REFERENCE

TRUSTED COMPUTING GROUP™

Trusted Multi-Tenant Infrastructure Work Group

Reference Framework

Version 1.00
Revision 1
December 12, 2013

Contact: admin@trustedcomputinggroup.org

TCG

TCG Published

Disclaimers, Notices, and License Terms

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, DOCUMENT OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this document and to the implementation of this document, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this document or any information herein.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on document licensing through membership agreements.

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

Revision History

R 1	Initial Release

Table of Contents

1 Scope and Audience

The TCG Trusted Multi-Tenant Infrastructure Reference Framework describes a broad set of foundational principles and requirements as well as a library of re-usable patterns where TCG technology may be applied between components in an enterprise context. They may likewise influence facets of other TCG committees and external standards bodies. These requirements and patterns serve as the “building blocks” for establishing Trusted Systems Domains and implementation of Trusted Multi-Tenant Infrastructure solutions. The requirements and patterns have been derived from the TCG Trusted Multi-Tenant Infrastructure Use Cases and are not intended to be a complete list of requirements or patterns, but to form the foundation of a library of best practices that will grow and change over time.

We anticipate the TMI Reference Framework will provide guidance and implementation patterns for cloud providers and consumers to implement a trusted computing base using shared multi-tenant infrastructure.

1.1 Key words

Highlighted Terms such as **Systems Domain** represent reserved terms within the presentation of best practices content. These terms have a specific defined meaning when used. When all or part of the reserved term is italicized, as in ***Challenger* Management Agent**, then then the term has been abstracted to refer to one or more specific terms (such as **Consumer Management agent** or **Provider Management Agent**, rather than create patterns otherwise duplicated for each of the similar terms.

1.2 Statement Type

Please note the text in this document will be of the kind informational statements, as a reference document is not intended to be normative. While not normative, the reference material does form the basis for assessment of best practices in the design and implementation of Trusted Multi-Tenant Infrastructure solutions, and may form the basis for future compliance and assessment approaches, at which time normative standards would be established.

2 TCG Trusted Multi-Tenant Infrastructure Reference Framework

The reference framework defines requirements and implementation patterns that use TCG technology and other appropriate industry standards to describe the foundational relationship between the various components in a trusted multi-tenant infrastructure (TMI) domain and how they interact. This interaction is based on three core foundational primitives:

- Establish a Trusted Context in which information can be exchanged between parties
- Exchange Information between parties within the trusted context
- Enforce Policy using the integrity measurements, assertions and attestations exchanged between parties

With these core primitives in place, a consumer domain could validate the ability of an environment provider to enforce separation and operational policy within a cloud or shared infrastructure. In terms of context – “separation” means that the services, systems and data that comprise a trusted security domain are completely separate from other trusted security domains within the cloud so that only by explicit allowances in operational policy from both trusted security domains can one domain even be aware of another domain. This separation may be either logical or physical depending on the policy of consumer and the capabilities of the provider.

A number of approaches could be taken to define a reference model. We could start with a proscribed architecture that should be implemented to solve a particular pre-defined problem set and then document the requirements and protocols to be used between components of that architecture. This assumes a well-known common problem set and can be very restrictive when applied to new problems or technology domains. An alternate path is to define the requirements that should be true to allow a set of components to come together and establish trusted relationships, then create a “tool box” of implementation patterns that may be used to meet the requirements. This allows for greater flexibility in the problem set to which the model can be applied, but takes much longer to build to the point where it can be applied to real world problems. This reference model is based on the second approach and defines the initial release of the tool box.

The framework defines core requirements and design principles that are necessary to establish an end to end trusted infrastructure. The core requirements give the basic concepts of the TMI and generic information relative to TMI functionality.

The framework then describes implementation patterns, measurements and validation mechanisms to address the security concerns of enterprise consumers. The patterns in this document are intended to be generic in nature, applicable to many specific industries and implementation needs.

The next document in the reference model set is the implementation guidance. This establishes a set of real world problems based on the use cases previously defined and shows how the patterns and requirements can be used to create a trusted multi-tenant infrastructure solution within a set of assumed policy constraints.

A later set of industry or implementation profile documents will describe how to use these patterns and design principles to meet the specific needs of various industries and establish infrastructures compliant with the standards and regulations associated with the subject industry or implementation type.

A TMI implementation designer should review and implement the information in the TMI reference framework specification and review the domain specific document for the intended industry or implementation type. The implementation specific document will contain normative statements that affect the design and implementation of a TMI. A TMI designer should review and implement the core requirements, including testing and evaluation, as set by the TCG Conformance Workgroup. The TMI should comply with the requirements and pass any evaluations set by the Conformance Workgroup. The TMI can undergo more stringent testing and evaluation based on industry requirements.

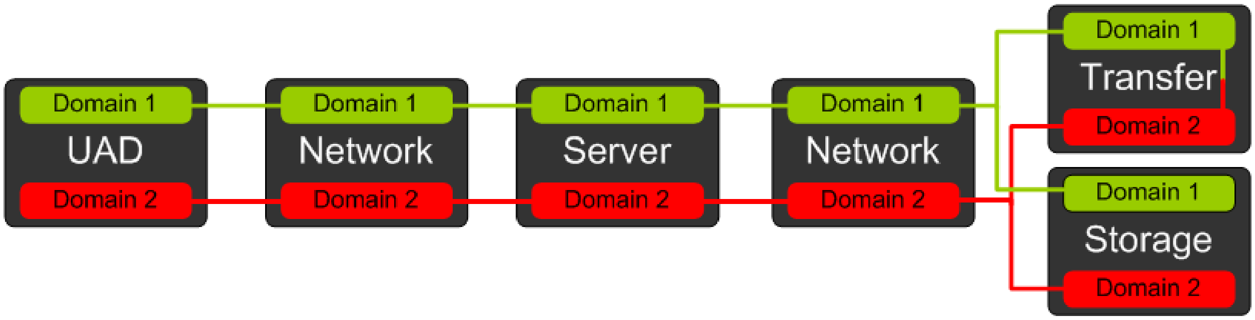
2.1 TMI Terminology

In this section we will discuss some of the specific terminology for the TMI Reference Framework – some of the terms that are going to be used are industry wide terms that have specific connotations when used in the TMI Context.

It is important to understand how ***trust*** is thought of in the context of the reference model. Trust is not a binary concept. Trust can be better thought of as acceptance of ***risk*** mitigation as sufficient. The degree of mitigation should exceed the level of risk exposure. If the mitigations are sufficient to address the risks then a solution can be described as trustworthy in that context. The reference model also talks about measuring and enforcing ***policy*** compliance. Policy in this case is a set of testable statements describing evaluation of the level of mitigation necessary to address the risk and establish trust.

Multi-Tenancy is described in many of the use cases, requirements and patterns in the context of an Infrastructure as a Service (IaaS) cloud offering. This leads to a discussion of the provisioning of servers, storage, network connections and so forth within or across data center providers. The reference model applies equally well to other constructs, such as multi-tenancy within a server, storage device, application mobile device or laptop.

The diagram below is the simplified view of the TMI Reference Architecture for IaaS and the view of the TMI in terms of multiple domains within a single logical service.



The table below is a list of those most common terms and some contextual information on each of the terms. In most cases the terms are actually "actors" within the use cases.

Term	Definition – Context
Asset	A functional IT component available for use within a Trusted Systems Domain
Client Device	An external (not a part of the Trusted Systems Domain) end user device that allows the consumer to access the Trusted Systems Domain
Communications Channel	A point-to-point or point to multipoint path as defined by all participants' policies that allows for communications between distinct domains.
Compliant Asset	An asset that has met the pre-determined criteria for use within the Trusted Systems Domain
Consumer	The party responsible for the assets within a Trusted Systems Domain
Consumer Audit Agent	Requests from the assets logs of their activity within the Trusted Systems Domain. The data required for each asset is controlled by the policy of the Trusted Systems Domain. Owned by the consumer.
Consumer Centralized Audit Collection Environment	Collects audit data from various Assets within the Trusted Systems Domain.
Consumer Management Agent	The Systems Management automation suite acting on behalf of a consumer organization as an operator and PEP for the Trusted Systems Domain

Term	Definition – Context
Data Exchange Gateway	Provides controlled information exchange across the boundary between asset domains. The data exchange gateway is a logical construct that is dictated by both the consumer policy and provider policy that allows for only a set of communications and protocols as dictated by the policies of both the consumer and provider. Responsibility of providing the Data Exchange Gateway is typically on the Provider and the policies of actual communication on the Consumer.
Peripheral Device	A device such as a printer, copier, scanner or other network connected device allocated within a Trusted Systems Domain
Policy	A principle or rule to guide decisions and achieve rational outcome(s)
Policy Decision Point	See RFC3198. *TMI uses the strict definition which may differ in some ways from the more focused usage in the TCG Trusted Network Connect (TNC) specifications*
Policy Enforcement Point	See RFC 3198. *TMI uses the strict definition which may differ in some ways from the more focused usage in the TCG Trusted Network Connect (TNC) specifications*
Policy Information Point	A mechanism that can provide information and attributes about users, environment and other facts useful in reaching a policy decision
Provider Audit Agent	Requests from the assets logs of their activity within the Provider Systems Domain. The data require for each asset is controlled by the policy of the Provider Systems Domain. Owned by the provider.
Provider Centralized Audit Collection Environment	Collects audit data from various Assets within the Provider Systems Domain.
Provider Environment	A logical grouping containing one or more components available for allocation to a consumer and governed by a consistent set of operational and security policies
Provider Environment Policy	A set of rules that establish a given policy of actions and allowed activity that governs the Provider Environment
Provider Management Agent	The Systems Management automation suite acting on behalf of a provider organization as an operator and PEP for the provider.
Provider Systems Domain Policy Store	The default repository of Policy Statements for each provider. Owned by the Provider

Term	Definition – Context
Quarantine	The Quarantine holds assets that have become non-compliant. Assets that are quarantined may be able to be provisioned so that they can be returned to service.
Server	A physical or virtual server machine
Storage Volume	A physical or virtual storage container capable of being mounted as a volume on an OS instance
Trusted Entity Store	The repository of information about assets and operators with which a trusted context has been established in a trusted systems domain. The store contains the identity, attestation keys, compliance statements and policy store location for each asset or operator
Trusted Systems Domain	A logical grouping containing infrastructure assets, service providers (operators), users, applications and information where a trusted context has been established and governed by a consistent set of operational and security policies
Trusted Systems Domain Policy Store	The default repository of Policy Statements for each Trusted Systems Domain. Owned by the Trusted Systems Domain.

