MUST** be used

[CR1]< [R9] When OAuth 2.0 is used, the minimum version of TLS **MUST** be TLS 1.2 as defined in RFC 5246 [4]

A typical example is a Service Provider offering a SASE Service using Zero Trust Network Access which uses the OAuth protocol to authenticate a User Subject Actor, running a SASE client, to access the Subscriber's private network over the Internet.

8.3.2 OpenID Connect (OpenIDC) Protocol

OpenID Connect [10] is an open standard Authentication protocol that provides the identity layer on top of the OAuth 2.0. OpenID Connect enables clients to verify the Identity of the User based on the Authentication performed by an Authorization Server, as well as to obtain basic profile information about the User in an interoperable and REST-like manner. OpenID Connect allows websites, referred to as "Relying Parties", to authenticate Users using a third-party service instead of providing their own Authentication systems. OpenID Connect caters client systems of all types. The specification is extensible and allows for participants to use optional features such as encryption of Identity data, discovery of OpenID Providers, and Session management.

[R10] When OpenID Connect is used, it **MUST** be used with OAuth 2.0

OpenID Connect is used to provide Authentication over and above the Authorization provided by OAuth. A typical use case would be when a User grants an Application access to the User's contacts list, e.g., a User grants Application X access to the User's email Application's contacts. This is similar to the example provided in section 8.3.1, however, the User Authentication is performed using OpenID Connect and an OpenID Provider before granting Authorization to the requested Target Actor.

8.3.3 Security Assertion Markup Language (SAML)

Security Assertion Markup Language or SAML [11] is an XML-based open standard that provides Authentication and Authorization for Users (Subject Actors) to gain access to Target Actors (referred to as resources in SAML standard). SAML functions through SAML assertions that facilitate the transfer of Identity data between two entities and forms the basis for most SSO (Single Sign-On) implementations. SAML was originally intended for managing access to Target Actors on the Internet exclusively but has been adopted by most enterprises instead to facilitate the SSO process. This is largely due to its thorough support of common Authentication protocols used by enterprises namely Kerberos, LDAP (Lightweight Directory Access Protocol), and RADIUS (Remote Authentication Dial-In User Service).

SAML consists of three main components:

- SAML service provider[1] – the entity that acts as the gatekeeper of the Target Actor that a User wishes to access. It refers all access requests to a specific identity provider to authenticate Users and authorize access
- Identity Provider – a trusted entity that will determine whether the User is who they are claiming to be (Authentication) and establishes which Target Actor the User (Subject Actor) will be permitted to access (Authorization). The Identity Provider will send the SAML assertion necessary to the SAML service provider for Users to access the requested Target Actor
- User agent – the entity, i.e., the User requesting access to Target Actors controlled by the SAML service provider

A typical SAML example would be when a User Actor requests access to Application Target Actor using the User's Credentials. In this use case, the SAML service provider providing the SSO function will redirect the User to the IdP to authenticate the User. Once authenticated, the SAML service provider will return the required information for the User to access the Target Application. Note that a Service Provider delivering a MEF Service using this ZTF may provide the SSO function or may rely on a third-party to provide the SSO function to authenticate the User (Subject Actor) prior to the Service Provider authorizing access to the Target Actor based on Access Control Policies.

[R11] When the Service Provider is the IdP and SAML is used, SAML 2.0 **MUST** be used as the protocol for Authentication and Authorization

[1] Note that the term "service provider" is based on terminology used by the SAML standard [11] and is not to be confused with the MEF Service Provider definition in this document, the latter of which is referred to using upper case text.

9 Actors

An Actor is a User, Device, or Application that either wants to access another Actor or allow themselves to be accessed. An Actor requesting access to another Actor is defined as a Subject Actor, or simply Subject. An Actor that a Subject Actor wants to access is defined as a Target Actor, or simply Target. Actors are continuously evaluated against Policies that will either Allow or Block the Subject Actor from accessing some or all Target Actors. Allow is the Policy Action which permits one or more operations as specified by an Access Control Policy. Block is the Policy Action which denies one or more operations as specified by an Access Control Policy.

A User is defined as a person. A Device is a physical instance of electronic equipment that connects to a network. Devices provide compute, memory, storage, and networking capabilities. Devices can be either purpose-built, e.g., IoT temperature sensor, or a general purpose, e.g., laptop, smartphone, or server, that can run multiple Applications. An Application is defined as a software program. An Application runs on a single physical Device. Complex Applications can be distributed and operate on multiple physical Devices. A complex Application could consist of multiple virtual machines, containers, or microservices. In this case, each virtual machine, container or microservice could be considered an Application Actor for which Policies can be applied. Finally, Application Actors also include their respective Application Programming Interfaces (APIs) since they are a part of the Application. An API is a software computing interface that exposes Application functionality to external systems.

A User can use a Device. A User can also use an Application that runs on a Device. An Application can run autonomously on a Device without any User intervention. Figure 2 illustrates the relationship between the different types of Actors for which Policies are applied. Some examples will help clarify the different permutations listed in Figure 2.

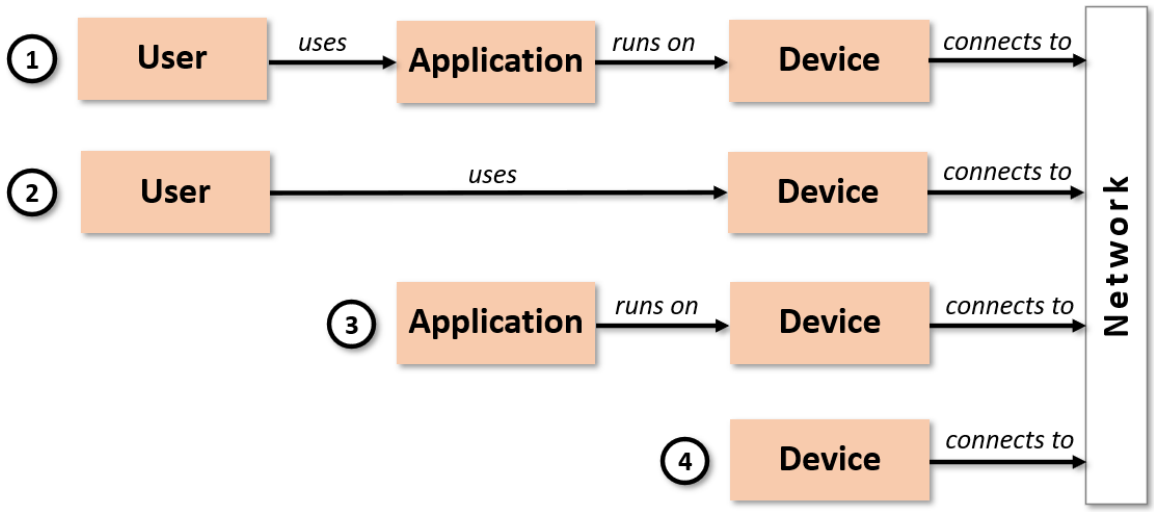

Figure 2 - Relationship between different Actors

In scenario 1, a User, i.e., a person, could be using an Application, e.g., a web browser, that is running on a Device, e.g., laptop, that connects to the network. This enables different Polices to be applied to the User, Application, and Device.

In scenario 2, a User could be using a Device, e.g., a heart rate monitor, that connects to the network. While the heart rate monitoring functionality on the Device could be considered an Application running on a

Device as in scenario 1, it is simpler to apply a single Policy to the Device since it only runs a single Application. This enables different Polices to be applied to the User and Device.

In scenario 3, an Application may be operating autonomously without requiring a User to use it, e.g., a video recording Application, which runs on a Device, e.g., video camera mounted on a traffic signal pole, that connects to the network. The camera Device may also have an audio Application running on it which records sounds. Unique Policies can be applied to the Device, video recording Application, and audio recording Application. This enables different Polices to be applied to the Application and Device.

In scenario 4, a Device may be a temperature sensor operating in a freezer case at a grocery store. While the temperature sensor could be considered an Application running on the Device, it is simpler to apply a single Policy to the Device since it only runs a single Application. This enables Polices to be applied only to the Device.

This Zero Trust Framework defines Actors and associated Service Attributes used to create Policy criteria. Each Actor can be either a Subject Actor or Target Actor for a given Session. Each Actor has different Service Attributes used to ascertain the Actor's Identity, Role, or Capabilities for which Policies are applied. In this Zero Trust Framework, each Subject Actor's Identity is retrieved from, or validated against, one or more IdPs before any Access Control Policies can be applied to the Actor. A Subscriber will assign at least one Role to each Actor as defined by the User, Device, and Application Role Service Attributes in sections 9.1.2, 9.2.2, and 9.3.2. For example, a User may have the Role of "employee" and "IT administrator". The "employee" Role could be defined to enable the User access to corporate employee IT Applications such as email, conferencing, and file servers. The "IT administrator" Role could be defined to enable the User access to the IT systems used by the company that is restricted to those who work in the IT department. Finally, Capabilities refers to inherent or advertised functionality of a Device or Application or the permitted Capabilities of an Application when the Application's Capabilities are programmable and thus can vary. Each Device will have inherent or advertised, i.e., 'built-in', Capabilities as defined by the Device Capabilities Service Attribute in section 9.2.3. Application Capabilities are not defined.

[R12] A Subject Actor authenticated by the IdP, **MUST** have at least one Role agreed for it as defined in the User Roles Service Attribute (refer to section 9.1.2), Device Roles Service Attribute (refer to section 9.2.2), and Application Roles Service Attribute (refer to section 9.3.2)

Note that a Subscriber may not have any control over a Target Actor, e.g., public Internet web site, and thus be unable to assign a Role to it.

Figure 3 illustrates how Actors are modeled for how Identity, Roles, and Capabilities relate to the different type of Actors for which Policies are applied.

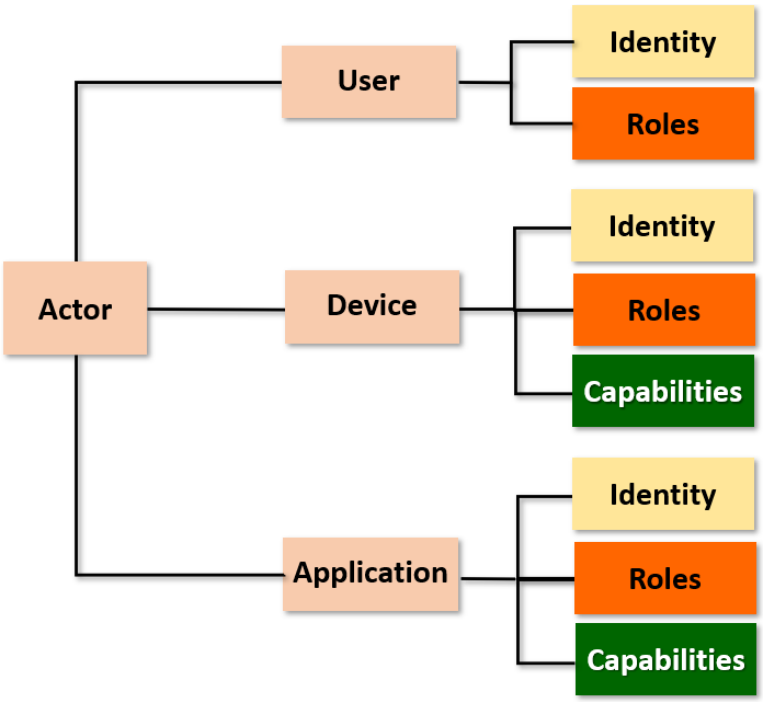

Figure 3 - Identity, Role, and Capabilities for different types of Actors

[R13] The Service Provider **MUST** ensure that an Actor has a unique identifier within the context of a Service

An Actor's unique identifier is an Actor ID. An Actor ID generalizes the reference to identifiers for User Actors (UserID, defined in section 9.1.1), Device Actors (DevID, defined in section 9.2.1), and Application Actors (AppID, defined in section 9.3.1).

[R14] Any Service utilizing Delegation **MUST** enforce that an Actor ID be associated with a set of public keys

[R14] ensures an Actor can prove that it controls and thus can authenticate the utilized unique identifier without a verifying 3rd party, such as an IdP.

[D4] Any Service utilizing Delegation **SHOULD** enforce that an Actor ID follow the W3C DID Core specification [30]

[D4] supports the self-issuance of unique identifiers that allow for cryptographically verifiable non-repudiation. Note that the usage of commonly used public key infrastructure (PKI) based on ITU-T X.509 [9] digital certificates is permissible.