3 TCG Trusted Multi-Tenant Infrastructure Core Requirements and Design Principles

The requirements and design principles are the first of two linked parts of the TMI Reference Framework. Each of the requirements in this section can be met using one or more of the related patterns in the next section. This provides a set of comprehensive high level requirements for establishing and maintaining a TMI, as well as the logical plan to meet the requirement.

3.1 Core Functions

The Core functions use TCG technology and other appropriate industry standards to describe the foundational relationship between the various components in a trusted computing domain and how they interact. The core functions are:

- Establish a Trusted Context in which information can be exchanged between parties
- Exchange Information between parties within the trusted context
- Enforce Policy using the integrity measurements, assertions and attestations exchanged between parties

With these functional primitives in place, a consumer trusted systems domain can validate the ability of an environment provider to enforce separation and operational policy within a cloud or shared infrastructure context. In terms of context – "separation" means that the services, systems and data that comprise a trusted security domain are completely separate from other trusted security domains within the cloud so that only by explicit allowances in operational policy from both trusted security domains can one domain even be aware of another domain. This separation occurs as a logical construct.

3.1.1 Establish a Trusted Context

Probably the most fundamental of the core functions, the requirement to establish a trusted context in which to create and operate a systems domain ensures a basic understanding of the identity and compliance levels of the device and operational parties involved. A trusted context involves gathering a few key artifacts that represent the trusted state of a trust domain; a unique and verifiable identity for the device or party, a statement of compliance, the information necessary for policy resolution, and an Attestation Key that is used to sign information in communication with the device or party. In addition to the Attestation Key, it may also be desirable to generate an Encryption Key. It is recognized that it is bad practice to both sign and encrypt messages using the same key. While the nature of keys generated is necessarily aligned to the standard or protocol to which the pattern is mapped, it is also recommended that protocols are selected that operate in accordance with recognized best practice.

NOTE: While the name of the attestation key is similar to the TPM Attestation Identity Key (AIK), its function within this context is to logically describe the key that signs attestations of state, policy or other information exchanged between parties in a TMI.

3.1.1.1 All active participants in a trusted multi-tenant environment should establish a trusted context within which interactions occur.

The intent is to generate an understanding of the degree to which one party will trust, or rely upon, the information provided by another party. A trusted context is established when the various parties who are interacting with or managing a TMI environment have implemented processes, controls and protocols for assuring the confidentiality, integrity, availability and auditability of the environments and the messages they send and receive. The trust can be through direct exchange of identity assertions or through a trusted third party.

Among the elements of a trusted context are:

- the ability to assure that messages sent and received are not tampered with or intercepted
- The ability to measure the integrity of assets or processes within the TMI
- The ability to support non-repudiation

Users who do not have an ability to exert control over the provider or consumer resources in a TMI can be trusted parties. If they are not trusted parties, their interactions should be monitored to ensure that the trusted state of the environment is not compromised.

3.1.1.2 The provider and consumer Domain management agents should each establish and maintain a Trusted Entity Store (TES) to record information about the trust relationships with each other and any other party or asset with which they interact

Once a trusted context has been established with a device or party, the context information about that entity should be maintained to allow future communications. The TES is the authoritative repository of information about assets and operators with which a trusted context has been established in a trusted systems domain. The store contains information about the identity credential, attestation keys, compliance statements and policy store location for each asset or operator. This information might be appropriate to store in the TNC MAP, for example, as state and event measurement information is collected on an asset.

To facilitate the requirement to establish a trusted context and exchange information within that trusted context data is collected about assets and parties. The information is initially collected as the assets or parties are added to the trusted systems domain, and then may be updated as needed based on domain policy. The TES can be used to:

- Identify all entities within the trusted systems domain for broadcast communication
- Identify eligible parties for targeted messages
- Identify the capabilities and level of compliance of parties within the trusted systems domain
- Hold credentials or other tokens necessary to encrypt or sign messages to another party

- Resolve policy statements requiring attributes about parties or assets
- And other functions of this type

It would be very inefficient to have to re-establish trust every time there is a need for interaction, so the TES serves as a repository or cache for the information necessary to operate a TMI.

3.1.2 Information Flow between Trusted Parties

Once a trusted context has been established and information about the assets and parties is available, then it is possible for the assets within the Trusted Systems Domain (tenant organization) to communicate with each other. Parties utilize the credentials and measurements of the trusted context to verify the integrity and source/destination of messages. Parties may also encrypt content to protect integrity or the messages. The measurements and assertions of policy compliance allow decisions on the degree of trust placed in the parties in a transaction, supporting trustworthy execution in a multi-tenant, multi-provider environment.

The flow of information between participants in a trusted context within a shared environment where knowledge of other tenants sharing the same infrastructure may be fluid and difficult to ascertain causes a certain amount of healthy paranoia. The intent of the patterns in this section is to ensure that communication only flows between entities that have been measured and identified as participants in the trusted systems domain. Where prior trust does not exist, or privacy on behalf of one or more parties should be maintained, a brokered pattern is defined that can place a trusted 3rd party within an information flow. The broker can serve as an intermediary for establishing trust, within the communication flow, or both depending upon whether the requirement is to establish a trusted context or to serve as a communications proxy.

The information flow patterns are a key part of the core functionality of a Trusted Multi-Tenant Infrastructure, as they allow trusted information flow between the assets and operating parties of the TMI. This forms the basis for separation between tenants.

3.1.2.1 Information flow between trusted parties should occur within a trusted context

In order to maintain the trusted relationship between the key parties in a TMI, the environment provider and the consumer domain owner, it is critical that all information flows that could affect the state of the overall environment be conducted using the trusted context that has been established. If one tenant in an environment were to make back channel changes, then the other tenants would have cause to question the trustworthiness of the assets they were using within their own domain. Conformance to this requirement preserves the confidentiality, integrity, availability and auditability of events and changes within the environment. It is also fundamental to establishing and managing separation between tenants in a multi-tenant environment. When a trusted context is established, there is an exchange of keys that can protect and support separation between information flows using shared infrastructure.

3.1.2.2 The integrity of the information flow between trusted parties should be assured

The use of the trusted context for information flow between trusted parties provides the environmental conditions under which trust can be maintained. It also provides the tools to ensure that the information sent by one party is the same as what is received by the intended recipient. This requirement to assure data integrity ensures that the *content* of a flow can be trusted. Confidentiality, availability and auditability of information may be critical policies enforced within some domains, but integrity should always be maintained, therefore it is a normative requirement. The ability to rely on the information flow helps to ensure that providers and consumers of TMI assets can act as they would if the infrastructure was local to a dedicated environment. This requirement also restricts the types of communications protocols that can be implemented within a TMI. Protocols that do not assure the integrity of the information transferred are not supported. The use of signed and/or encrypted payloads may be used to increase the reliability of protocols, but the integrity of the information flow between entities in a TMI is critical.

3.1.3 Determine, Validate and Enforce Policies

A Trusted Systems Domain is a logical construct that is intended to serve the needs of the owner and stakeholders of the domain. These consumers use services from one or more provider environments. In many cases, the provider environments, especially those delivered as a shared service among a wide range of consumer organizations, tend to have a fairly fixed set of services governed by terms and conditions for their use. These T&C provide the foundation for the provider policy that all consumers should adhere to. Each tenant of the provider environment is doing so in the context of a particular business or mission need. Whether the provider represents IT services within the same organization or services provided to a large community the requirements and policies of the consumer should be defined and reconciled with the policies of the provider.

Each party, provider and consumer, should be able to clearly define, measure, monitor and enforce compliance with their policies. There may be more than 2 parties involved in managing policy compliance. For example, there may be a broker serving as an intermediary between 2 or more parties. There may be multiple consumers within a shared trusted systems domain. There may be multiple providers with resources allocated in support of a consumer's trusted systems domain.

Key functionality includes:

Policy Determination. A policy is, in essence, a conditional expression followed by one or more declarative statements – essentially an if-then-else construct. This is generally populated with one or more attribute variables from a pre-defined dictionary of terms. Each of these variable terms is bound to a mechanism to resolve the value appropriate to the policy statement execution context. Policy definition also includes the rules for combining multiple policy statements into a combined rule or decision hierarchy, so that the resulting decisions will be unambiguous.

Policy Validation. Once the policy has been defined and the rules for resolution of ambiguity are defined, the state of compliance should be tested. Within the trusted systems domain compliance validation could be driven by events, timed intervals or on request. Within the patterns in the TMI Reference Model, there are many references to policy validation. This assures that the actions taken do not compromise the integrity of the trusted systems domain. Policy compliance is tested using a Policy Decision Point (PDP). The PDP is responsible for resolution of the policy statements into an executable rule, the resolution of variables (attributes) using the Policy Information Point (PIP) and the execution of the policy rule. A decision can be pass, fail or pass with obligations. An obligation is an additional step that should be taken in policy enforcement.

Policy Enforcement. The primary controller of policy within a trusted systems domain is a Policy Management Controller (PMC). This component serves as a controller for interaction between the PDP, Policy Information Point (PIP) to resolve attribute values and the Policy Enforcement Point (PEP) to act on the decision. The PMC is responsible to determine, from information in the Trusted Entity Store, which PDP's need to be engaged in the resolution of policy within the context at hand. It determines the entities involved and determines the proper combination of PDP and PEP to engage. Once a policy decision has been reached, the PEP takes the necessary action, based on the policy, in response to the policy decision.

The Policy Management patterns form the last element of the core functionality of the TMI Reference Model. All other functionality is dependent on the trusted context and compliance enforcement provided by policy enforcement capabilities within a trusted context.

3.1.3.1 Domain owners should define, manage and assure the integrity of the policies in the domain policy store.

The intent is to generate an understanding of the degree to which each party will define and manage their policies within the TMI environment. All providers and consumers should define and manage their specific domain and environment policies. Providers and consumers may leverage a trusted third party to conduct policy management.

Among the elements of a defining and managing policy are:

- The ability to assure that messages sent and received are in accordance with the domain owners policies.
- The ability to allow the domain owner the ability to update and reconfigure their domain policy to maintain compliance with policy changes.