[R15] Any Service utilizing Delegation **MUST** enforce that an Actor ID be linked to a Legal Entity of the Actor

[R16] Any Service utilizing Delegation **MUST** enforce that the Actor has a Legal Entity Credential

[R17] Any Service utilizing Delegation **MUST** enforce that the Legal Entity Credential be cryptographically signed, based on the public keys associated with the unique identifier of the Credential issuer or IdP

[R18] Any Service utilizing Delegation **MUST** enforce that the Legal Entity Credential is cryptographically verifiable

[R19] Any Service utilizing Delegation **MUST** enforce that the Legal Entity Credential is cryptographically revocable

This document makes no assumptions as to how a Credential to establish a Legal Entity is created, which IdPs are mutually acceptable between Subscriber and Service Provider, and how these identity Credentials are exchanged to establish mutual trust across Trust Domains.

Note that Credentials may be self-issued. The acceptance of self-issued Credentials is up to the Subscriber and Service Provider that need to rely on the claim(s) within a self-issued Credential.

[R20] A Service utilizing Delegation **MUST** ensure that the issuer of the Legal Entity Credential has a Credential linking the unique identifier of the issuer to an entity accepted by the Service Provider and Subscriber

When the Legal Entity Credential, e.g., a company registration in a country or an ITU X.509 certificate from a known Certificate Authority, of a Subject Actor is presented to a Target Actor for the first time, it should be accompanied by a Legal Entity Credential of the issuer of the Credential of the Subject Actor. The Legal Entity Credential is the certificate of the registration authority or the company registration Credential of the certificate authority. This enables the Target Actor to independently verify both certificates and establish a trust relationship with the Subject Actor using the Legal Entity Credential issuer as the root of trust.

[D5] Any Service utilizing Delegation **SHOULD** enforce that the Credential follows W3C - Verifiable Credential Data Model v1.1 [31]

[R21] Any Service utilizing Delegation **MUST** enforce that a Credential has a unique identifier

Resolvable in this context means that the unique identifier is a URI (RFC 8820 [8]) that resolves either to a registry of the issuer or a data store where the original Credential is stored.

Note, that the unique and resolvable identifier of a Credential does not have to be associated with any cryptographic keys.

[R22] Any Service utilizing Delegation **MUST** enforce that if present, the status of a Credential be discoverable by a party verifying the Credential, called the Credential verifier

In the context of this document, a Credential status signals if a Credential has been revoked or not. In the context of this document, a Credential verifier is defined per W3C - Verifiable Credential Data Model v1.1.

[D6] Any Service utilizing Delegation **SHOULD** enforce that a Credential be discoverable by a Subject and Target Actor

[R23] Any Service utilizing Delegation **MUST** enforce that the presentation of a Credential be cryptographically signed by the presenter of the Credential, also known as the Credential holder

See the W3C Verifiable Credential Standard for a definition of Credential holder.

[R24] Any Service utilizing Delegation **MUST** enforce that a Credential holder has a unique identifier that has been established within the ZTF security context the holder operates

[R24] places a requirement on any Credential holder to have a unique identifier that has been issued within a ZTF, e.g., a DID method found to be acceptable for a given security Context, which is entirely controlled by the DID controller, and nothing else. This unique identifier and controlling keys must be independently verified as being valid by any 3rd party without the active participation of the Credential holder.

[R24] also means that when a Credential is presented by the Credential holder, the Credential verifier can independently verify not only the conformance and content of a Credential but also whether the Credential holder is being spoofed by a 3rd party or not.

9.1 User Actor Service Attributes

A User is a person who interacts with Applications that operate on Devices or interacts with a Device as illustrated in scenarios 1 and 2 in Figure 2. The User Identity Service Attribute uniquely identifies a User associated with the Subscriber (see section 9.1.1) and the User Role Service Attribute provides the list of Roles assigned for a given User (see section 9.1.2). Since there will be more than one User, the Service would specify the User Service Attributes for all Users of the Service. A Service Attribute that captures this list of Users and their Roles would be defined by the Service and utilize the Parameters specified and defined in section 9.1.

9.1.1 User Identity Service Attribute

The User Identity Service Attribute identifies the User (Subject Actor). The User Identity Service Attribute has parameters and example values as described in Table 4. Note that a MEF Service could define additional parameters and values for this Service Attribute beyond those defined in Table 4. Note that User Identity is authenticated by the IdP or IdM system used by the Subscriber.

This document specifies requirements for four parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
UserID	Unique string for the User	52688-2e93
UserCommonName	Human-readable name of the User, associated with the UserID	John Smith
UserName	Human-readable identity of the User associated with the UserID	jsmith
UserCredentialType	Type of identity Credentials of the User requesting Authentication	sha3{concat(UserName+password)}, biometric identifier

Table 4 - User Identity Service Attribute Parameters

The UserID is the unique string for the User, e.g., 52688-2e93, to be processed by Applications using a machine-readable format.

[R25] Each UserID **MUST** be a string

[R26] Each UserID **MUST** be unique for a given Service

The UserCommonName is used to convey the name of the User, e.g., “John Smith”, associated with the UserID, using a human-readable format.

[R27] The UserCommonName **MUST** be assigned one of the following values:

- *Null* (meaning that no UserCommonName value is provided)
- String of the name of the User

[R28] When the UserCommonName value is not *Null*, the UserCommonName value **MUST** be unique among all UserCommonNames a Subscriber uses

The UserName is used to convey the identity Credential of the User, e.g., "jsmith", associated with the UserID, using a human-readable format. All Users of a given Service must have unique UserNames.

[R29] The UserName value associated with a UserID **MUST** be unique among all other Users of a given Service

The UserCredentialType conveys the type of identity Credentials of the User requesting Authentication, e.g., SHA3-256 hash of "jsmith, password23" combination or a biometric identifier such as facial, fingerprint or palmprint recognition.

An example of the User Identity Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"UserIdentityServiceAttribute":[
  { "UserID":"52688-2e93" },
  { "UserCommonName":"John Smith" },
  { "UserName":"jsmith" },
  { "UserCredentialType":" certificate" }
]}
```

9.1.2 User Roles Service Attribute

The User Roles Service Attribute defines the Roles that can be assigned to a User (Subject Actor) that are utilized by the Policies of a Service incorporating this Zero Trust Framework. Policies utilize each User Role to determine which Device or Application Actors the User is permitted to access.

[R30] The Service Provider **MUST** ensure that each User Actor is assigned at least one Role

The User Roles Service Attribute has parameters and example values as described in Table 4. Note that a MEF Service could define additional parameters and values for this Service Attribute beyond those defined in Table 5

This document specifies requirements for two parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
UserRoleID	Unique identifier of the Role assigned to a User	52688-2e93
UserRoleCommonName	Human-readable name associated with the UserRoleID	Administrator, Management, Technical, Contributor, Owner, Contractor, Employee

Table 5 - User Roles Service Attribute Parameters for each User Role in the list

The UserRoleID provides the unique identifier of the Role assigned to a User, e.g., 52688-2e93, to be processed by Applications using a unique machine-readable format.

[R31] Each UserRoleID **MUST** be use a unique value for a given Subscriber

The UserRoleCommonName is used to convey the name of the Role assigned to the User, e.g., Owner, associated with the UserRoleID, using a human-readable format.

[R32] The UserRoleCommonName **MUST** use one of the following values:

- *Null* (meaning that no UserRoleCommonName value is provided)
- String of the name of the User Role

[R31] and [R32] mean that each UserRoleID/UserRoleCommonName must be unique for a given Subscriber organization. For example, a Subscriber defines a UserRoleCommonName "Employee" with UserRoleID 2e93-95e4. The Subscriber would use that User Role throughout the Subscriber organization and thus define it once.

The Service Provider assigns a special System Administrator Role one or more Users enabling them to create and maintain Policies for a Service. Because this Role has much authority, it should be assigned to only a few Users at most. The System Administrator Role can be performed by the Service Provider or a third-party, e.g., consultant, approved by the Subscriber. For example, assigning a Service Provider or third-party the Role of System Administrator would be useful for a small business that does not have IT personnel available to perform this function. The System Administrator Role could also be performed by the Subscriber which is commonly the case for a larger organization with an IT department.

[R33] Actors that do not have the Role of System Administrator **MUST NOT** be authorized to create, modify, or delete Policies

[R34] A Service Provider that adopts a ZTF **MUST** ensure that at least one User Actor has the Role of System Administrator

[R35] The Service Provider and Subscriber **MUST** agree which Actor or Actors are assigned the Role of System Administrator

An example of the User Roles Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
UserRolesServiceAttribute = {
  "UserRole1" : { "UserRoleID":"52688-2e93","UserRoleCommonName":"Employee" },
  "UserRole2" : { "UserRoleID":"52688-4393","UserRoleCommonName":"Contractor" },
  "UserRole3" : { "UserRoleID":"52688-2293","UserRoleCommonName":"Management" }
}
```

9.2 Device Actor Service Attributes

A Device is a server, laptop, desktop, smartphone, tablet, IoT device, etc. Devices send and receive IP packets to and from the network. Devices may be driven by a User, e.g., laptop or smartphone, or may operate autonomously, e.g., an IoT camera or temperature sensor. Service Attributes for a Device uniquely identify the Device to determine if the Device (Subject Actor) is compliant with Access Control Policies that will either Allow or Block the Device from accessing some or all Target Actors.

The Device Identity Service Attribute uniquely identifies a Device associated with the Subscriber (see section 9.2.1), the Device Role Service Attribute provides the list of Roles assigned for a given Device (see section 9.2.2), and the Device Capabilities Service Attribute provides the list of Capabilities assigned for a given Device (see section 9.2.3). Since there will be more than one Device, the Service would specify the Device Service Attributes for all Devices of the Service. A Service Attribute that captures this list of Devices, their Roles, and their Capabilities would be defined by the Service and utilize the Parameters specified and defined in section 9.2.

9.2.1 Device Identity Service Attribute

The Device Identity Service Attribute contains information about a Device used to identify it. The Device Identity Service Attribute has parameters and example values as described in Table 6. Also note that Device Identity can be provided by the Subscriber's IdM system or an IdP used by the Subscriber.

This document specifies requirements for two parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
DevID	Unique identifier for a Device	95e4-e9ac221
DevCommonName	Human-readable name of the Device	HP-DESKTOP-IT-001, IPHONE-USA-GA-ATL-USER-001

Table 6 - Device Identity Service Attribute Parameters

The DevID provides the unique identifier for the Device, e.g., 95e4-e9ac221, to be processed by Applications using a machine-readable format of a unique Identifier String. A common form of DevID is a serial number or other unique identifier readable from the Device.

[R36] The DevID **MUST** be unique among all DevIDs used by the Subscriber in order to uniquely identify each Device

If the DevID value cannot be retrieved from the Device, the Subscriber and Service Provider can agree on a DevID value for the Device.

The DevCommonName is used to convey the name of the Device, e.g., HP-DESKTOP-IT-001, associated with the DevID, using a human-readable format.

[R37] The Subscriber **MUST** assign a DevCommonName using one of the following values:

- *Null* (meaning that no DevCommonName value is provided)
- String of the name of the Device

An example of the Device Identity Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"DeviceIdentityServiceAttribute":[
  { "DevID":"95e4-e9ac221" },
  { "DevCommonName":"HP-DESKTOP-IT-001" }
]}
```

9.2.2 Device Roles Service Attribute

The Device Roles Service Attribute is the list of Roles assigned to a Device (Subject Actor) that can be utilized by Policies of a Service incorporating this Zero Trust Framework. Policies utilize each Device Role to determine which Device or Applications Actors the Device is permitted to access.

[R38] The Service Provider **MUST** assign at least one Role to each Device Actor

The Device Roles Service Attribute has parameters and example values as described in Table 7.

This document specifies requirements for two parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
DevRoleID	Unique identifier of the Role assigned to a Device associated with the DevRoleCommonName	52678-2e73
DevRoleCommonName	Human-readable name of the Role assigned to the Device	guest laptop, lab server

Table 7 - Device Role Service Attribute Parameters for each User Role in the list

The DevRoleID provides the unique identifier for the Role assigned to the Device, e.g., 52678-2e73, to be processed by Applications using a machine-readable format.

[R39] A DevRoleID **MUST** use a unique identifier for each Role for a Device

The DevRoleCommonName is used to convey the name of the Role of the Device, e.g., "guest laptop", associated with the DevRoleID, using a human-readable format.

[R40] The DevRoleCommonName **MUST** use one of the following values:

- *Null* (meaning that no DevRoleCommonName value is provided)
- String of the name of the Device Role

An example of the Device Roles Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
DeviceRolesServiceAttribute = {
  "DevRole1" : { "DevRoleID":"52678-2e73","DevRoleCommonName":"GuestLaptop" },
  "DevRole2" : { "DevRoleID":"52678-4373","DevRoleCommonName":"LabServer" },
  "DevRole3" : { "DevRoleID":"52678-2273","DevRoleCommonName":"CorporateLaptop" }
}
```

9.2.3 Device Capabilities Service Attribute

The Device Capabilities Service Attribute provides information about a Device to identify its capabilities. The Device Capabilities Service Attribute has parameters and example values as described in Table 8. These are the minimum amount that should be supported. A Device may have more capabilities.

This document specifies requirements for three parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
DevType	Human-readable name that describes the type of Device	camera, temperature sensor, laptop, smartphone, IoT Device
DevInterfaceTypeList	List of types of network interfaces on the Device	Ethernet, Wi-Fi, 3G, LTE, 5G, Bluetooth
DevOS	Operating system running on the Device	Windows, macOS, Linux, Android, iOS

Table 8 - Device Capabilities Service Attribute Parameters

The DevType is used to convey the type of Device, e.g., laptop, associated with the DevID, using a human-readable format. The DevType conveys the primary purpose of the device. For example, a laptop Device may have a camera in it. In this example, DevType would be defined as a laptop, not a camera since the primary purpose of the device is a laptop and not a camera.

[R41] A Device **MUST** be assigned only one DevType value

DevInterfaceTypeList is a non-empty list of the types of network interfaces that a Subject Device can use to reach a Target Actor, e.g., "LTE", using a human-readable format.

[R42] The DevInterfaceTypeList value **MUST** be a non-empty list

DevOS conveys the operating system running on the Device and can be used to determine whether a supported operating system is being used on a Device or whether other policies specific to a specific operating system can be applied.

[R43] DevOS **MUST** use one of the following values:

- *Null* (meaning the operating system is not recognized or no DevOS value is provided)
- String of the name of the recognized operating system

[R44] When the value of DevOS is not *Null*, the possible value of DevOS **MUST** include:

- Windows
- macOS
- Linux
- Android
- iOS

[D7] The DevOS value **SHOULD** allow for an operating system specified by the Subscriber that is not listed in [R44]

An example of the Device Capabilities Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"DeviceCapabilitiesServiceAttribute":[
  { "DevType":"Desktop PC" },
  { "DevInterfaceTypeList":"Wi-Fi" },
  { "DevOS":"Windows" }
]}
```

9.3 Application Actor Service Attributes

An Application is a software program that runs on a Device or in a cloud computing environment. Example Applications are software programs used by Users, e.g., email client software, and those that operate autonomously on Devices, e.g., SQL database, microservices, virtual machines, containers, etc. Applications may be driven by a User or may operate autonomously and communicate to other Actors via APIs. Service Attributes for an Application uniquely identify the Application to determine if the Application (Subject Actor) is compliant with Access Control Policies that will either Allow or Block the Application (Subject Actor) from accessing some or all Target Actors.

The Application Identity Service Attribute uniquely identifies an Application associated with the Subscriber (see section 9.3.1) and the Application Role Service Attribute provides the list of Roles assigned for a given Application (see section 9.3.2). Since there will be more than one Application, the Service would specify the Application Service Attributes for all Applications of the Service. A Service Attribute that captures this list of Applications and their Roles would be defined by the Service and utilize the Parameters specified and defined in section 9.3.

A key intention of Zero Trust and therefore Zero Trust-enabled Services is the focus on preventing unauthorized data breaches and data exfiltration that can be introduced intentionally or inadvertently by Applications.

In perimeter-less cloud ecosystems most interactions are between Applications (between Subject and Target Actors) either operating within the same or connected cloud, edge compute or other hosted

environment. Policies governing Subject and Target Application Actors are essential to ensure authorized delivery of requested data based on the Application Identity and Application Role Service Attributes in sections 9.3.1 and 9.3.2.

9.3.1 Application Identity Service Attribute

The Application Identity Service Attribute contains information that can be read from an Application or its APIs that is used to identify the Application (Subject or Target Actor). The Application Identity Service Attribute has parameters and example values as described in Table 9.

This document specifies requirements for four parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
AppID	Unique identifier of the Application	e9ac222
AppCommonName	Human-readable name of the Application associated with the AppID	SAP, Zoom, YouTube, Office365
AppVersion	Version number of the Application	6.2, v12, 2.14.22
AppType	Human-readable name of the type of Application	Accounting, Productivity, Database

Table 9 - Application Identity Service Attribute Parameters

The AppID provides the unique identifier for the Application, e.g., e9ac222, to be processed by Applications using a machine-readable format of a unique identifier string. AppID can be used for Authentication purposes. AppID can also be used to apply Policies for a given Session between a Subject Actor and Target Actor.

[R45] An AppID **MUST** use a unique value among all Applications for each type of Application to uniquely identify each Application within a given Service

The AppCommonName is used to convey the name of the Application, e.g., Office365™, associated with the AppID, using a human-readable format.

[R46] The AppCommonName **MUST** use one of the following values:

- *Null* (meaning that no AppCommonName value is provided)
- String of the name of the Application

The AppVersion is used to convey the version number of Application, e.g., 6.2, associated with the AppID, using a human-readable format.

[R47] An AppVersion **MUST** use one of the following values:

- *Null* (meaning that no AppVersion value is provided or can be queried)
- String of the version number of the Application

The AppType is used to convey the type of Application, e.g., Productivity, associated with the AppID, using a human-readable format.

[R48] An AppType **MUST** use one of the following values:

- String of the type of Application

An example of the Application Identity Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"ApplicationIdentityServiceAttribute":[
  { "AppID":"e9ac222" },
  { "AppCommonName":"Office365" },
  { "AppVersion":"6.2" },
  { "AppType":"Productivity" }
]}
```

9.3.2 Application Roles Service Attribute

The Application Roles Service Attribute is the list of Roles assigned to an Application (Subject or Target Actor) that can be utilized by Policies of a Service incorporating this Zero Trust Framework. Policies utilize each Application Role to determine which Device or Applications Actors the Application is permitted to access. An Application must have at least one Role assigned.

[R49] The Service Provider **MUST** ensure that each Application Actor is assigned at least one Role

The Application Roles Service Attribute has parameters and example values as described in Table 10.

This document specifies requirements for two parameters and associated values for this Service Attribute and does not preclude additional parameters and associated values.

Parameters	Summary Description	Example Values
AppRoleID	Unique identifier of the Role assigned to an Application	52478-2e23
AppRoleCommonName	Human-readable name of the Role assigned to the Application	Lab Application

Table 10 - Application Role Service Attribute Parameters

The AppRoleID provides the unique identifier for Role assigned to the Application, e.g., 52688-2e93, to be processed by Applications using a machine-readable format.

[R50] Each AppRoleID **MUST** use a unique identifier for each Role for an Application

The AppRoleCommonName is used to convey the name of the Role of the Application, e.g., "Lab Application", associated with the AppRoleID, using a human-readable format.

[R51] The AppRoleCommonName **MUST** use one of the following values:

- *Null* (meaning that no AppRoleCommonName value is provided)
- String of the name of the Application Role

[R52] An Application **MUST** have at least one Role assigned in the Application Roles Service Attribute