Users who do not have an ability to exert control over the provider or consumer resources in a TMI can be trusted parties. If they are not trusted parties, their interactions should be monitored to ensure that their actions are in compliance with the defined domain policy.

3.1.3.2 Policy interaction within and between trusted systems domains should use Trusted Information Flows

The intent is to generate an understanding of the degree to which parties within the TMI environment interact with each other's policies in a trusted fashion. All providers and consumers should utilize trusted information flow when conducting policy references. Providers and consumers may leverage a trusted third party to interact with their policies via a trusted information flow. Utilization of trusted information flow maintains confidentiality, integrity, and accountability of parties interfacing with domain policies.

Among the elements of a trusted policy interface are:

- The ability to assure that messages received by the parties interacting with the domain policy are permitted.
- The ability to allow the domain owner the ability to verify the integrity of parties interfacing with their policy.

Users who do not have an ability to exert control over the provider or consumer resources in a TMI can be trusted parties. If they are not trusted parties, their interactions should be monitored to ensure that their actions are in compliance with the defined domain policy.

3.1.3.3 Policy decisions should be controlled by the owners of the policy.

The intent is to generate an understanding of the degree to which the owner controls the ability to make policy decision on their policy. All providers and consumers should control policy decisions on their own policy. Providers and consumers may leverage a trusted third party to interact with their policies to make policy decisions via a trusted information flow.

Among the elements of a controlled policy decisions are:

- The ability to assure that policy decision is only executed by the policy owner in a trusted fashion.
- The policy owner should appropriately prioritize a variety of policy sets and construct policy hierarchies that maintain compliance across all policy sets.

Users who do not have an ability to exert control over the provider or consumer resources in a TMI can be trusted parties. If they are not trusted parties, their interactions should be monitored to ensure that their actions are in compliance with the defined domain policy.

3.1.3.4 Policy decisions should be enforced by the owner of the protected resource and should include and implement valid policy decisions from all stakeholders

The intent is to generate an understanding of the degree to which the owner provides proper access controls to enforce policy to ensure compliance. All providers and consumers should enforce policy of protected resources and implement policy decisions from all stakeholders. Providers and consumers may leverage a trusted third party to enforce their policies and make policy decisions via a trusted information flow.

Among the elements of a controlled policy decisions are:

- The policy owner should properly configure policy to make decisions that account for all stakeholders and maintains policy compliance within their domain
- Protected resources should have policy enforcement controls that are maintained by the policy owner to maintain compliance.

Users who do not have an ability to exert control over the provider or consumer resources in a TMI can be trusted parties. If they are not trusted parties, their interactions should be monitored to ensure that their actions are in compliance with the defined domain policy.

3.2 Management Services

Management Services use TCG Technology and other appropriate industry standards to describe the foundational relationship between the various components in a trusted Multi-tenant infrastructure (TMI) and how they are managed. The ability to manage configuration of services, proactively monitoring assets, reporting compliance, and responding to events/audits provide the main implementation focus for Management Services within a cloud or share infrastructure environment.

A consumer can manage assets within the trusted systems domain environment against defined policies and a provider can manage the provider environment as well as the various consumer domains within a cloud or shared infrastructure. In terms of context – “management” means the ability to perform administrative functions against assets within the Consumer trusted systems domain and Provider environment in order to achieve and maintain policy compliance.

3.2.1 Monitoring Services

3.2.1.1 Parties should establish a Management Service that monitors asset state and events within a Trusted Multi-tenant Infrastructure.

The intent is to monitor state and events within the TMI.

It is important for both providers and consumers within a multi-tenant environment to be able to maintain awareness of the state of assets within a domain as well as monitor and detect changes in state as they occur to maintain trust in the environment and level of compliance.

It is also important to be able to monitor events within the domain that may indicate a need to respond.

The Monitoring Repository serves as a Policy Information Point (PIP) while the Policy Store just contains policy statements.

3.2.2 Management/Control Services

3.2.2.1 Each domain should establish a Management Control Service that provides reporting, service initiation/decommission, asset adjustment, monitoring and management of assets within their domain

The intent is to generate an understanding of the management, service initiation/decommission, asset control, configuration and monitoring service aspects of the components within the TMI.

3.2.3 Reporting Services

3.2.3.1 Each domain should establish a Management Service that provides reporting of service events/audits/state within their domain.

The intent is to generate an understanding of the reporting service components within the TMI.

3.2.4 Audit Services

3.2.4.1 Each domain should establish a Management Service that provides audit mechanisms to record policy decisions and actions.

The intent is to generate an understanding of the audit service components within the TMI.

3.2.4.2 Each domain should establish a Management Service that evaluates audited decisions and actions and triggers events when non-compliance is detected

The intent is to generate an understanding of the audit service components within the TMI.

3.3 Provisioning Services

Provisioning is a fundamental function within Trusted Multitenant Infrastructure. Provisioning is used to create, change, or destroy resources. The provisioning agent acts on behalf of the requestor. The provisioning agent may be acquiring or acting on a resource or set of resources. If there is a policy store associated with an item, there should be policy allowing the request in the policy store or the request will fail. For every request the credentials of the requestor should be validated.

A consumer can provision assets for a trusted systems domain and define policies that govern the use and acquisition of assets. Providers manage their environments as well as the various consumer domains within a cloud or shared infrastructure. By environment we mean the infrastructure they use and the assets that they make available to consumers. By management we mean the ability to perform administrative functions against assets within the Consumer trusted systems domain and Provider environment in order to achieve and maintain policy compliance.

3.3.1.1 All Provisioning requests should be on Trusted Information Flows

The intent is to assure that provisioning requests originate with an authorized consumer and are received by the provider. No information leakage should occur in these transactions

3.3.1.2 The Trusted Systems domain should store (or maintain) information about resources that it has control over in its Trusted Entity Store.

Resiliency of the Trusted Systems Domain is a critical feature that should be supported. We do not in this document try to tell implementers how to design for resiliency. However, we expect the Trusted Entity Store to be highly available, resilient and recoverable. Consequently maintaining asset control information in this store increases the resiliency of the Trusted Systems Domain.

3.3.1.3 Providers of assets should store (or maintain) information about the assets they manage in their Trusted Entity Store.

Resiliency of the Trusted Systems Domain is a critical feature that should be supported. We do not in this document try to tell implementers how to design for resiliency. However, we expect the Trusted Entity Store to be highly available, resilient and recoverable. Consequently asset providers should support these features. Maintaining asset control information in a Trusted Entity Store increases the resiliency of the Trusted Systems Domain.

3.3.1.4 Provisioning Actions should be logged and auditable

It should be possible to confirm and trace the provisioning actions independent of any request for monitoring or logging from a consumer. The use of assets within a Trusted Systems Domain will be the basis for financial interactions as well as a driver of policy. Therefore all of this activity should be logged and auditable. By auditable we mean that it should both be examinable by an independent third party and available for consumer audit requests.

3.3.1.5 The log of provisioning Actions should be traceable in the Trusted Entity Store

Resiliency of the Trusted Systems Domain is a critical feature that should be supported. We do not in this document try to tell implementers how to design for resiliency. However, we expect the Trusted Entity Store (TES) to be highly available, resilient and recoverable. Consequently maintaining logs of provisioning actions in the Trusted Entity Store increases the resiliency of the Trusted Systems Domain.

Each asset that is or has been provisioned, deprovisioned or configured within a Trusted Systems Domain should have established a trusted context, therefore should be present in the Trusted Entity Store. This does not replace the CMDB, although a viable design option may be that the TES and CMDB overlap.

TCG Trusted Multi-Tenant Implementation Patterns

The implementation patterns are the second of two linked parts of the TMI Reference Framework. Each of the requirements in the previous section can be met using one or more of the related patterns in this section. This provides a set of comprehensive high level requirements for establishing and maintaining a TMI, as well as the logical plan to meet the requirement.

3.4 Core Functions

The Core functions use TCG technology and other appropriate industry standards to describe the foundational relationship between the various components in a trusted computing domain and how they interact. The core functions are:

- Establish a Trusted Context in which information can flow between parties
- Flow Information between parties within the trusted context
- Enforce Policy using the integrity measurements, assertions and attestations exchanged between parties

With these functional primitives in place, a consumer trusted systems domain can validate the ability of an environment provider to enforce separation and operational policy within a cloud or shared infrastructure context. In terms of context – "separation" means that the services, systems and data that comprise a trusted security domain are completely separate from other trusted security domains within the cloud so that only by explicit allowances in operational policy from both trusted security domains can one domain even be aware of another domain. This separation occurs as a logical construct.

3.4.1 Establish a Trusted Context

Probably the most fundamental of the core functions, the requirement to establish a trusted context in which to create and operate a systems domain ensures a basic understanding of the identity and compliance levels of the device and operational parties involved. A trusted context involves gathering a few key artifacts that represent the trusted state of a trust domain; a unique and verifiable identity for the device or party, a statement of compliance, the information necessary for policy resolution, and an Attestation Key that is used to sign information in communication with the device or party. In addition to the attestation key, it may also be desirable to generate an Encryption Key. It is recognized that it is bad practice to both sign and encrypt messages using the same key. While the nature of keys generated is necessarily aligned to the standard or protocol to which the pattern is mapped, it is also recommended that protocols are selected that operate in accordance with recognized best practice.

NOTE: While the name of the attestation key is similar to the TPM Attestation Identity Key (AIK), its function within this context is to logically describe the key that signs attestations of state, policy or other information exchanged between parties in a TMI.
The protocol used by these patterns is independent of transport or delivery mechanism. It is anticipated that existing communications, messaging and remote procedure call infrastructures can be leveraged to transport attestation messages.

The patterns make reference to “appropriate steps to protect the integrity of the data”. Because this pattern can be implemented using a number of standards and protocols, the specific measures are not identified here. Examples of appropriate measures might include generation of a hash to protect the content, a nonce to prevent replay attacks, data encryption to protect the confidentiality of the data or other schemes. The message content should include at a minimum the identity and measurements of the asset, such that the measurement is linked to the asset or party and not subject to random recombination of identities and measurements. While the specific means are not called out, an implementer should take measures to protect the integrity or the data.

3.4.1.1 Platform Attestation

Synopsis

Platform Attestation is the process of establishing trust in an asset within the environment. It is based upon hardware platform measurement and attestation of the platform asset. A platform can attest to its description of platform characteristics that affect the integrity (trustworthiness) of the asset. It is important to recognize that a platform asset may be a physical or virtual device or connection. Where possible the root of trust for a virtual asset should be bound to the underlying physical asset to enable full integrity attestation. All forms of attestation require reliable evidence of the attesting entity.