An example of the Application Roles Service Attribute is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
ApplicationRolesServiceAttribute = {
 "AppRole1" : { "AppRoleID":"52478-2e23","AppRoleCommonName":"LabApplication" },
 "AppRole2" : { "AppRoleID":"52478-4323","AppRoleCommonName":"FinanceApplication" },
 "AppRole3" : { "AppRoleID":"52478-2223","AppRoleCommonName":"EcommerceApplication" }
}
```

10 Actor Relationships among Identity, Roles, Context, and Capabilities

Zero Trust Policies are applied using a Subject and Target Actor Identifier, Role, and Capabilities (only applies to Devices and Applications) while factoring in the Context. Refer to section 16.1 for more details on Context. The Policy can be expressed in the 7-tuple, {*SubjectActorID, SubjectActorRoles, SubjectActorCapabilities, TargetActorID, TargetActorRoles, TargetActorCapabilities, Context*} where:

- *SubjectActorID* is the Identity of the Subject Actor
- *SubjectActorRoles* is the Role or Roles the Subject Actor is performing
- *SubjectActorCapabilities* are the Capabilities of the Subject Actor
- *TargetActorID* is the Identity of the Target Actor
- *TargetActorRoles* is the Role or Roles the Target Actor is performing
- *TargetActorCapabilities* are the Capabilities of the Target Actor
- *Context* is the Context when the Policy is applied

Figure 4 illustrates the relationships.

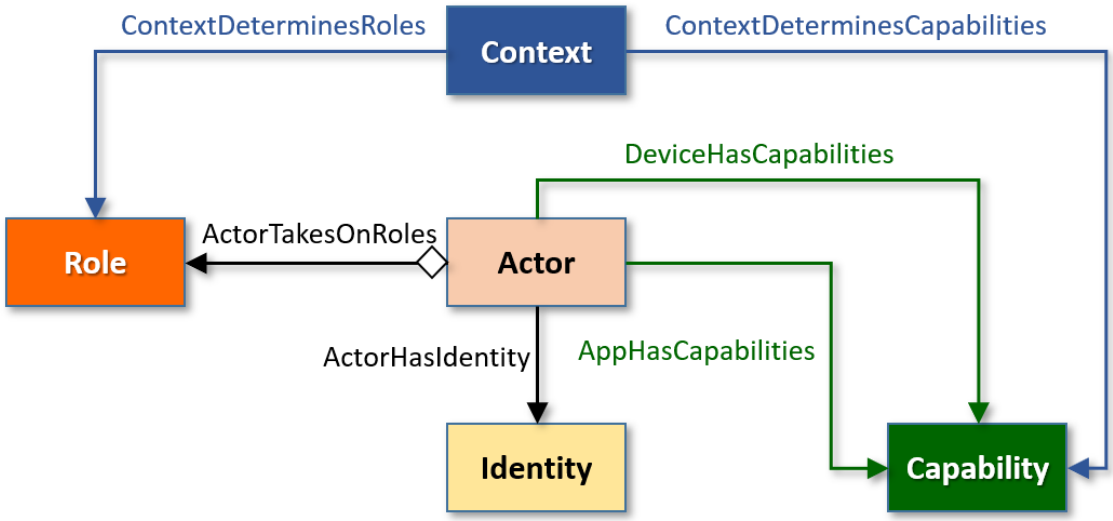

Figure 4 - Relationship between Actors, their Identity and Capabilities, Role and Context

Per the Actor Definitions in section 9, an Actor can be either a User, Device, or Application. Only Devices and Application Actors have capabilities as illustrated by the two green arrows with labels "DeviceHasCapabilities" and "AppHasCapabilities" in Figure 4. The blue arrow labeled "ContextDeterminesCapabilities" places restrictions on which Capabilities may be used for a given Actor for a particular Context.

Actors may have one or more Roles. Roles may be combined, e.g., a User Actor may concurrently have the "Employee", "Manager", and "SupportDeskLead" Roles. This is shown by the black arrow with a diamond on it (the diamond means "aggregate"). The blue arrow labeled "ContextDeterminesRoles" places restrictions on which Roles may be used by which Actor for a particular Context.

11 Roles, Least Privilege, and Separation of Duty

This section expands upon the concept of Actor Roles discussed earlier in this standard including the principal of least privilege and the Separation of Duty for different Actor Role assignments.

11.1 Roles, Role Activation, and Role Engineering

A Role (MEF 78.1 [13]), is assigned to an Actor within a particular Context. For the purposes of Access Control:

- Users are Subject Actors, both trusted and untrusted, who want to access a Device or Application Actor
- Subject Actors duties and responsibilities are dictated by the set of Roles associated with each Subject
- Operations are the set of functions a Subject Actor performs on a Target Actor
- Permissions are a set of ordered {Operation, Target} pairs that are used by Policies to Allow or Block a Subject Actor to perform an operation on a Target Actor

A Role Hierarchy defines a subordinate-superordinate relationship among Roles. This enables a subordinate Role to inherit the properties and behavior of its superordinate Role.

Role activation is a mechanism that constrains the assignment of a particular Role in each Context.

Role engineering is the design and management of a set of Roles based on the structure of the organization that uses Roles. Role mining is similar but extracts Roles from existing identity and other information.

11.2 Principle of Least Privilege

The principle of least privilege mandates that a given Subject Actor, or its Role(s), can only be granted the minimum privileges and Authorizations required to perform its function.

11.3 Principle of Separation of Duty

Separation of Duty (SoD) (NIST 800-192 [26]), is a security principle that mandates that no User should be given enough privileges to be able to misuse or compromise a system on their own. Hence, two or more Subject Actors or their Roles are required to complete a given task. There are different types of SoD approaches, such as sequential (one Subject Actor or their Roles followed by another, etc.), collaborative (two or more Subject Actors or their Roles working together in the same location, where each Subject Actor or their Roles has to agree), historical (the same Subject Actor or their Roles cannot assume a Role within some time period), multi-factor (two or more factors are used to separate the task), and geographical (two or more Subject Actors or their Roles must be in separate geographical locations). Some of these approaches may also be aggregated.

SoD properties may be static or dynamic.

11.3.1 Static SoD

Static SoD properties are based on Subject Actor Role assignment and are enforced at Role Authorization time. They are maintained using Role activation. They include Static Cardinality of Roles, Separation of Duty, and Operational Separation of Duty, as described below. They do not involve the Subject Actor of the Role, or mapping from the Subject Actor to other components.

Static Cardinality of Roles limits the number of Roles.

Static Separation of Duty (SSoD) splits responsibilities among multiple Roles. In effect, this makes each set of authorized Roles that enforces SSoD mutually exclusive, so a given Subject Actor can only assume one Role at a given time. Put another way, SSOD ensures that to have all permissions necessary to complete a task, a separate Role should be used for each step in the task.

Static Operational Separation of Duty (SOSoD) splits a given task into a set of operations and constrains all Roles. Put another way, each Role used to perform a step in a task must have mutually exclusive permissions.

11.3.2 Dynamic SoD

Dynamic SoD properties enhances SSOD by enabling the SOD constraint to be applied at Role activation time (instead of at Role Authorization time). Dynamic properties are typically used in conjunction with static properties to enable richer constraints on the activities that could occur when a Role is active.

Dynamic Cardinality restricts the number of Subject Actors that have a given Role at any one time.

Dynamic Separation of Duty (DSoD) defines a set of Roles as mutually exclusive of each other at Role activation. This ensures that at any one time, a Subject Actor may be active in only one of the set of Roles so designated. Note that extra care must be taken to ensure that DSoD Roles are not activated consecutively.

Dynamic Operational Separation of Duty (DOSoD) defines a group of permissions as mutually exclusive of one another with regards to the Roles activated by a Subject Actor on behalf of any single Subject Actor. Note that extra care must be taken to ensure that DOSoD Roles are not activated consecutively.

12 Access Control

Access Control (AC) defines which Subject Actors can perform which operations on a set of Target Actors according to a set of Policy criteria. AC building blocks include the following four items:

- Authentication
  - Authentication is the process of verifying the Identity of a Subject Actor
  - The three Authentication factors are 'something you know', 'something you have', and 'something you are'
  - Policies may be used to dictate the set of verification criteria used
- Authorization
  - Authorization is the process that results in Allowing or Blocking a Subject Actor from accessing a Target Actor
  - Policies may be used to prescribe the criteria for the Authorization decision
  - Authentication and Authorization are pre-requisites to making an Access Control decision
- Accounting
  - Accounting is the set of processes that measure, analyze, and report on information for the purpose of making a decision related to Actors
  - Policies may be used to specify the accounting criteria
    - For example, accounting applied to Access Control is a systematic review and analysis of all the steps performed in granting or denying an Access Control decision
- Auditing
  - Auditing is the process of independently reviewing records and activities of an operation, e.g., Access Control
  - Policies may be used to specify the criteria for performing audit operations
  - This is done to ensure compliance with established operational procedures and Policies

12.1 Common Access Control Approaches

This section describes four established access control approaches in use today.

12.1.1 Mandatory Access Control (MAC)

Mandatory Access Control (MAC) [19], is an Access Control Policy that is enforced on all Subject Actors that interact with a given information system. The Policy evaluates a Security Label (NIST 800-53 [19]) of the information contained in the Target Actor to determine what operations the User Subject Actor is authorized to perform. The Operating System (or a Security Kernel) constrains the ability of a Subject Actor to perform operations on a Target Actor.

12.1.2 Discretionary Access Control (DAC)

Discretionary Access Control (DAC) [26], is an Access Control Policy that is enforced on all Subject Actors that interact with a given information system. The Policy compares a Security Label of the information contained in the Target Actor to the Authorization of the Actor, and only grants the requested operation if the two are equal. In this approach, Subject Actors are not constrained by the action they take with information for which they have been granted access.

Note that MAC restricts DAC. Therefore, when using MAC, one must be aware that MAC Policies can override DAC Policies.

12.1.3 Role-based Access Control (RBAC)

Role-based Access Control (RBAC), defined in NIST 800-53 [19] is a family of RBAC Policies that is enforced only for User (User Subject Actor) that interact with a given information system. This ZTF standard expands NIST's RBAC concept to apply to any type of Subject Actor (User, Device, and Application).

RBAC is based on the set of Roles assigned to a Subject Actor. Each Role has a set of permissions. Roles govern what operations a Subject Actor may perform on a particular Target Actor. RBAC may be mapped to support MAC or DAC.

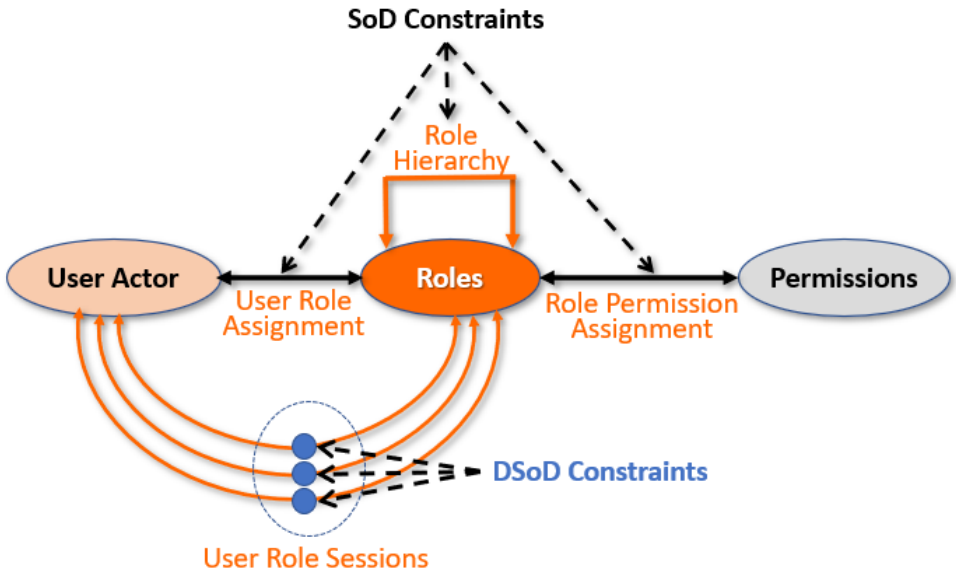

Figure 5 - Simplified Role Objects

Role-based Policies assign Users a set of Roles that govern the set of operations a User can perform. Privileges or permissions are encapsulated in Roles which are then designated to Users to gain the corresponding privileges. These concepts can be visualized in Figure 5 which shows an example using a User Subject Actor.

Roles can be reassigned easily to Users, and multiple Users can be assigned the same Role concurrently. This flexibility can also be constrained, e.g., using SoD.

In Flat RBAC (RBAC level 0), Subject Actors may have one or more Roles; permissions are assigned to Roles, and Subject Actors acquire permissions by assuming a set of Roles.

In Hierarchical RBAC (RBAC level 1), Role hierarchies are supported. A Role hierarchy is a partially ordered set (poset) which enables superordinate Roles to acquire the permissions of their subordinate Roles. Restricted Hierarchical RBAC imposes restrictions on this Role hierarchy and limits the hierarchy to simple structures such as trees. General Hierarchical RBAC supports arbitrary partial orders.

Two additional levels of RBAC add power but introduce complexity.

RBAC Model	Hierarchies	Constraints
Flat RBAC (level 0)	NO	NO
Hierarchical RBAC (level 1)	YES	NO
Constrained RBAC (level 2)	NO	YES
Symmetric RBAC (level 3)	YES	YES

Table 11 - RBAC Hierarchies and Constraints

Constrained RBAC (level 2) introduces the concept of constraints. A constraint in RBAC is used to prohibit a state that would otherwise be authorized. Note that hierarchies are not allowed in Constrained RBAC to reduce the computational complexity of computing permissions. Constraints may be viewed as predicates which, when applied to a permission, Allow or disallow that permission. In Constrained RBAC, Separation of Duty (SoD) [discussed in section 11.3] is enforced. Role activation can be associated with the Roles corresponding to a User Actor's Session.

In Symmetric RBAC (RBAC level 3), permissions can be reviewed and changed for each specific Role. Constraints can be applied to the Role Hierarchy (e.g., the number of subordinate Roles may be limited, and two or more Roles can be constrained to not have a common superordinate Role). Dynamic SoD places constraints on Subject Actors that can be assigned to a set of Roles, which reduces the number of potential permissions that can be made available to a given Subject Actor. For example, no Subject Actor may activate more than *n* Roles in each Session. Dynamic SOD also enables Roles to be chained (e.g., the Supervisor Accountant Role cannot open the accounting system for review until it has been closed by an Accountant Role). Another example is that a Subject Actor cannot activate two Roles.

12.1.4 Attribute-based Access Control (ABAC)

Attribute Based Access Control (ABAC) [24], is an Access Control Policy in which the operations that can be performed are a function of the attributes of the Subject and Target Actors that want to interact. Basic ABAC evaluates three types of attributes: Subject attributes, Target attributes, and environmental attributes. For example, a Subject attribute for a User Subject could be "employee", an environmental attribute could be "location", and a Target attribute could be "HR database".