Platform Attestation involves 2 key elements: attestation of the platform and authentication of the platform.

Attestation of the platform is an operation that provides proof of a set of the platform’s integrity measurements. The measurements may be based on information known to the platform, measurements taken of the platform by an external agent, or both. This is done by digitally signing the integrity measurement data using an attestation key. The acceptance and validity of both the operational measurements and the attestation key itself are determined by a challenger’s verifier. The Attestation Key is obtained using either a trusted Credential Authority or via a trusted attestation protocol. If the asset has a TPM or vTPM, the actual measurements may be signed by the platform AIK.

Authentication of the platform provides evidence of a claimed platform identity. The claimed identity reflects a unique identity for the platform asset and may or may not be related to a user or any actions performed by a user. The acceptance and validity of the credential itself are determined by a challenger’s verifier. The credential is obtained using either a trusted Credential Authority or via a trusted attestation protocol.

Context

In order to operate in a trusted multi-tenant environment, trust should be established between parties. This pattern describes establishment of trust in the platform assets within the environment and the ability to attest to the integrity and the state of the platform asset to establish a trusted baseline for the asset to be used within a domain. This pattern uses hardware capabilities based in the systems' root of trust to establish a trusted context for attestation of integrity measurements of the state of a platform asset.

Systems' roots of trust are components that should be trusted because misbehavior might not be detected. A complete set of Roots of Trust has at least the minimum functionality necessary to describe the platform characteristics that affect the trustworthiness of the platform.

According to the TCG, there are commonly three Roots of Trust in a trusted platform; a root of trust for measurement (RTM), root of trust for storage (RTS) and root of trust for reporting (RTR). The RTM is a computing engine capable of making inherently reliable integrity measurements, typically the normal platform computing engine, controlled by the core root of trust for measurement (CRTM). The CRTM is the instructions executed by the platform when it acts as the RTM. The RTM is also the root of the chain of transitive trust. The RTS is a computing engine capable of maintaining an accurate summary of values of integrity digests and the sequence of digests. The RTR is a computing engine capable of reliably reporting information held by the RTS. [TPM Architecture v1.4]

In deriving this pattern from the TMI Use Cases, a ***challenger*** could be either a provider or consumers management agent. The ***platform*** could be either the providers' or consumers' assets. A ***platform asset*** in this case could be either a physical or virtual asset. In the case of a virtual asset, one of the integrity measurements that could be requested by a ***challenger*** is a manifest that describes the chain of trust back to the underlying physical asset.

Selection Criteria

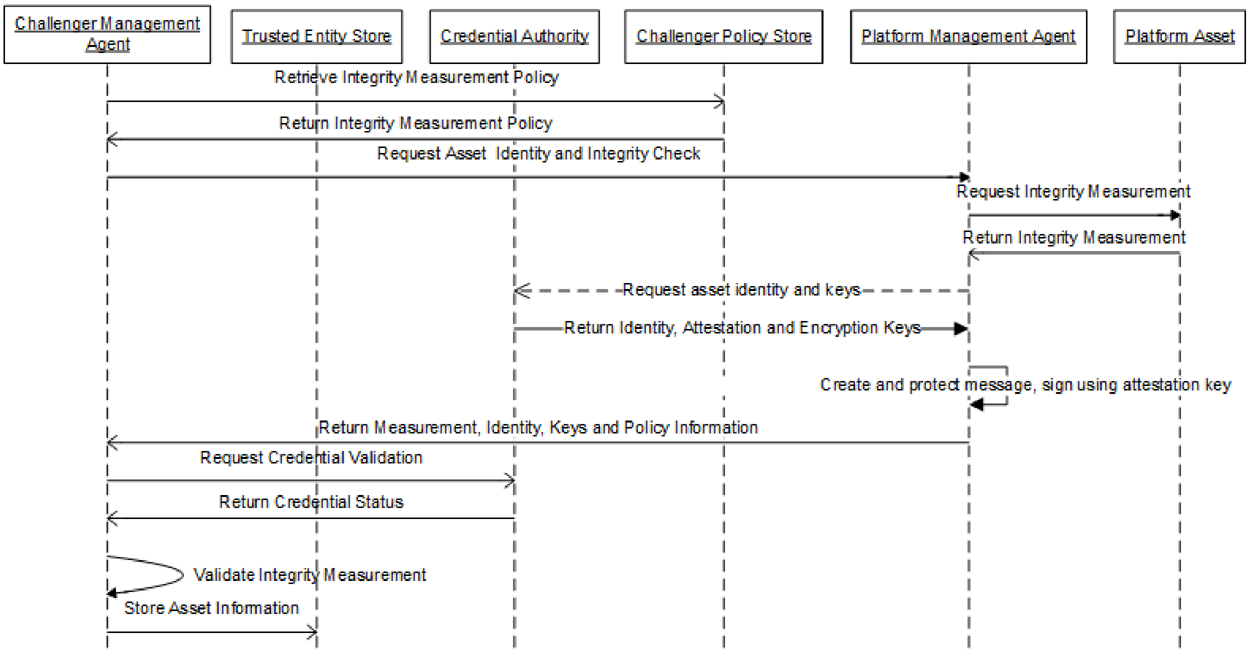
Platform attestation can be selected when the physical assets are equipped with Trusted Platform Modules and the Credential Authority for the attestation key is trusted by both parties. This pattern establishes the trusted context for the flow of information about an asset based on a hardware root of trust.

Solution

Platform attestation consists of several steps:

1. A ***Challenger*** **Management Agent** retrieves policy for integrity measurement types needed from the ***Challenger*** **Policy Store**.
2. A ***Challenger*** **Management Agent** requests Asset Identity and Integrity Measurements from a ***Platform*** **Management Agent**.
3. A ***Platform*** **Management Agent** collects integrity measurement data
4. The ***Platform*** **Management Agent** collects the identity credentials, attestation key and encryption key for the asset.
5. The ***Platform*** **Management Agent** identifies the Policy Management Controller information for policy decisions and enforcement for the asset.

6. The ***Platform* Management Agent** creates a message containing the information to be returned and signs the message using the ***Platform* Management Agent Attestation Key** and takes appropriate steps to protect the integrity of the message.
7. The protected integrity measurement data, keys and device credentials are returned to the ***Challenger* Management Agent**.
8. The ***Challenger* Management Agent** verifies the request. The integrity measurement is verified to ensure it matches the data sent by the ***Platform* Management Agent**. The device credentials are evaluated and signatures validated.
9. The device identity, keys, policy enforcement information and measurements are stored in the **Trusted Entity Store** for the ***Challenger*** domain.


Challenger Management Agent
Trusted Entity Store
Credential Authority
Challenger Policy Store
Platform Management Agent
Platform Asset
Retrieve Integrity Measurement Policy
Return Integrity Measurement Policy
Request Asset Identity and Integrity Check
Request Integrity Measurement
Return Integrity Measurement
Request asset identity and keys
Return Identity, Attestation and Encryption Keys
Create and protect message, sign using attestation key
Return Measurement, Identity, Keys and Policy Information
Request Credential Validation
Return Credential Status
Validate Integrity Measurement
Store Asset Information

Implications

The trust relationship is based on certification by and attestations from the platform agent and it is the use of trusted platform assets to collect and store the measurements that provides the context. This pattern does not in and of itself guarantee the measurements or assertions made by the asset.

This pattern establishes trust by verification of the integrity and identity of individual assets within the TMI. This provides a basic context for evaluation of the degree to which assertions made by the asset can be trusted.

Related Requirements

Platform Attestation is one possible implementation of the requirement (1.1.1.1) to establish a trusted context. As one of the core functions underlying the TMI framework, the requirement is a pre-requisite to establishment of a TMI compliant trusted multi-tenant infrastructure.

Related Patterns

Platform Attestation is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for establishing a trusted context is mandatory for TMI compliance and along with the patterns for Information Flow and Policy enforcement form the core of the TMI pattern library.

Related Use Cases

Platform Attestation is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for establishing a trusted context is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

3.4.1.2 Operator Certification Based Trust

Synopsis

Operator Certification Based Trust is the process of establishing trust between operational parties based on operational policy and procedural compliance attestation. The trust is implemented through the use of trusted credentials to sign and/or encrypt attestations and information flow between entities. Parties establish this trust based on direct knowledge or the reputation of the other party. Operating entities within an environment can attest identities of the parties, policies, certifications, compliance measurements and operational practices (SLA). All forms of attestation require reliable evidence of the attesting party.

Operator Certification Based Trust can be understood along several dimensions, attestation by the operator and authentication of the operator.

Attestation by the operator is an operation that provides claims of policies, practices, compliance and other information by the operating party. This may also include operational measurements taken by the operator or an external agent. Attestation is made by digitally signing specific operator measurement data using an attestation key. The acceptance and validity of both the operational measurements and the attestation key itself are determined by a challenger's verifier. The attestation key is obtained using either a trusted Credential Authority or via a trusted attestation protocol.

Authentication of the operator provides evidence of a claimed party identity. The claimed identity may or may not be related to a user or any actions performed by the user. The acceptance and validity of the credential itself are determined by a challenger's verifier. The credential is obtained using either a trusted Credential Authority or via a trusted attestation protocol. Certified keys (i.e. signed by an Attestation Key) have the added semantic of being attestable.

Context

In order to operate in a trusted multi-tenant environment, trust should be established between parties. Operator Certification describes establishment of trust using credential based capabilities to verify the identity of an operating party and then attestation of policy statements to establish a trusted context between parties operating within a domain. This pattern uses trusted certificates from a trusted credential authority to establish a trusted context for attestation of operational policy and measurements of the behavior of a platform asset or environment.

In deriving this pattern from the TMI Use Cases, a ***challenger*** could be either a provider or consumer management agent. The ***operator*** is the party with whom the challenger has a direct relationship. The ***operator*** may or may not be the owner of the assets that are provided to a ***challenger***. One of the compliance statements a ***challenger*** may request of an ***operator*** is a manifest that defines the chain of accountability for entities and compliance statements back to the owner of the assets. This chain of accountability would be common when dealing with service brokers or OEM relationships between the service offeror and the asset owner. The objective is to a) understand the compliance of the chain of operators with the consumer's policy and b) establish identity and Attestation Key credentials to use for trusted communication between the ***challenger management agent*** and the ***operator*** or their ***management agent.***

Selection Criteria

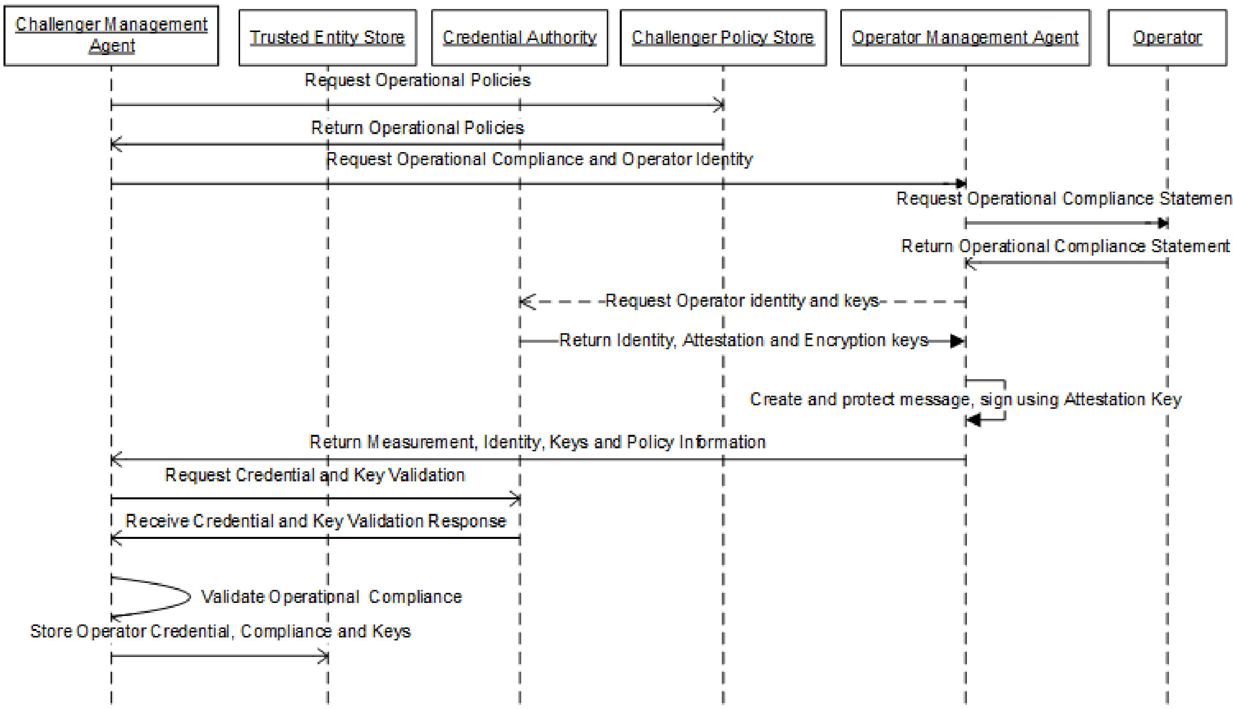
Operator Certification Based Trust is used to certify the identity of an operating party and establish the certificates and keys used to sign messages between the parties. Operator certification can be selected when the parties operating within the TMI environment are able to rely on knowledge of the reputation of the other party and the Trusted Credential Authority for the Attestation Key is trusted by both parties. The degree to which the parties are aware of each other prior to establishing a trusted context is not the key factor in selection of this pattern. It can be used by parties previously unknown to each other, such as establishing the ability to interact with a customer you have done business with or to establish the context for interbank transfers. In other words, the degree of trust is based on reputation and other factors, and should be used by the parties to determine what information can be safely exchanged. This pattern establishes the trusted context for the flow of information based on proper signing or encryption using credentials issues by a Trusted Credential Authority.

This pattern may be selected when it is necessary to collect hardware integrity measurements from platform assets which do not support hardware based attestation (i.e. no TPM) or between operational entities exchanging information not rooted in a hardware root of trust (i.e. operational practices, certification or events). This operator certification of platform assets instead of direct measurements, while not meeting the same level of Assurance of a direct measurement, allows additional flexibility in use of platform assets.

Solution

The Operator Certification protocol consists of several steps:

1. A ***Challenger* Management Agent** retrieves policy for operational policies needed from the ***Challenger* Policy Store**.
2. A ***Challenger* Management Agent** requests certification of one or more statements of operational policy compliance from the operator.
3. The ***Operator* Management Agent** collects operational certification data. The ***Operator* Management Agent** collects the identity credentials, Attestation Key and Encryption Key for the operator.
4. The ***Operator* Management Agent** identifies the Policy Management Controller information for policy decisions and enforcement for the operator.
5. The ***Operator* Management Agent** creates a message containing the information to be returned and signs the message using the ***Operator* Management Agent** Attestation Key and takes appropriate steps to protect the integrity of the message.
6. The protected operational data, keys and operator credentials are returned to the ***Challenger* Management Agent**.
7. The ***Challenger* Management Agent** verifies the request. The integrity measurement is verified to ensure it matches the data sent by the ***Operator* Management Agent.** The operating party's credentials are validated against the **Credential Authority** and signatures validated
8. The operator identity, keys, policy enforcement information and measurements are stored in the **Trusted Entity Store** for the ***Challenger*** domain


Challenger Management Agent
Trusted Entity Store
Credential Authority
Challenger Policy Store
Operator Management Agent
Operator
Request Operational Policies
Return Operational Policies
Request Operational Compliance and Operator Identity
Request Operational Compliance Statement
Return Operational Compliance Statement
Request Operator identity and keys
Return Identity, Attestation and Encryption keys
Create and protect message, sign using Attestation Key
Return Measurement, Identity, Keys and Policy Information
Request Credential and Key Validation
Receive Credential and Key Validation Response
Validate Operational Compliance
Store Operator Credential, Compliance and Keys

Implications

The trust relationship is based on certification by and attestations from the operating entities and it is the use of trusted credentials and reputation of the parties to collect and store the measurements that provides the context. This pattern does not in and of itself guarantee the measurements or assertions made by the party.

Related Requirements

Operational Certification is one possible implementation of the requirement (1.1.1.1) to establish a trusted context. As one of the core functions underlying the TMI framework, the requirement is a pre-requisite to establishment of a TMI compliant trusted multi-tenant infrastructure.

Related Patterns

Operational Certification is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for establishing a trusted context is mandatory for TMI compliance and along with the patterns for Information Flow and Policy enforcement for the core of the TMI pattern library.

Related Use Cases

Operational Certification is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for establishing a trusted context is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

3.4.1.3 Broker Certification Based Trust

Synopsis

Broker Certification Based Trust is the process of establishing trust based upon the use of Trusted Credentials to sign and/or encrypt attestations and information flow between entities. Parties establish their trust relationship based upon the direct knowledge or certification by a trusted 3rd party, or trust broker. Brokering agents can attest identities of the parties, policies, certifications, compliance measurements and operational practices (SLA). All forms of attestation require reliable evidence of the attesting party. Broker Certification based Trust encapsulates other patterns for establishing a trusted context, serving as an intermediary or proxy for the primary pattern.

Broker Certification Based Trust can be understood along several dimensions, Attestation of the Broker, Attestation to the challenger, attestation to the challenged party and authentication of the parties.

Attestation of the Broker is an operation that provides certification that a broker can be trusted to report integrity measurements by providing certification of a set of the Broker's policies, practices and reputation. This is done by digitally signing a set of policy certifications about the Broker using an Attestation Key to both the Challenger and Challenged parties.

Attestation to the Challenger is an operation that provides certification to the Challenger of the Challenged parties identity and compliance. This is performed using the set or subset of the credentials associated with the broker; used to issue an Attestation Key credential on behalf of the Challenged party. The attestation key is assigned by the broker and can be revoked as necessary based on a change in trust status.

Attestation to the Challenged Party is an operation that provides certification to the Challenged party of the Challenger's identity and compliance. This is performed using the set or subset of the credentials associated with the broker; used to issue an Attestation Key credential on behalf of the Challenger. The attestation key is assigned by the broker and can be revoked as necessary based on a change in trust status.

Authentication of the parties provides evidence of a claimed party identity. The claimed identity may or may not be related to a user or any actions performed by the user. Certified keys (i.e. signed by an Attestation Key) have the added semantic of being attestable. The Attestation Key is generated by the broker on behalf of both parties, as the broker is vouchsafing for the trustworthiness of the parties. The credential can be revoked by the broker as necessary based on a change in trust status. The identity credentials can be generated by the Broker if anonymity of one or both parties is desired.

Context

In order to operate in a trusted multi-tenant environment, trust should be established between parties. Broker Certification describes establishment of trust using credential based capabilities to certify the identity of an operating party and then attestation of policy statements to establish a trusted context between parties operating within a domain. This pattern uses trusted credentials from a trusted credential authority to establish a trusted context for attestation of operational policy and measurements of the behavior of a platform asset or environment.

When a broker is used, it is assumed that the other parties do not have a trust relationship appropriate to the context of the Trusted Systems Domain. In some cases, this pattern may be used to protect the identities of one or both parties from disclosure, with the broker serving as a trusted proxy between parties.