ABAC is a logical access control methodology where Authorization to perform a set of operations is determined by evaluating attributes associated with the Subject Actor, Target Actor, requested operations, and, in some cases, environment attributes against Policy, rules, or relationships that describe the allowable operations for a given set of attributes. With ABAC:

- Attributes are defined as a three-tuple: {category[0..1], attribute [1..1], value[1..1]} (meaning that the category field is optional, but the attribute and value fields are required)
- Subject Actors are Users, Devices or Applications that cause information to flow to Target Actors, which in turn changes the system state
- Target Actors are Actors that need to be protected by Access Control, such as devices, files, records, programs, networks, and domains
- Operations are the execution of a function at the request of a Subject Actor upon a Target Actor.
- Policy is the representation of rules that define the set of allowable operations that a Subject Actor may perform upon a Target Actor

Each Subject and Target Actor has a set of attributes, as well as an environment. The operations to be performed are governed by a Policy, which evaluates if the set of attributes presented allows the operations to be performed or not. Note that ABAC does *not* define what a Policy is; it just assumes that they exist.

An ABAC Policy does not have to be a Policy language. The Policies that bind the Subject Actor and Target Actor attributes specify permissions.

ABAC is conceptually defined as follows:

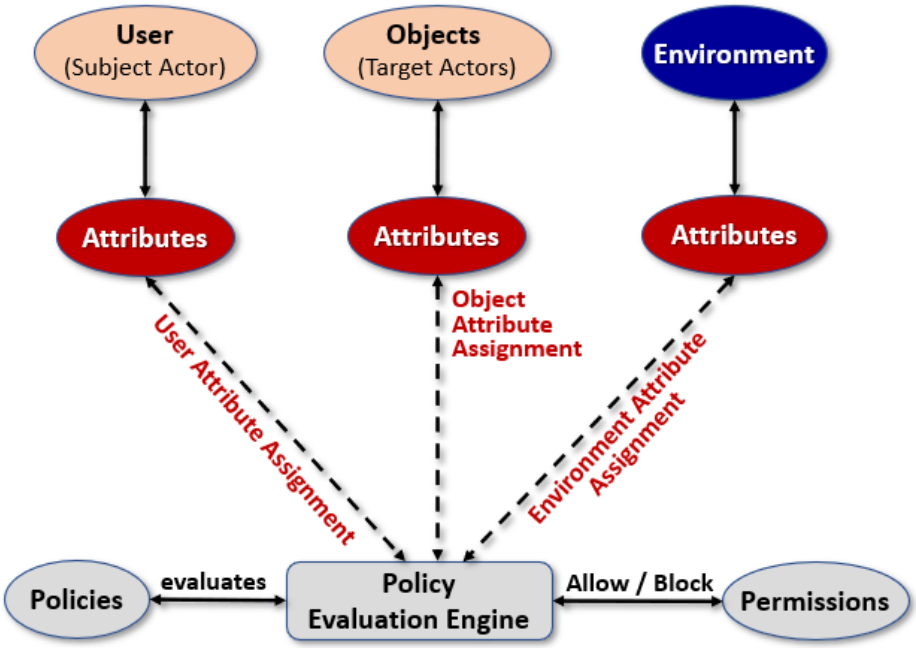

Figure 6 - Simplified Conceptual Model of ABAC

This is translated to the terminology of this document as follows: ABAC controls access to Target Actors by evaluating Policies against the Subject Actor and Target Actor attributes, along with the attributes of the requested operation and possibly environmental conditions. Environmental conditions, which are part of Context, are independent of Subject and Target Actors, and may include the current time, day of the week, location of a user, type of connection, or the current threat level. For example, a Policy evaluates the Attributes of the Subject and Target Actors. ABAC can, like RBAC, be configured to support MAC and DAC.

12.2 Rationale for Policy-Based Access Control

After reviewing the different Access Control approaches described in section 12.1, the common denominator is Policy. Recall that the definition of a Policy is: "Policy is a set of rules used to manage and control the changing or maintaining of the state of one or more managed objects". Thus, whether the requirement is to base Access Control decisions on Context, content, multiple factors, or other criteria, each Access Control method can be represented by making Policy, as opposed to Subjects, objects, Roles, and attributes, the center of the model.

This standard therefore emphasizes the Policy portion of the Access Control decision, and uses Policy in two distinct ways:

1. to choose which criteria to use to make an Access Control decision for a given situation, and
2. a vehicle to convey the Access Control decision

13 Policy Functional Entities

A Policy defines the type of rules, conditions, and constraints for a Service. A system that implements Policies utilizes the following Policy functions:

- **Policy Administration Point** (PAP): an entity that is responsible for configuration and maintenance of all other functional entities that make up a Policy
- **Policy Information Point** (PIP): a repository that contains generic information used by Policies. Examples include Session, geolocation, reputation, and risk calculation data, and associated metadata.
- **Policy Decision Point** (PDP): an entity that evaluates an access request using a set of Evaluation Policies, along with applicable information from the PIP. The term "Evaluation Policies" may include Authorization Policies, as well as more robust mechanisms (e.g., RBAC or ABAC) that provide a binary response (i.e., permit or deny access) to send to the PEP. Put another way, the PDP decides whether to Authorize the Actor based on applicable information.
- **Policy Enforcement Point** (PEP): An entity that implements Policy decisions that were made by the PDP. The PEP receives an access request and forwards this request to the PDP. When the PEP receives the response from the PDP, the PEP implements that Policy decision.

Figure 7 provides a simplified example of how access requests are treated with PBAC. This figure presumes that the Actors have been Authenticated and Authorized.

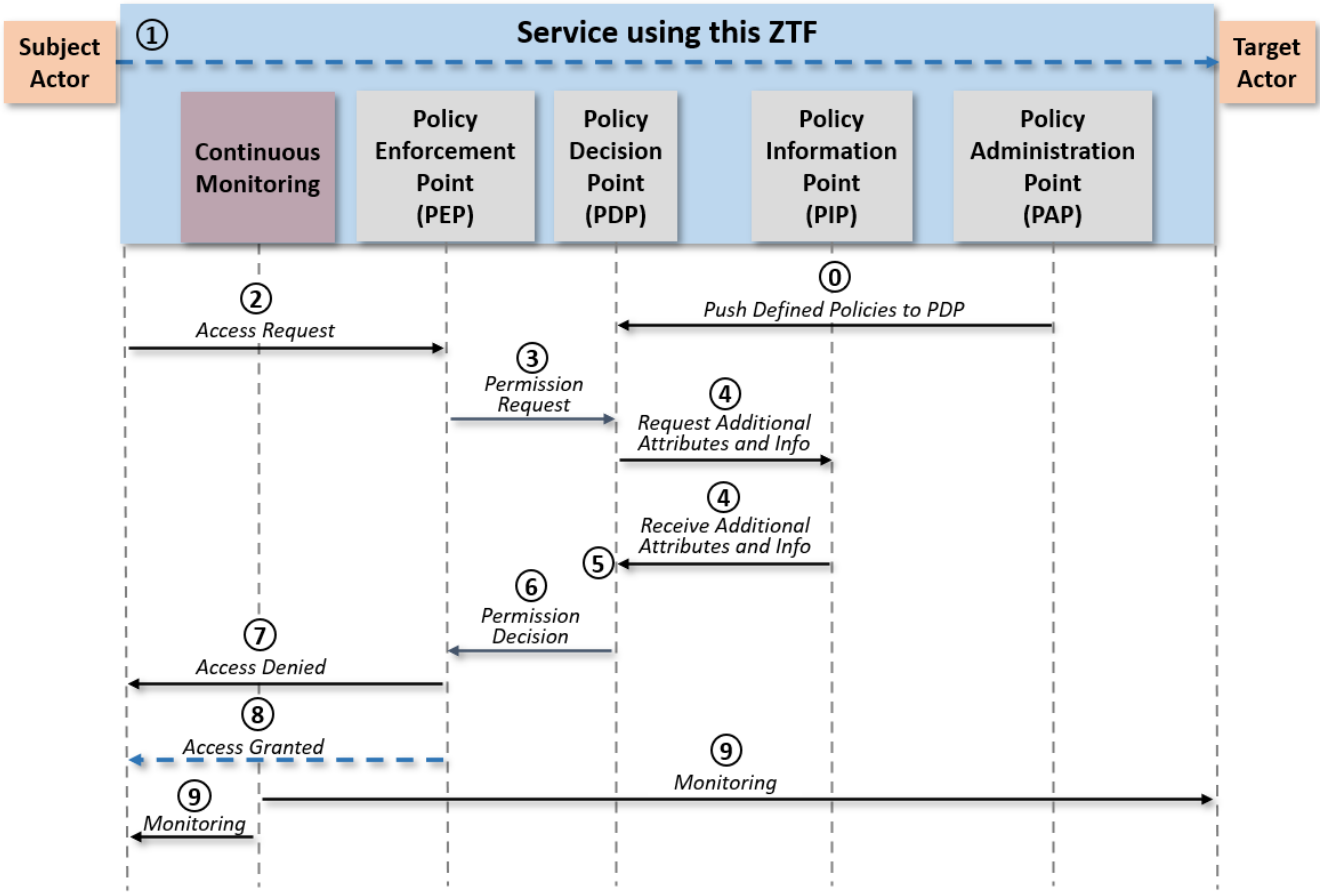

Figure 7 - Steps of Policy functions in action

These steps occur after the Subject Actor is authenticated by the IdP and authorized to use the Service.

0. Access Control Policies defined and stored in the PAP are pushed to the PDP
1. A Subject Actor wants to access a Target Actor for a Service using a Zero Trust Framework
2. The Service sends the access request to the PEP
3. The PEP sends the access request to the PDP
4. The PDP requests and receives any required additional attributes and information, e.g., Context and reputation, for deciding whether to grant access for this operation from the PIP
5. The PDP evaluates the permission request through one of the following mechanisms:
    a. A set of Access Control Policies is selected based on an existing Access Control mechanism for this request type, e.g., MAC, DAC, RBAC, or ABAC
    b. A set of Access Control Policies is used to evaluate this request
    c. A combination of the two
6. The permission decision reached by the PDP is sent back to the PEP
7. If the permission decision is to Block, then execution terminates and the Service Blocks the Subject Actor from accessing the Target Actor. Access Control Policies may be used to determine what, if any, additional information is provided to the Service which may or may not inform the Subject Actor.
8. If the permission decision is Accept, then the Service Allows the Subject Actor to access the Target Actor
9. The Subject and Target Actors' behaviors are continuously monitored by the Service using the methods described in section 17, regardless of the permission decision

[R53] An Access Control Policy, used by a Service that uses this ZTF, **MUST** use a Policy Information Point to store and retrieve attributes and data required for Policy evaluation by the Policy Decision Point

[D8] An Access Control Policy, used by a Service that uses this ZTF, **SHOULD** use a Policy Administration Point to create, manage, and delete Policies

[R54] An Access Control Policy, used by a Service that uses this ZTF, **MUST** use a Policy Decision Point to make Access Control decisions

[R55] An Access Control Policy, used by a Service that uses this ZTF, **MUST** use a Policy Enforcement Point to implement decisions made by the Policy Decision Point with which it is associated

The Policy functional entities and their requirements described in this section need to be incorporated as part of a Policy management system. However, how they are implemented is within the discretion of the Service Provider, e.g., a Service Provider may combine PEP and PDP functions in their implementation rather than implement them as separate entities.

13.1 Policy Actions

Policy Actions apply to a Subject Actor wanting to access a specific Target Actor. The Policy Actions may change triggered by the passage of a predetermined amount of time, or an Event-based on the continuous monitoring of the Actors and the corresponding Session. Refer to section 17 for Events created during continuous monitoring that can trigger a change in Policy Actions.

[R56] Only one of the following Policy Actions **MUST** be applied when a Policy Criterion is matched:

- Allow
- Block

The "Allow" Policy Action means that for a given Session between a specific Subject Actor and Target Actor, the Policy End Point permits the Subject Actor to send all IP Packets to the Target Actor and the receive IP Packets from the Target Actor for which Policies are applied.

The "Block" Policy Action means that for a given Session, between a specific Subject Actor and Target Actor, the Policy End Point denies the Subject Actor from sending any IP Packets to the Target Actor and denies the receipt of IP Packets from the Target Actor for which Policies are applied.

[R57] Each applied Policy Action **MUST** create an audit Event capturing the parameters agreed to from the following list:

- The Policy which was applied
- How the Policy was applied
- Where the Policy was applied
- Who/What triggered the Policy Action
- What Event triggered the Policy Action
- What was the Policy Action
- Why the Policy Action was applied
- When the Policy was applied

How the Service Provider and Subscriber agree to the items listed in [R57] and how the Service Provider conveys them to the Subscriber is beyond the scope of this document and defined by the Service using this ZTF.

[R58] The audit Event **MUST** be tamper-proof

Tamper-proof in the context of this document is defined per NISTIR 8202 [16] to be a process which makes alterations to the data difficult (hard to perform), costly (expensive to perform), or both.

[D9] An audit Event **SHOULD** be encrypted with a security strength of at least 256 bits

Security strength in the context of this document is defined per NIST Special Publication 800-57 Part 1 Revision 5 [20] as a number associated with the amount of work, i.e., the number of operations, that is required to break a cryptographic algorithm or system. Note, the security strength is typically specified in bits.

Each Session is typically continuously monitored because either Actor may have Risk calculation data from the Policy Information Point, or the Risk level determined by Risk-Awareness to be sufficiently high to Allow but not sufficiently high to Block the Session. Note that this monitoring is specific to this Session using the Continuous Monitoring techniques defined in section 17.

13.2 Policy End Points

A Policy End Point is a location where one or more Policy-related functions are placed. Policies are executed (via PEP) and monitored at a Policy End Point. A Policy End Point is placed at locations where practical when implementing a Zero Trust Framework. Policy End Points are optimally placed in a Device or Application Actor. However, this may not be practical, e.g., inside an IoT Device with minimal

processing capability, in which case the Policy End Point would be placed as close as possible to the Actor. Furthermore, a Subject Actor may be able to reach different Target Actors via different networks. Therefore, a Policy End Point must be placed at each access point where the Subject Actor can reach the Target Actor to ensure that Policies can be effectively applied. Example use cases for the placement of Policy End Points is illustrated in Appendix A:.

[R59] A Service compliant with a Zero Trust Framework **MUST** provide a Policy End Point that serves each Actor

Note that [R59] could be a Policy End Point for each Actor or a common Policy End Point that serves multiple Actors, e.g., a Policy End Point in an SD-WAN Edge that serves all Actors connected to the SD-WAN Edge.

[R60] An Actor **MUST** be associated with, at least, one Policy End Point

14 Policy-based Access Control (PBAC)

Policy-based Access Control (PBAC), is an Access Control method that uses Policies to determine the appropriate type of Access Control based, e.g., MAC, DAC, RBAC, or ABAC, on the Subject and Target Actors' Service Attributes and behavior, the current Situation, and applicable business requirements

The common denominator in the Access Control approaches described in section 12 is Policy. A Policy is a set of rules that is used to manage and control the changing or maintaining of the state of one or more managed objects. Thus, whether the requirement is to base Access Control decisions on Context, content, multiple factors, or other criteria, all Access Control decisions require a set of Policies to evaluate whether access is granted or not. Other actions, e.g., logging, may also be defined by these Policies.

This standard therefore emphasizes the Policy portion of the Access Control decision, and uses Policy in two distinct ways:

1. Choose which criteria to use to make an Access Control decision for a given Situation
2. Determine the Access Control decision

[R61] All Policy-Based Access Control Decisions **MUST** use one or more Policies to decide if access is granted

The customizable Policy-based Access Control (PBAC) decision provided is Situation-Aware. Conceptually, PBAC has two approaches:

1. PBAC uses a Policy to choose MAC, DAC, RBAC, or ABAC as the Access Control scheme for a given Access Control request
2. PBAC can use Policies to define multiple, customized Access Control for a given Access Control request

Item 1 above provides backwards compatibility for current and legacy systems. Item 2 above enables Policies to implement customized Access Control that may include elements of, extensions to, or a combination of, MAC, DAC, RBAC, and ABAC Access Control mechanisms.