Selection Criteria

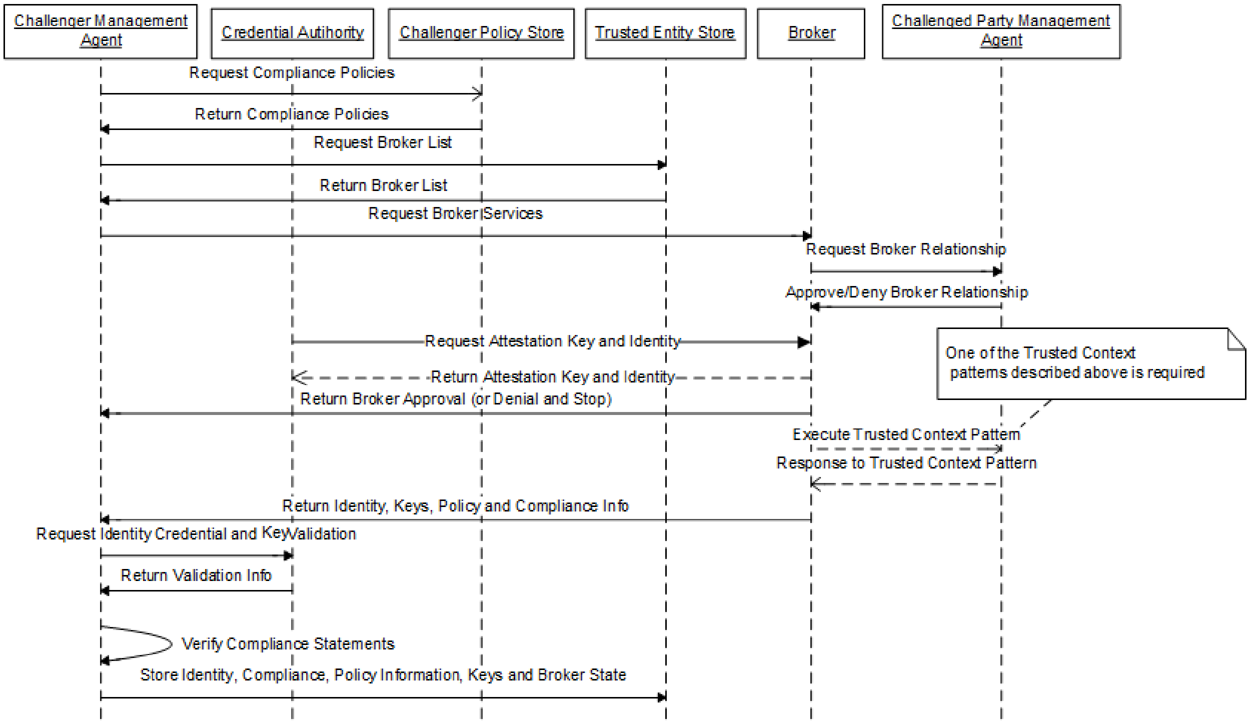
Broker Certification Based Trust is used to certify the identity of an operating party and establish the certificates and keys used to sign messages between the parties. Broker certification can be selected when the parties operating within the TMI environment are able to rely on knowledge of the reputation of a common party (the Broker) and the Trusted Credential Authority for the Attestation Key is trusted by both parties. This pattern establishes the trusted context for the flow of information based on proper signing or encryption using credentials issues by a trusted Credential Authority.

This pattern may be selected when it is necessary to exchange hardware integrity measurements from platform assets which do not support hardware based attestation (i.e. no TPM) or between operational entities exchanging information not rooted in a hardware root of trust (i.e. operational practices, certification or events) and the two parties do not have a direct trust relationship.

Solution

The Broker Certification protocol consists of several steps:

1. The ***Challenger*** **Management Agent** retrieves policy for compliance measurements needed from the ***Challenger*** **Policy Store**.
2. The ***Challenger*** **Management Agent** retrieves a list of brokers from the **Trusted Entity Store.** This presumes that a trusted context has already been established with the **Broker**
3. **The *Challenger Management Agent*** requests services from the **Broker**.
4. The **Broker** requests permission to serve as the broker to the **Challenger** from the ***Challenged Party Management Agent***
5. The ***Challenged Party Management Agent*** approves or rejects the request and the response is returned to the ***Challenger Management Agent***.
6. If the request for broker services is approved, the process continues, otherwise it is terminated. Another broker many be queried or another means for establishing a trusted context may be established.
7. The **Broker** collects the *Parties* identity credentials and generates an Attestation Key for future brokered exchanges of attestations with the *Parties*. The identity credentials can be generated by the **Broker** to protect the privacy of the *Parties***.**
8. The **Broker** serves as the **Challenger** to the **Challenged Party** and the **challenged Party** to the **Challenger** through execution of one of the non-brokered patterns for establishing trusted context.
9. The **Broker** information and role is stored in the **Trusted Entity Store** for both parties along with whether it services as a guarantor or intermediary for communication and policy compliance actions.


Challenger Management Agent
Credential Autihority
Challenger Policy Store
Trusted Entity Store
Broker
Challenged Party Management Agent
Request Compliance Policies
Return Compliance Policies
Request Broker List
Return Broker List
Request Broker Services
Request Broker Relationship
Approve/Deny Broker Relationship
Request Attestation Key and Identity
Return Attestation Key and Identity
Return Broker Approval (or Denial and Stop)
One of the Trusted Context patterns described above is required
Execute Trusted Context Pattern
Response to Trusted Context Pattern
Return Identity, Keys, Policy and Compliance Info
Request Identity Credential and Key Validation
Return Validation Info
Verify Compliance Statements
Store Identity, Compliance, Policy Information, Keys and Broker State

Implications

The trust relationship is based on certification by and attestations from the operating entities and the trust broker and it is the use of trusted credentials and reputation of the parties to collect and store the measurements that provides the context. This pattern does not in and of itself guarantee the measurements or assertions made by the party.

Related Requirements

Broker Certification is one possible implementation of the requirement (1.1.1.1) to establish a trusted context. As one of the core functions underlying the TMI framework, the requirement is a pre-requisite to establishment of a TMI compliant trusted multi-tenant infrastructure.

Related Patterns

Broker Certification is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for establishing a trusted context is mandatory for TMI compliance and along with the patterns for Information Flow and Policy enforcement for the core of the TMI pattern library.

As the requesting and brokered parties should both have established a trusted context with the broker, the Operator Certification Based trust pattern is used.

Related Use Cases

Broker Certification is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for establishing a trusted context is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

Information Flow between Trusted Parties

Probably the most pervasive of the core functions, the requirement for information exchange within a trusted context ensures that controls are in place to protect the confidentiality, integrity and availability of information between parties in a multi-tenant ecosystem. Use of the trusted context involves the use of an Attestation Key to sign and an Encryption Key to optionally encrypt information in communication with the device or party.

The patterns described here reflect abstract types of communication, focused on the constraints and obligations necessary for maintaining separation and trust in the TMI. Each of these patterns may be mapped to one or more standards or protocols for operational implementation. The choice of implementation standard can affect the reliability and policy support. Not all protocols will be compliant with the TMI patterns, so care should be taken to ensure the protocol can be implemented in a manner that supports the constraints of the patterns.

3.4.1.4 Broadcast

Synopsis

The broadcast pattern, in the context of a Trusted Multi-tenant Infrastructure (TMI), is the one way transmission of a message to all eligible receivers within the TMI context. Filters may be used to limit the scope of the broadcast, but in general it is a one way form of communication from a sender to one or more receivers within the trusted context of a TMI.

Context

The broadcast pattern is used to send information when the sender is not expecting a reply. The content is often informational in nature, although it may generate an action to be taken by receivers. What is important is that the receivers can identify the sender as a member of a trusted context. The sender should filter the receivers list to parties or devices within a shared context. For example, a provider may send an information broadcast to all of the consumers using resources within a provider environment. A consumer management agent may send a broadcast to all devices within the trusted systems domain. The use of filters to limit the scope of broadcast messages is highly recommended.

Selection Criteria

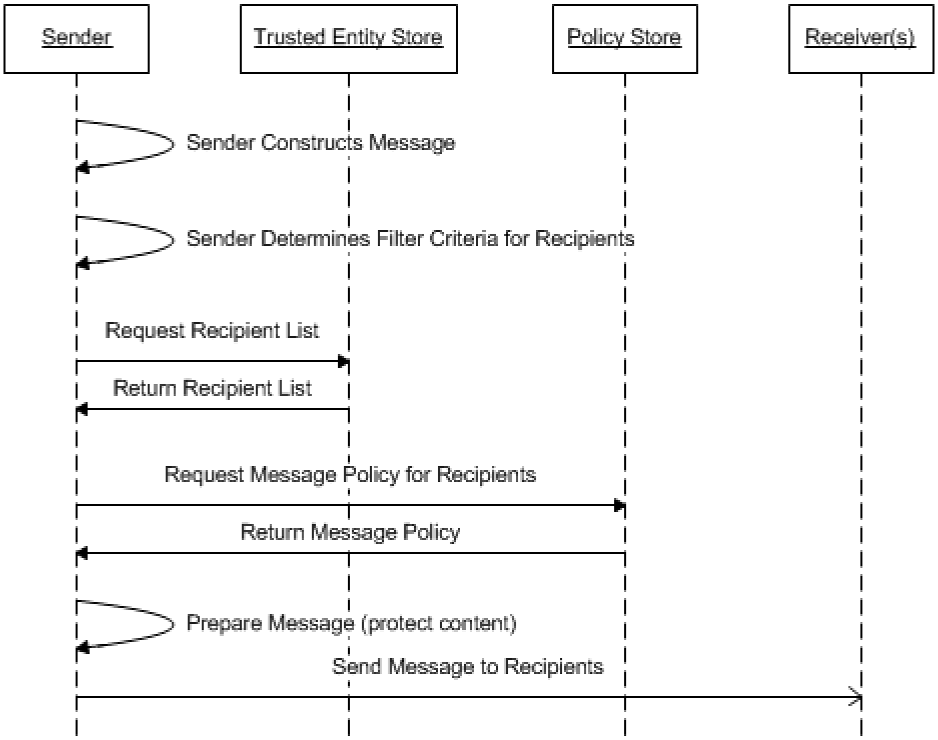
The broadcast pattern is used to send information when the sender is not expecting a reply. The message may be sent to all parties and devices where a trusted context has been established, or it may be sent to a filtered list of receivers. It is not normally used for directing critical actions where acknowledgement or confirmation is required.

Solution

The broadcast pattern consists of the following steps:

1. The **Sender** constructs the message to be sent

2. The **Sender** determines filter criteria to request the list of trusted **Receiver(s)**
3. The **Sender** requests the list of **Receiver(s)** from the **Trusted Entity Store**
4. The **Sender** requests message policy (i.e. encryption required?)
5. The **Sender** should protect the integrity and, if required by policy, the confidentiality of the message
6. The **Sender** should sign the message using the Attestation Key identified for communication with the **Receiver(s)**
7. The **sender** can encrypt the message using the Encryption Key
8. The message is sent to all **Receiver(s)**


Sender
Trusted Entity Store
Policy Store
Receiver(s)
Sender Constructs Message
Sender Determines Filter Criteria for Recipients
Request Recipient List
Return Recipient List
Request Message Policy for Recipients
Return Message Policy
Prepare Message (protect content)
Send Message to Recipients

Implications

The messages sent using the broadcast pattern may or may not be received and acted on by the receiver. As the pattern explicitly precludes a response, there is no way for the sender to verify receipt. The Broadcast pattern may be used to send a message to a single receiver or a group of receivers.

Related Requirements

The broadcast pattern is one method of implementing the requirements regarding exchange of information between trusted parties. The selection of receiver(s) and the signing of the message implement the requirement that information exchange between trusted parties should occur within a trusted context. The creation of the message hash and the optional encryption implement the requirement that the integrity of the information exchanged between trusted parties should be assured

Related Patterns

All of the patterns in the section information Exchange between Trusted Parties address similar problems, and all are dependent on the patterns in the section Establish a Trusted Context.

Related Use Cases

Broadcast is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for information exchange between trusted parties is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

Implementation Standards

While a multicast implementation can be made compliant, many implementations do not support the requirement in this pattern that recipients share a trusted context with the sender.

3.4.1.5 Publish / Subscribe

Synopsis

The publish/subscribe pattern, in the context of a Trusted Multi-tenant Infrastructure (TMI), is the one way transmission of a message to all eligible receivers within the TMI context who have expressed an interest in receiving messages of that type from the publisher. Filters may be used to limit the scope of the broadcast, but in general it is a one way form of communication from a sender to one or more receivers within the trusted context of a TMI.

In order to subscribe to a topic, a Receiver should have established a trusted context with the Sender and have permission to access the topic. Both subscribing to and receiving published messages are late binding activities, ensuring that changes to policy or access controls are appropriately implemented. The management of published topics are abstracted from the sender and receiver through an intermediary role, described as the *Subscription Publisher*. While the Sender and Subscription Publisher can be the same entity, the separation is defined in the pattern to clarify the responsibilities of each role.

Context

The publish/subscribe pattern is used to allow recipients to receive messages on topics in which they have registered an interest. Messages should only be sent to authorized receivers with a valid trusted context. There are several models against which policy may be applied in the publish/subscribe pattern. The most efficient is to apply policy to the ability to subscribe to a topic. Another potential approach is to apply policy to the each message that is sent within a topic. The documentation of this pattern describes policy application at the topic level, but it could be modified to apply policy to message type or individual messages at send time. All of these approaches are valid within a TMI context, it is a deployment choice and thus will not be broken into separate patterns, the requirement is that policy and authorization should be applied at one of these levels.

Selection Criteria

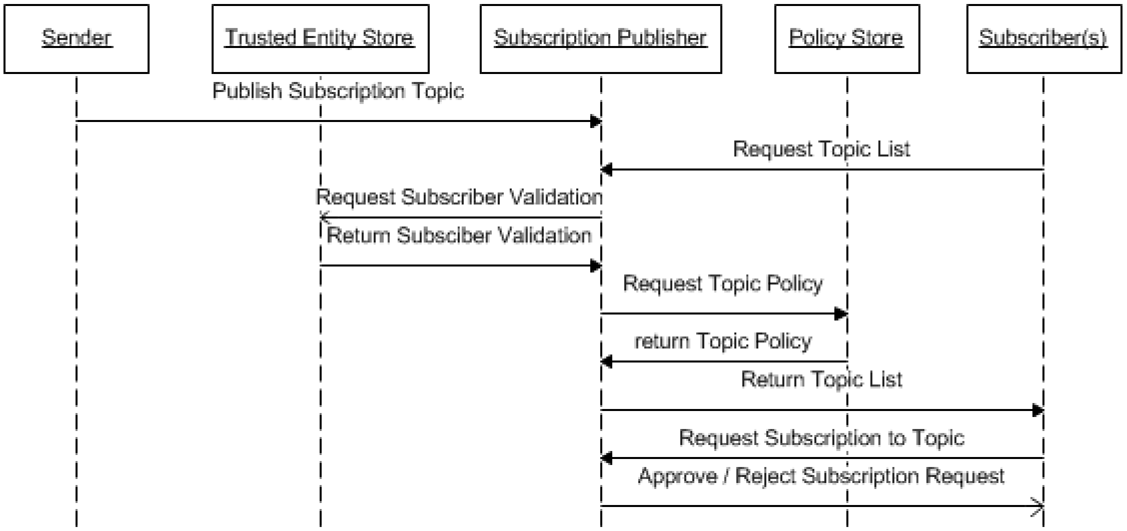
The publish/subscribe pattern is used to allow senders to publish messages to a set of receivers who have expressed an interest in receiving them. The message may be sent to any party or device where a trusted context has been established. Subscriptions may be offered for specific events or topics covering a broad set of events or informational messages. Publish/Subscribe is not normally used for directing critical actions where acknowledgement or confirmation is required.

Solution

The publish/subscribe pattern is broken up into 2 parts, describing subscriptions and publishing separately.

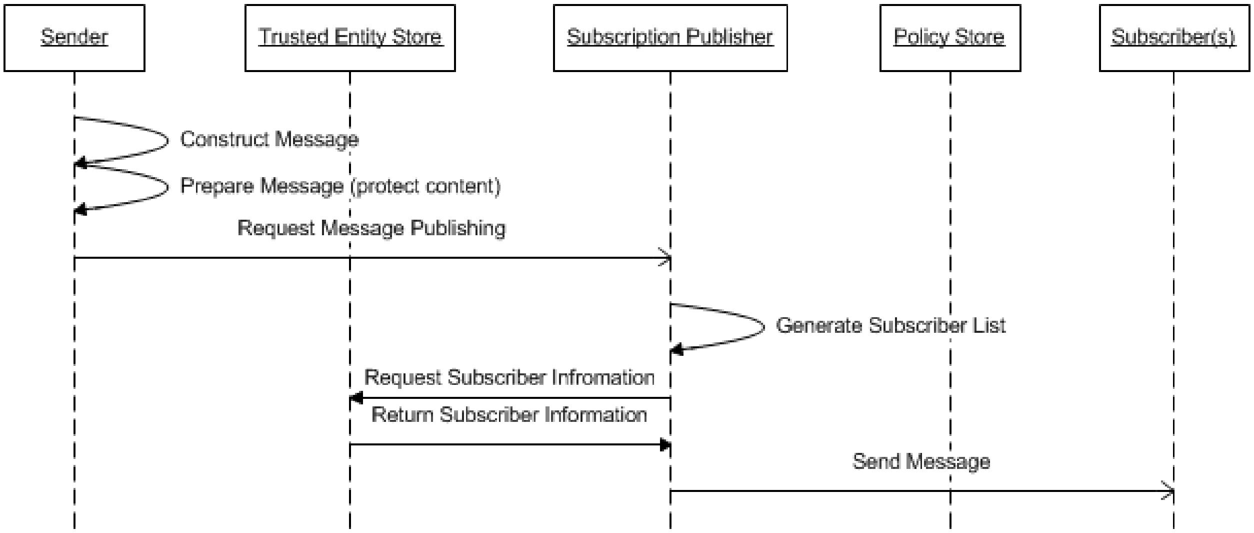
The Subscribe solution consists of the following steps:

1. The ***Sender*** publishes the availability of messages on a topic
2. A **Subscriber** requests a list of topics from the **Subscription Publisher**
3. The **Subscription Publisher** validates the **Subscriber** against the Trusted Entity Store.
4. The **Subscription Publisher** retrieves the topic policy from the **Policy Store.** The **Subscription Publisher** generates a list of topics based on the **Subscriber** and policy
5. A **Subscriber** requests a subscription to a topic
6. The **Subscription Publisher** approves / rejects the subscription request. If approved the **Subscription Publisher** adds the **Subscriber** to the verified subscriber list for the topic. If rejected the **Subscriber** is notified that the request was denied.


Sender
Trusted Entity Store
Subscription Publisher
Policy Store
Subscriber(s)
Publish Subscription Topic
Request Topic List
Request Subscriber Validation
Return Subsciber Validation
Request Topic Policy
return Topic Policy
Return Topic List
Request Subscription to Topic
Approve / Reject Subscription Request

The Publish solution consists of the following steps:

1. The **Sender** constructs the message to be sent
2. The **Sender** should protect the message integrity in accordance with policy
3. The **Sender** should sign the message using its Attestation Key
4. The **Sender** can encrypt the message
5. The **Sender** requests the **Subscription Publisher** send a message to subscribers of a Topic
6. The **Subscription Publisher** accepts or rejects the message based on whether the **Sender** is a verified **Sender** for the Topic
7. The **Subscription Publisher** generates the list of verified **Subscribers**
8. The **Subscription Publisher** requests Subscriber information from the **Sender Trusted Entity Store**
9. The message is sent to all **Subscriber(s)**



Implications

The messages sent using the publish/subscribe pattern may or may not be received and acted on by the receiver. As the pattern does not explicitly require a response, there is no way for the sender to verify receipt. The messages are only sent to those who have explicitly subscribed, so not all affected users may be on the recipient subscriber list.

Related Requirements

The publish/subscribe pattern is one method of implementing the requirements regarding exchange of information between trusted parties. The selection of recipients in step 2 and the signing of the message implement the requirement that information exchange between trusted parties should occur within a trusted context. The creation of the message hash and the optional encryption implement the requirement that the integrity of the information exchanged between trusted parties should be assured.

Related Patterns

All of the patterns in the section information Exchange between Trusted Parties address similar problems, and all are dependent on the patterns in the section establish a Trusted Context.

Related Use Cases

Publish/subscribe is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for information exchange between trusted parties is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

3.4.1.6 Request / Reply

Synopsis

The request/reply pattern, in the context of a Trusted Multi-tenant Infrastructure (TMI), is a conversational transmission of a message and response between a sender and receiver within the TMI context. This pattern represents the primary means of interactive communication between a sender and receiver. Each iteration of the pattern represents a single message and response exchange. The response can be as simple as an acknowledgement of receipt or a question back to the original sender that requires a new response (a second iteration of the send/receive pattern). As the sender is expecting a response, if a reply is not sent by the receiver, policy may dictate a follow up action be taken by the sender.

Context

The request/reply pattern is used to exchange information when the sender is expecting a reply. The content may generate an action to be taken by receivers beyond a simple acknowledgement. What is important is that the senders and receivers can verify each other as a member of a trusted context. The sender only communicates with receiver parties or devices within a shared context. The cardinality between sender and receiver is 1:1. For example, a provider may send a request to a consumer using an asset within a provider environment and the consumer will respond.