Both forms of PBAC described above have three important qualities (described in more detail in later sections of this document):

- Context Awareness (section 16.1)
- Situation Awareness (section 16.2)
- Risk Awareness (section 16.3)
- Business Awareness (section 16.4)

14.1 PBAC Policies using MAC

The following requirements apply:

[R62] When PBAC Policies, used by a Service that uses this ZTF, use Mandatory Access Control, the MAC criteria, set by the Actor with the System Administrator Role, **MUST NOT** be altered by any other Actor without the Role of System Administrator

[R63] When PBAC Policies, used by a Service that uses this ZTF, use Mandatory Access Control, the Security Label, if present, **MUST** be used as input into the Authorization process

[R64] When PBAC Policies, used by a Service that uses this ZTF, use Mandatory Access Control, a Subject Actor that has been granted access to a Target Actor **MUST NOT** be able to perform any of the following actions:

- Change security attributes of the Subject Actor, Target Actor, or system components of the ZTF
- Choose security attributes to be created with newly created or modified Target Actors
- Grant privileges to other Subject Actors as they would a Delegate
- Change the rules governing Access Control

A Delegate is the Actor receiving the assignment of a set of privileges from another Actor that has the authority to assign said privileges and Policies for a given time period.

[D10] When PBAC Policies, used by a Service that uses this ZTF, use Mandatory Access Control, a Subject Actor that has been granted access **SHOULD** be prevented, whenever possible, from sending information to unauthorized Target Actors

[D10] indicates that it is impossible to prevent a Subject Actor from sending information to an unauthorized Target Actor 100% of the time. Economic incentive/disincentive models to prevent unauthorized data sharing are generally used if data is leaked illegally resulting in economic punishment that can be exacted swiftly and without recourse.

14.2 PBAC Policies using DAC

The following requirements apply:

[R65] When PBAC policies, used by a Service that uses this ZTF, use Discretionary Access Control, a Subject Actor that has been granted access to a Target Actor **MUST NOT** perform any of the following:

- Send the accessed information to other authorized Subject Actors or Target Actors
- Change security attributes on authorized Subject Actors, Target Actors, and system components of the ZTF
- Choose the security attributes to be associated with the Target Actors
- Grant its privileges to other authorized Subject Actors via Delegation
- Change the rules governing Access Control

14.3 PBAC Policies using ABAC

The following requirements apply:

[R66] When PBAC Policies, used by a Service that uses this ZTF, use Attribute-Based Access Control, a Policy **MUST** assign one or more Subject attributes to a Subject Actor

[R67] When PBAC Policies, used by a Service that uses this ZTF, use Attribute-Based Access Control, a Policy **MUST** assign one or more Target attributes to a Target Actor

[R68] When PBAC Policies, used by a Service that uses this ZTF, use Attribute-Based Access Control, a set of Policies **MUST** be used to assign access privileges to a Subject Actor in order for a Subject Actor to access any Target Actor

[R69] When PBAC Policies, used by a Service that uses this ZTF, use Attribute-Based Access Control, the Environmental Conditions **MUST** be represented by one or more objects whose attributes represent at least the current conditions of an Actor

[R70] When PBAC Policies, used by a Service that uses this ZTF, use Attribute-Based Access Control, the Policy **MUST** assess the Subject Actor attributes, Target Actor attributes, Environmental Conditions, and applicable Policy Rules to determine if access is authorized or not.

14.4 PBAC Policies using RBAC

The following requirements apply:

[R71] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), the Policy **MUST** be able to assign to a Subject Actor one or more Roles

[R72] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), a Role **MUST** support the capability to have one or more permissions

[R73] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), an operation **MUST** support the capability to be assigned to one or more permissions

[R74] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), a Permission **MUST** support the capability to be assigned to one or more operations

[R75] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), each Session **MUST** be associated with a single Subject Actor

[R76] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), each Session **MUST** authorize one or more Roles

[R77] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), all Roles **MUST** be authorized before any access for a given Role is permitted

[R78] When a PBAC Policy, used by a Service that uses this ZTF, uses Role-Based Access Control (RBAC), an Actor with Role System Administrator **MUST** be able to change the Authorization of one or more Roles within a Session

14.4.1 PBAC Policies using RBAC Level 0

The following requirements apply:

[R79] When a PBAC Policy, used by a Service that uses this ZTF, uses Flat RBAC (RBAC level 0), the Policy **MUST** Allow an operation only if the Subject Actor has been assigned an appropriate Role

14.4.2 PBAC Policies using RBAC Level 1

The following requirements apply:

[R80] When a PBAC Policy, used by a Service that uses this ZTF, uses Hierarchical RBAC (RBAC level 1), PBAC implementations **MUST** include all requirements of RBAC level 0

[R81] When a PBAC Policy, used by a Service that uses this ZTF, uses Hierarchical RBAC (RBAC level 1), all Roles **MUST** support Role inheritance for arbitrary structures

14.4.3 PBAC Policies using RBAC Level 2

Note that RBAC Level 2 does not support Role hierarchies.

The following requirements apply:

[R82] When a PBAC Policy, used by a Service that uses this ZTF, uses Constrained RBAC (RBAC level 2), PBAC implementations **MUST** include all requirements of RBAC level 0

[R83] When a PBAC Policy, used by a Service that uses this ZTF, uses Constrained RBAC (RBAC level 2), constraints on Roles assigned to a Subject Actor **MUST** be enforced

[R84] When a PBAC Policy, used by a Service that uses this ZTF, uses Constrained RBAC (RBAC level 2), constraints on Permissions assigned to a Subject Actor **MUST** be enforced

[R85] When a PBAC Policy, used by a Service that uses this ZTF, uses Constrained RBAC (RBAC level 2), constraints on the number of Sessions that a Subject Actor can have **MUST** be enforced

[R86] When a PBAC Policy, used by a Service that uses this ZTF, uses Constrained RBAC (RBAC level 2), SoD **MUST** be enforced

14.4.4 PBAC Policies using RBAC Level 3

The following requirements apply:

[R87] When a PBAC Policy, used by a Service that uses this ZTF, uses Symmetric RBAC (RBAC level 3), PBAC implementations **MUST** include all requirements of RBAC level 0

[R88] When a PBAC Policy, used by a Service that uses this ZTF, uses Symmetric RBAC (RBAC level 3), PBAC implementations **MUST** include all requirements of RBAC level 1

[R89] When a PBAC Policy, used by a Service that uses this ZTF, uses Symmetric RBAC (RBAC level 3), PBAC implementations **MUST** include all requirements of RBAC level 2

[R90] When a PBAC Policy, used by a Service that uses this ZTF, uses Symmetric RBAC (RBAC level 3), Dynamic SOD constraints on the system state **MUST** be enforced in RBAC level 3 policies

15 Delegation and Revocation of Actor Privileges

Delegation is the process where an Originating Actor assigns some or all its privileges derived from Policies to one or more Recipient Actors. A Recipient Actor is the Actor receiving some or all its privileges derived from Policies from one or more Originating Actors. An Originating Actor is the Actor assigning some or all its privileges derived from Policies to one or more Recipient Actors. The Recipient Actor takes on the privileges of the Originating Actor until the delegation is revoked. An Originating Actor or Recipient Actor can only be a Subject Actor.

[D11] A Service using this ZTF **SHOULD** support Delegation and Revocation of Actor privileges

[CR2]< [D11] Delegation operations **MUST** be defined for a specific time period if [D11] is met

Delegation has two common implementations.

The first copies the Authentication Credentials of the Originating Actor to one or more Recipient Actors. For MAC, DAC, and similar Access Control schemes, this enables the Recipient Actor (recipient of the delegation) to perform the same tasks as the Originating Actor. This is referred to as Direct Delegation. Direct Delegation is the process where a Recipient Actor receives all privileges derived from Policies from an Originating Actor.

The second uses a dedicated mechanism, such as a Role, that grants the Recipient Actor with unique permissions that the Originating Actor is delegating to the Recipient Actor. In RBAC, this takes the form of delegating a set of Roles that the Recipient Actor can assume. This is referred to as Indirect Delegation. Indirect Delegation is the process that uses a dedicated mechanism to grant an Originating Actor's unique privileges derived from Policies to one or more Recipient Actors. In either case, identifiers of both the Originating Actor and Recipient Actor(s) are used. If other entities, such as a Role, are used, then an identifier is used for each such entity.

[CR3]< [D11] The Originating and Recipient Actor **MUST** be distinguishable from one another in a cryptographically verifiable manner if [D11] is met

Distinguishable in a cryptographically verifiable (refer to NIST SP 800-63-3 [21]) manner in this context means that the Originating Actor and Recipient Actor(s) each have different unique identifiers provably associated with one or more cryptographic secrets only the Actor controls, but does not know, such as a cryptographic private key, and the respective Actor can cryptographically prove to a verifier that only the Actor controls the cryptographic secret. The requirements for this are defined for Actors in section 9.

In a ZTF, Direct Delegation is strongly discouraged, even if the Originating Actor and Recipient Actor are in the same Trust Domain, because sharing Credentials between two Actors does not allow for the disambiguation of Actor actions which is required in a ZTF. PBAC should always use Indirect Delegation, since it enables more fine-grained delegation decisions.

[CD1]<[D11] PBAC **SHOULD** use Indirect Delegation if [D11] is met

PBAC extends Indirect Delegation by defining three special delegating capabilities for the Originating Actor. These delegating capabilities define the set of permissions that the Originating Actor may delegate to a Recipient Actor. The first delegating capability, called delegation permission, determines whether an Actor has the authority to delegate permissions. If this delegating capability is true, then the second delegating capability, called delegation scope, defines the set of permissions to be delegated, and the third delegating capability (called delegating Policy) specifies a set of Policy rules that identify the permissions

that an Originating Actor can delegate to a Recipient Actor. Each Policy rule can define a combination of these delegating capabilities, Roles, or other mechanisms that represents a permission. The use of dedicated capabilities simplifies auditing delegations, and the use of Policy rules implements the permissions to be delegated in a flexible and extensible manner. Finally, the Originating Actor may set the Delegation permission capability of the Recipient Actor to true. If this is done, then the Recipient Actor may in turn Delegate the set of permissions specified in the Delegation scope capability using the set of Policies specified in the delegating Policy capability. If this is not done, then the Recipient Actor may not Delegate permissions.

[CR4]< [D11]A PBAC implementation **MUST** use a dedicated capability or mechanism to signify that an Actor can Delegate permissions if [D11] is met

[CR5]< [D11]PBAC **MUST** use a set of Policy rules to define delegated permissions and privileges if [D11] is met

[CO1]<[D11]An Originating Actor **MAY** enable a Recipient Actor to Delegate some or all of the permissions that it has delegated to the Recipient Actor if [D11] is met

Indirect mechanisms of specifying permissions, e.g., Roles, are preferred to direct assignment. This is because the set of permissions required to perform a task may be organized in a complex manner. For example, if the set of permissions required to execute an operation are defined in multiple objects, then each of those permissions are needed by the Recipient Actor. This can be exacerbated when the set of objects that contain these permissions also contain additional capabilities. Then, the Delegation option must be careful to select only those permissions that apply. In contrast, if the set of permissions are gathered into a Role, then all that is needed is to assign that Role to the Recipient Actor.

[CD2]<[D11]PBAC **SHOULD** use an indirect mechanism to define permissions that are delegated if [D11] is met

Care should be taken to avoid combining Roles (including inherited Roles) without performing proper Role engineering. Otherwise, it becomes difficult for the Recipient Actor to understand what Roles are required to perform a particular task.

Revocation is the act of cancelling a delegated set of permissions. Revocation may be specified for a particular time or time period. Revocation in PBAC mirrors Delegation and requires three revocation capabilities for the Actor performing the revocation operation. The first revoking capability, called revocation permission, determines whether an Actor has the authority to revoke permissions. If this revoking capability is true, then the second revoking capability, called revocation scope, defines the set of permissions to be revoked, and the third revoking capability (called revocation Policy) specifies a set of Policy rules that perform the revocation operation.

16 Lifecycle Management of Zero Trust Policies

Throughout the lifecycle of a Subject-Target Actor interaction, inclusive of the Session, the Policy management system must actively monitor and be made aware of any changes that may occur that warrant the interaction to be more closely monitored, limited, or blocked. Such changes to be aware of discussed in this section include Context Awareness, Situation Awareness, Business Awareness, and Risk Awareness.

16.1 Context Awareness

Context Awareness is the knowledge and understanding of how changes in Subject Actor needs, Environmental Conditions, and business goals affect decision-making. The Context of an Actor is the measured and inferred knowledge that collectively describes the Environmental Conditions in which an Actor exists or has existed. This definition emphasizes two types of knowledge: facts, which can be measured, and inferred data, which results from machine learning and reasoning processes applied to past and current Context. It also includes Context history, so that current decisions based on Context may benefit from past decisions, as well as observation of how the environment has changed.

16.1.1 Application of Context

Context can be applied in several different ways. Below is a listing of some examples.

- Personal and group information for a User, e.g., contact information, Roles played by the User, and profile and preference information
- Location of the User and any Devices that the User is using or interacting with, e.g., geo-code or the centroid of a surrounding polygon
- Some Devices are stationary while other Devices are mobile or restricted to a particular set of geographic coordinates or a geometric area
- Characteristics and behavior of a Device, e.g., type of access provided (if any), MAC and IP addresses, and features such as forward, routing, encryption
- Characteristics and behavior of any Applications used by a User
- Type of connection, e.g., public open Wi-Fi or home Wi-Fi
- Time
- Zones – Campus or branch office
- Risk

16.2 Situation Awareness

Situation Awareness is the perception of data and behavior that pertain to the relevant circumstances or conditions of a system or process, the comprehension of the meaning and significance of these data and behaviors, and how processes, actions, and new situations inferred from these data and processes are likely to evolve in the near future. Situation Awareness of an Actor enables more accurate and fruitful decision-making.

Situation Awareness consists of four actions:

- Gathering data (perception)
- Understanding the significance of the gathered data (through both facts and inferences)
- Determining what to do (if anything) in response to a given Event
- Performing those actions

Situation Awareness enables Policies Actions to be altered for a particular situation and can use inference as well as historical data to understand what is happening at a particular Context, why, and what, if anything, should be done in response.

16.3 Risk Awareness

Risk Awareness is the knowledge, understanding or recognition of the potential hazards that could be caused by or to a Subject or Target Actor that need to be considered for Access Control decisions. Risk Awareness may result in restricting a Subject Actor from accessing a Target Actor based on the predetermined risk assessment level of either the Subject Actor or Target Actor. A measure of the extent to which an Actor is threatened by a potential circumstance or event, and typically a function of: (a) the adverse impacts that would arise if the circumstance or event occurs; and (b) the likelihood of occurrence.

[D12] A Service adopting this ZTF **SHOULD** support Risk Awareness

[R91] When using Risk to take a Policy Action, the Service Provider **MUST** quantify the Risk by defining Risk levels from lowest to highest

[R92] When using Risk to take a Policy Action, the Service Provider **MUST** communicate the Risk levels defined in [R91] to the Subscriber

The Subscriber uses these Risk levels in their Policies which in turn would take a particular action, e.g., Block a particular Subject Actor, if the Risk level is above what is acceptable.

Note that the methodology for how Risk is calculated is beyond the scope of this document.

16.4 Business Awareness

Business Awareness is the knowledge and understanding of business concerns and regulations that need to be considered for Access Control decisions. Context, Situation, and Risk Awareness are critical to representing business goals and requirements in any environment that requires Policy-based management. Business Awareness of an Actor ensures that business rules are considered, when applicable, for Access Control decisions.

17 Continuous Monitoring Method Service Attribute

In a Zero Trust Framework, an Actor's status for a given Session must be continuously monitored and evaluated to ensure that the Actor is compliant with the Policies. Otherwise, the Subject Actor's access to a Target Actor, for a given Session, may need to be revoked or limited. Such monitoring would be driven by:

- A time-based evaluation
  - E.g., check the Policy every 30 minutes
- An event-based evaluation
  - E.g., check the Policy based on the occurrence of a particular event
- A data-driven evaluation
  - E.g., check the Policy based on a change in state of an Actor or an Actor's context, a behavioral change, or a change in the situation or risk assessment determined by machine learning (ML) or artificial intelligence (AI)

After a time-based or event-based evaluation has occurred and an Actor, for a given Session, is no longer deemed compliant with Policies, the Service will perform the pre-defined Policy Action, e.g., Block the Subject Actor after the 30 minutes using the time-based method and require the Subject Actor to re-authenticate. An Event is an asynchronous action or occurrence that initiates further analysis. An Event is a state change that could have negative, undesirable implications or negatively impact the security posture, e.g., risk, or impact the proper functionality of an Actor or a network or a positive impact whereby an Actor may have their access privileges restored.

The Continuous Monitoring Method Service Attribute specifies the details of the type of monitoring to be used for a given Session, per the appropriate Policy. The Continuous Monitoring Method Service Attribute is a 3-tuple of the form {*time, event, data*}.

- *time*, which refers to the time-based monitoring method, has a value of *None* or time interval *t*
- *event*, which refers to the event-based monitoring method, has a value of *None* or a {*list of events*}
- *data*, which refers to the data-driven monitoring method, has a value of *None* or a {*list of decisions*}

[R93] For the Continuous Monitoring Service Attribute, at least one of the values of {*time*, *event*, *data}* **MUST NOT** be *None*

All three methods can be applied to a given Session because each provides different types of protection. When *None* is specified, the specific monitoring method is not used.

More details and requirements for the continuous monitoring methods used in the Continuous Monitoring Method Service Attribute are discussed in sections 17.1, 17.2, and 17.3.

17.1 Time-Based Monitoring Method

Time-based monitoring evaluates the system state at a pre-determined time to determine if the system state has changed (to a less desirable state). If it has, then one or more Policies are executed to return the system state to an acceptable state. When this method is used where the monitoring is performed for a specific duration of time *t*, expressed in minutes, after which the Zero Trust Policy must be re-evaluated to determine compliance and potentially revise Policy Actions.

[R94] The Service Provider delivering a Service using this Zero Trust Framework **MUST** support the time-based monitoring method

Note that some industry or corporate compliance standards may require time-based monitoring, e.g., log out a User from a system after 30 minutes when there is no activity. However, time-based monitoring is insufficient because a security event could occur at any time. Therefore, the event-based method is also required to provide maximal protection in a Zero Trust implementation.

[R95] The Service Provider delivering a Service using the time-based monitoring method **MUST** allow the Subscriber to use different *t* values for each Actor Session being monitored

For example, a Subscriber may select *t* = 30 minutes for monitoring Subject Actor A and *t* = 60 minutes for monitoring Subject Actor B.

17.2 Event-Based Monitoring Method

Event-based monitoring collects a set of events to determine if any actions are required. The collected events are first processed to meet a set of criteria, such as anonymization or pseudonymization, de-duplication, or clustering. The processed set of events are then analyzed to determine if the system state has changed (to a less desirable state). If it has, then one or more Policy Actions are executed to return the system state to an acceptable state.

When this method is used, monitoring is performed based on a *{list of events}* that are evaluated by the Zero Trust Policy to determine compliance and potentially re-evaluate Policy Actions. How the Service Provider describes the events in the {*list of events}* is beyond the scope of this document and would be specified in the Service that uses this ZTF.

[R96] The Service Provider **MUST** support the event-based monitoring method

[R97] When the value of the Continuous Monitoring Method Service Attribute parameter *Event* is not *None*, the Service Provider **MUST** provide the *{list of events}* that can be used with this monitoring method

[R98] For each Actor Session being monitored using the event-based monitoring method, the Service Provider **MUST** allow the Subscriber to select one or more events from the Continuous Monitoring Method Service Attribute parameter *{list of events}* specified in [R97] to be able to apply a Policy Action

17.3 Data-Driven Monitoring Method

Data-driven monitoring uses artificial intelligence (AI), machine learning (ML), or data trending to determine anomalous behavior of an Actor that results in a decision for which Policy Actions may need to be taken. The decision is then analyzed to determine if the system state has changed (to a less desirable state). If it has, then one or more Policies Actions are executed to return the system state to an acceptable state.

When this method is used, monitoring is performed based on *a {list of decisions}* that are evaluated by the Zero Trust Policy to determine compliance and potentially revise Policy Actions. How the Service Provider describes the decisions in the {*list of decisions}* is beyond the scope of this document and would be specified in the Service that uses this ZTF.

[D13] The Service Provider **SHOULD** support the data-driven monitoring method

[CR6]< [D13] When the value of the Continuous Monitoring Method Service Attribute parameter Data is not None, the Service Provider **MUST** provide the *{list of decisions}* that can be produced by this monitoring method

An example of the *{list of decisions}* could be as follows:

- Take no new Policy Action, i.e., use currently defined Policy Actions
- Override Policy Action with a different Policy Action based on the recommendation of the ML algorithm

Another example of the *{list of decisions}* could be the notification of possible actions to take based on predefined threat Risk assessment levels. In this example, using a color-coding scheme for Risk with green being the least amount of Risk, yellow being an acceptable amount of Risk, and red being the highest amount of Risk.

- Green Risk level - take no new Policy Action, i.e., use currently defined Policy Actions for the User, Device, or Application
- Yellow Risk level - propose taking a modified Policy Action, e.g., Allow but provide additional monitoring of the User, Device, or Application
- Red Risk level - override Policy Action and Block the User, Device, or Application

An example of the Continuous Monitoring Method Service Attribute where only the time-based monitoring method is used is described below using a JSON format with parameter-value pairs for the Service Attribute parameters and their respective values.

```
{"ContinuousMonitoringMethodServiceAttribute":[
  { "Time":"30" },
  { "Event":"None" }
  { "Data":"None" }
]}
```

18 References and Resources

[1] IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels, S. Bradner – March 1997

[2] IETF RFC 3198: Terminology for Policy-Based Management, A. Westerinen, et. al. – Nov. 2001. Copyright (C) The Internet Society (2001).

[3] IETF RFC 4511: Lightweight Directory Access Protocol (LDAP): The Protocol, J. Sermersheim - June 2006. Copyright (C) The Internet Society (2006).

[4] IETF RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, T. Dierks and E. Rescorla - August 2008. Copyright (C) The IETF Trust (2008).

[5] IETF RFC 5849: The OAuth 1.0 Protocol, E. Hammer-Lahav - April 2010. Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

[6] IETF RFC 6749: The OAuth 2.0 Authorization Framework, D. Hardt - October 2012. Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

[7] IETF RFC 8174: Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, B. Leiba - May 2017. Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

[8] IETF RFC 8820: URI Design and Ownership, M. Nottingham - June 2020. Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

[9] ITU-T X.509: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks, Oct. 2016

[10] OpenID Specifications: https://openid.net/developers/specs 16

[11] Organization for the Advancement of Structured Information (OASIS) SAML v2.0, March 2005

[12] MEF 70.1: SD-WAN Service Attributes and Service Framework

[13] MEF 78.1: MEF Core Model (MCM), July 2020

[14] MEF 95, Policy Driven Orchestration, July 2021

[15] ETSI GS ENI 005 v1.1.1: Experiential Networked Intelligence (ENI); System Architecture, Sept. 2019

[16] NISTIR 8202: Blockchain Technology Overview, Oct. 2018

[17] NIST SP 800-12 Rev. 1: An Introduction to Information Security, June 2017

[18] NIST SP 800-37 Rev. 2: Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy, December 2018

[19] NIST SP 800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations, Sept. 2020

[20] NIST SP 800-57 Part 1 Rev. 5: Recommendation for Key Management: Part 1 – General, May 2020

[21] NIST SP 800-63-3: Digital Identity Guidelines, June 2017

[22] NIST SP 800-95: Guide to Secure Web Services, Aug. 2007

[23] NIST SP 800-152: A Profile for U.S. Federal Cryptographic Key Management Systems (CKMS), Oct. 2015

[24] NIST SP 800-162: Guide to Attribute Based Access Control Definition and Considerations, August 2019

[25] NIST SP 800-175B Rev. 1: Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, March 2020

[26] NIST SP 800-192: Verification and Test Methods for Access Control Policies/Models, June 2017

[27] NIST SP 1800-161: Supply Chain Risk Management Practices for Federal Information Systems and Organizations, April 2015

[28] CNSSI No. 4009: Committee on National Security Systems (CNSS) Glossary, April 2015

[29] The Design of a New Context-Aware Policy Model for Autonomic Networking, J. Strassner, M. Foghlu, S. van der Meer, W. Donnelly, S. Samudrala, G. Cox, Y. Liu, M. Jiang, J. Zhang, IEEE Xplore, July 2008

[30] W3C - DIDs v1.0: Decentralized Identifiers (DIDs) v1.0, August 2021

[31] W3C - Verifiable Credentials Data Model v1.1: Verifiable Credentials Data Model 1.1, Mar. 2022

Appendix A: Use Cases for Policy End Points Placement (Informative)

This Appendix illustrates the placement of Policy End Points (EP) for different use case scenarios. It is only meant to be informative as they are many different use cases and scenarios for placement of Policy End Points.

A.1 Use Case for Policy EP placement for Applications within a Cloud

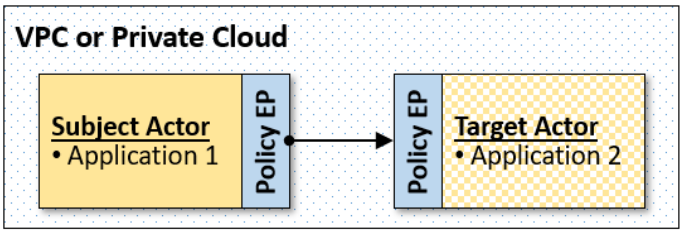

Figure 8 - Policy EPs in same Cloud

In use case A.1, Application 1, the Subject Actor, wants to access Application 2, the Target Actor. Applications 1 and 2 could be microservices running in containers or virtual machines (VMs) in a private cloud or virtual private cloud (VPC). A Policy End Point in the cloud is where Policies are enforced which could either Allow, Block or limit Application 1's access to Application 2.

A.2 Use Case for Policy EP placement for Applications between Clouds

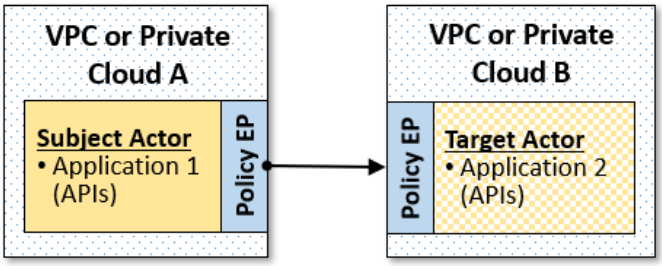

Figure 9 - Policy EPs in different Clouds

In use case A.2, Application 1 in Cloud A (Subject Actor) wants to access Application 2 (Target Actor) in Cloud B. Applications 1 and 2 could be microservices running in containers or VMs in a private cloud or VPC Each cloud could be running in the same physical facility or different locations, e.g., in different availability zones for one or more cloud service providers. A Policy End Point in each cloud is where Policies are enforced which could Allow, Block or limit Application 1 APIs access to Application 2 APIs.

A.3 Use Case for Policy EP placement for Applications on IoT Device and Applications in a Cloud

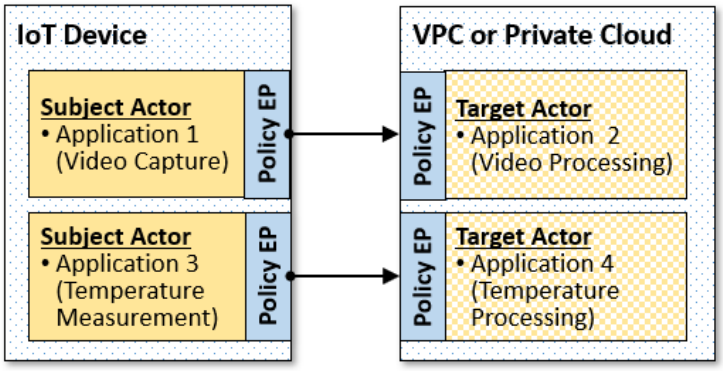

Figure 10 - Policy EPs on IoT Device and Cloud

In use case A.3, Application 1, a video capture Application (Subject Actor) operating in an IoT device wants to access Application 2, a video processing Application (Target Actor) operating in a cloud. Application 3, a temperature measurement Application (Subject Actor) operating in an IoT device, wants to access Application 4, a temperature processing Application (Target Actor) operating in a cloud.

A Policy End Point in the IoT Device is where Policies is enforced which could Allow, Block or limit Application 1's access to Application 2 in the cloud and could Block Application 1's access to Application 4 in the cloud. A Policy End Point in

the IoT Device is where Policies are enforced which could Allow, Block or limit Application 3's access to Application 4 in the cloud and could Block Application 3's access to Application 2 in the cloud.

A.4 Use Case for Policy EP placement on Edge Appliances and in Cloud

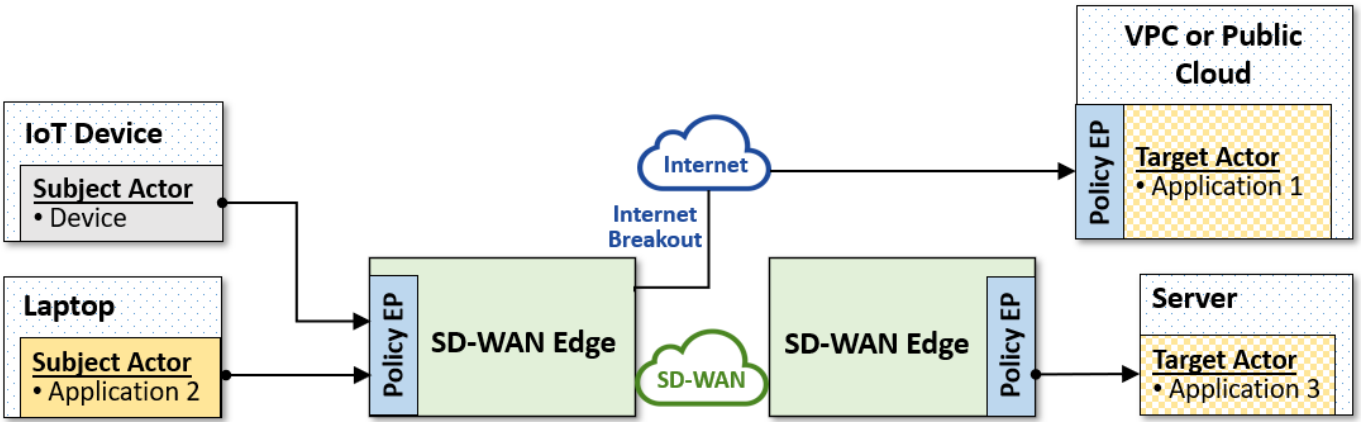

Figure 11 - Policy EPs on SD-WAN Edges and in Cloud

In use case A.4, an IoT Device is a Subject Actor that wants to access Application 1, a Target Actor operating in a cloud. Application 2 is the Subject Actor running on a laptop Device that wants to access Application 3, a Target Actor operating on a server.

For the IoT Device and laptop and server, the Policy End Point is located on an edge Device, e.g., SD-WAN Edge. Note that the edge Device in this use case provides an Internet breakout to connect to a VPC or public cloud where Application 1 is running. The Policy End Point on the SD-WAN Edge, on the left side of Figure 11, is where Policies are enforced which could Allow, Block, or limit the IoT device's access to Application 1 in the cloud and could Block the IoT Device's access to Applications 2 and 3.

A.5 Use Case for Policy EP placement on Residential Internet Gateway and in Public Cloud

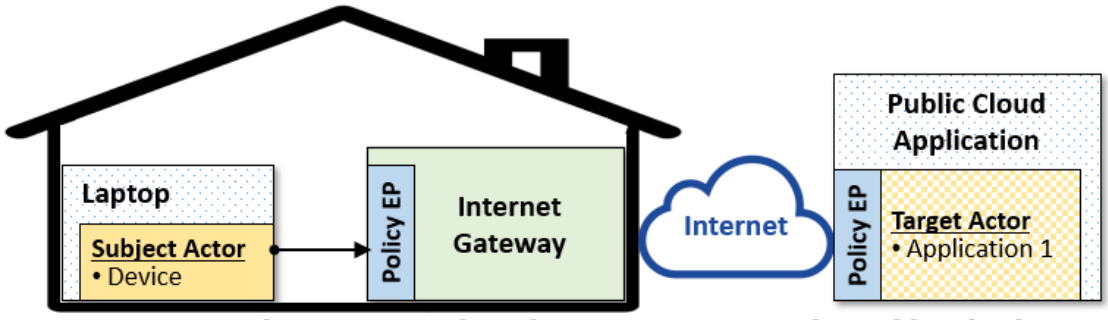

Figure 12 - Policy EP on Residential Internet Gateway and in Public Cloud

In use case A.5, a laptop Device is the Subject Actor that wants to access Application 1, a Target Actor operating in a public cloud. This would be a common use case for an employee working from home using an employer-supplied laptop and employer-supplied (directly from employer or from a service provider contracted by employer) Internet Gateway. The Internet Gateway could have an integral router, firewall and Wi-Fi access point and connects to a broadband access demarcation device such as a cable modem that connects to the Internet.

The Policy End Point is located in the Internet Gateway and is where Policies are enforced which could Allow, Block, or limit the Laptop's access to Application 1 in the cloud.

Appendix B: Use Case Example for Actor Delegation (Informative)

This Appendix illustrates the delegation of privileges between an Originating Actor and a Recipient Actor.

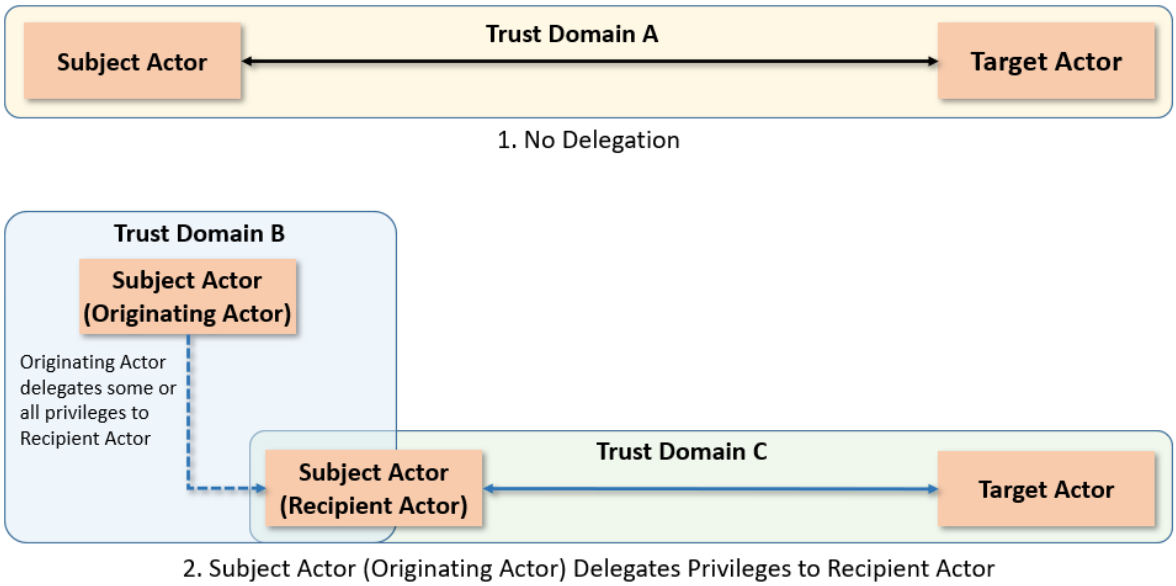

Figure 13 - Subject-Target Actor Interaction with and without Delegation

Figure 13 illustrates Subject Actor and Target Actor interaction with and without delegation. Case 1 illustrates no delegation between a Subject Actor and Target Actor. The Subject and Target Actors share a single Trust Domain (Trust Domain A).

Case 2 illustrates delegation of some or all of the Subject Actor privileges (the Originating Actor assigned the privileges) to a Recipient Actor (the receiver of privileges from the Originating Actor). This delegation enables the Recipient Actor to access a Target Actor as if they were the Originating Actor within the constraints of the privileges granted to the Recipient Actor if not all privileges were granted. Furthermore, in Case 2, the Originating (Subject) Actor can also access the Target Actor as they would have in the case with no delegation as in Case 1. Note, to simplify the diagram, this case is not illustrated in the case with Delegation. In Case 2, there are two Trust Domains (Trust Domains B and C). The Originating (Subject) Actor and Recipient (Subject) Actor are in Trust Domain B, since a trust relationship must exist between them. Recipient (Subject) Actor is also in Trust Domain C with the Target Actor since a trust relationship must exist between them.