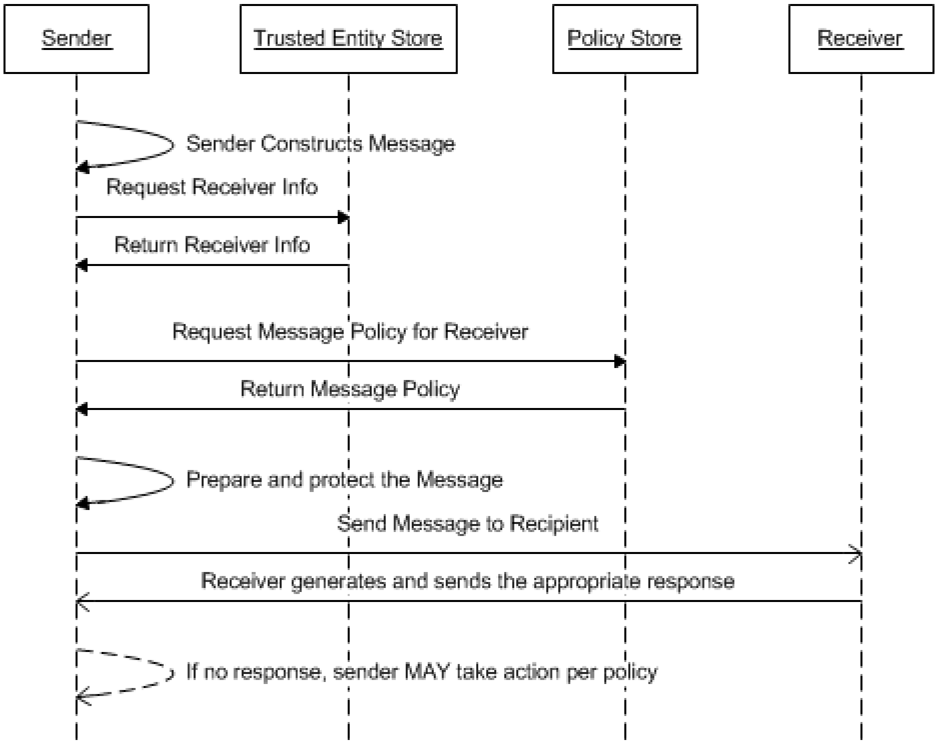
Selection Criteria

The request/reply pattern is used to send information when the sender is expecting a reply. The message may be sent to a party or device where a trusted context has been established. It is often used for directing critical actions where acknowledgement or confirmation is required.

Solution

The request/reply pattern consists of the following steps:

1. The **Sender** constructs the message to be sent
2. The **Sender** retrieves **Receiver** information from the **Trusted Entity Store**
3. The **Sender** validates message policy, including action to take if no reply
4. The **Sender** should take steps to protect the integrity of the message
5. The **Sender** should sign the message using its Attestation Key
6. The **Sender** can encrypt the message
7. The message is sent to the **Receiver**
8. The **Receiver** generates a reply to the message and sends the reply to the **Sender**
9. If no response is received, the **Sender** can take action as dictated by message policy


Sender
Trusted Entity Store
Policy Store
Receiver
Sender Constructs Message
Request Receiver Info
Return Receiver Info
Request Message Policy for Receiver
Return Message Policy
Prepare and protect the Message
Send Message to Recipient
Receiver generates and sends the appropriate response
If no response, sender MAY take action per policy

Implications

The messages sent using the request/reply pattern may or may not be received and acted on by the receiver. As the pattern explicitly requires a response, if a receiver does not respond, the implication is non-receipt of the message and follow up action can be required by policy.

A series of request/reply pattern executions can be used to implement a conversational dialogues between parties. It should be noted that the response does not require an acknowledgement that would lead to an infinite loop.

Related Requirements

The request/reply pattern is one method of implementing the requirements regarding exchange of information between trusted parties. The selection of recipients and the signing of the message implement the requirement that information exchange between trusted parties should occur within a trusted context. The creation of the message hash and the optional encryption implement the requirement that the integrity of the information exchanged between trusted parties should be assured.

Related Patterns

All of the patterns in the section information Exchange between Trusted Parties address similar problems, and all are dependent on the patterns in the section establish a Trusted Context.

Related Use Cases

Request/Reply is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for information exchange between trusted parties is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

3.4.1.7 Polling

Synopsis

The polling pattern, in the context of a Trusted Multi-tenant Infrastructure (TMI), is a way to ask one or more recipients a question. This can be used to vote on a topic or to survey potential providers for policy compliance or asset availability.

Context

The polling pattern is used to send information to one or more recipients in anticipation of a null, partial or full subset of responses from the recipients. The content is often interrogatory in nature, although it can generate an action to be taken by receivers. The cardinality of senders to receivers is normally 1:*. What is important is that the receivers can identify the sender as a member of a trusted context. The sender should filter the receivers list to parties or devices within a shared context. For example, a provider may send a poll to all of the consumers using resources within a provider environment to verify readiness for a change. A consumer management agent may send a poll to all devices within the trusted systems domain to determine availability for work. The use of filters to limit the scope of polling messages is highly recommended.

Selection Criteria

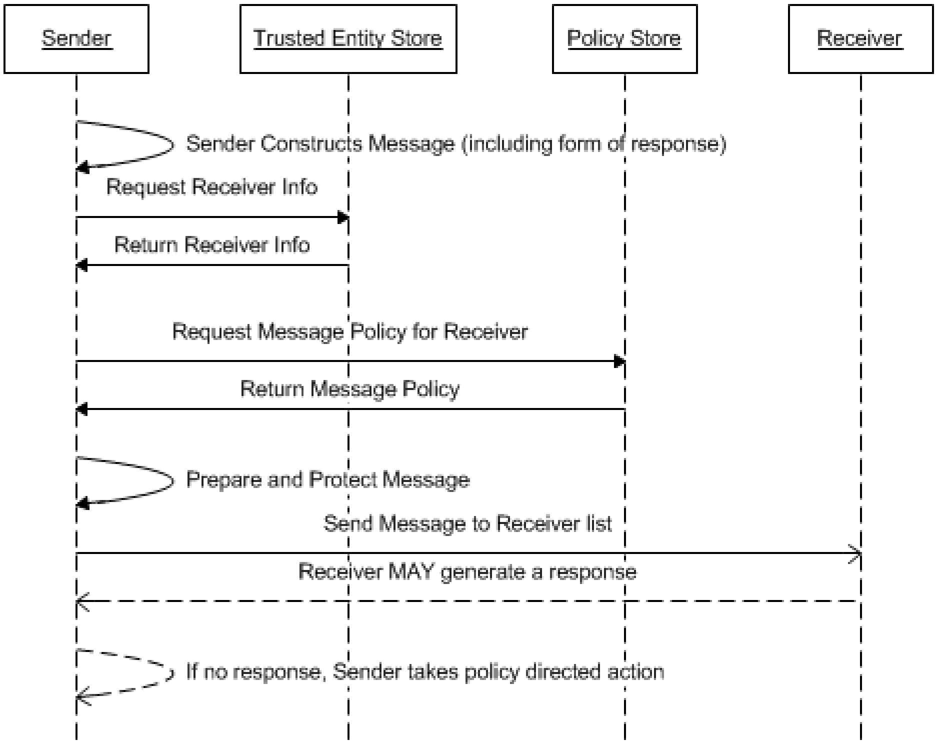
The polling pattern is used to send information when the sender is anticipating a reply. The message may be sent to all parties and devices where a trusted context has been established, or it may be sent to a filtered list of receivers. A response may be optional or required. It is not normally used for extended conversations, but for conducting a poll, or survey, of a group of recipients. It may be followed by a request/response conversational sequence with a receiver if required.

Solution

The polling pattern consists of the following steps:

1. The **Sender** constructs the message to be sent, including the form of response requested
2. The **Sender** identifies the recipients from the **Trusted Entity Store**
3. The **Sender** validates message policy, including actions to take

4. The **Sender** should apply appropriate protections to ensure the integrity of the message
5. The **Sender** should sign the message using its Attestation Key
6. The **Sender** can encrypt the message
7. The message is sent to the **Receiver(s)**
8. Each **Receiver** generates a reply to the message and sends the reply to the **Sender**
9. If no response is received, the **Sender** takes the action dictated by message policy


Sender
Trusted Entity Store
Policy Store
Receiver
Sender Constructs Message (including form of response)
Request Receiver Info
Return Receiver Info
Request Message Policy for Receiver
Return Message Policy
Prepare and Protect Message
Send Message to Receiver list
Receiver MAY generate a response
If no response, Sender takes policy directed action

Implications

The messages sent using the polling pattern may or may not be received and acted on by the receiver. As the pattern explicitly requires a response, if a receiver does not respond, the implication is non-receipt of the message and follow up action may be required by policy.

Related Requirements

The polling pattern is one method of implementing the requirements regarding exchange of information between trusted parties. The selection of recipients and the signing of the message implement the requirement that information exchange between trusted parties should occur within a trusted context. The creation of the message hash and the optional encryption implement the requirement that the integrity of the information exchanged between trusted parties should be assured.

Related Patterns

All of the patterns in the section information Exchange between Trusted Parties address similar problems, and all are dependent on the patterns in the section establish a Trusted Context.

Related Use Cases

Polling is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for information exchange between trusted parties is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

3.4.1.8 Brokered Exchange

Synopsis

The brokered exchange pattern, in the context of a Trusted Multi-tenant Infrastructure (TMI), is a way to proxy the exchange of information through a trusted 3rd party. It is not a standalone pattern, but is used in combination with one of the other information exchange patterns. Brokered Information Exchange encapsulates other patterns for establishing a trusted context, serving as an intermediary or proxy for the primary pattern.

Context

The Brokered Exchange pattern is used to proxy the exchange of information when the sender and receiver are not able to have a direct interaction. As the TMI requires information exchange to occur within a trusted context, it is sometimes necessary to use a trusted intermediary who has established a trusted context with all of the parties in an exchange. It is important is that the receivers can identify the sender as a member of a trusted context. The sender will send the receiver list and message to the broker, who then adds their signature to the message and any replies to establish the end to end trusted context.

When a broker is used, it is assumed that the other parties do not have a trust relationship appropriate to the context of the Trusted Systems Domain. In some cases, this pattern may be used to protect the identities of one or both parties from disclosure, with the broker serving as a trusted proxy between parties.

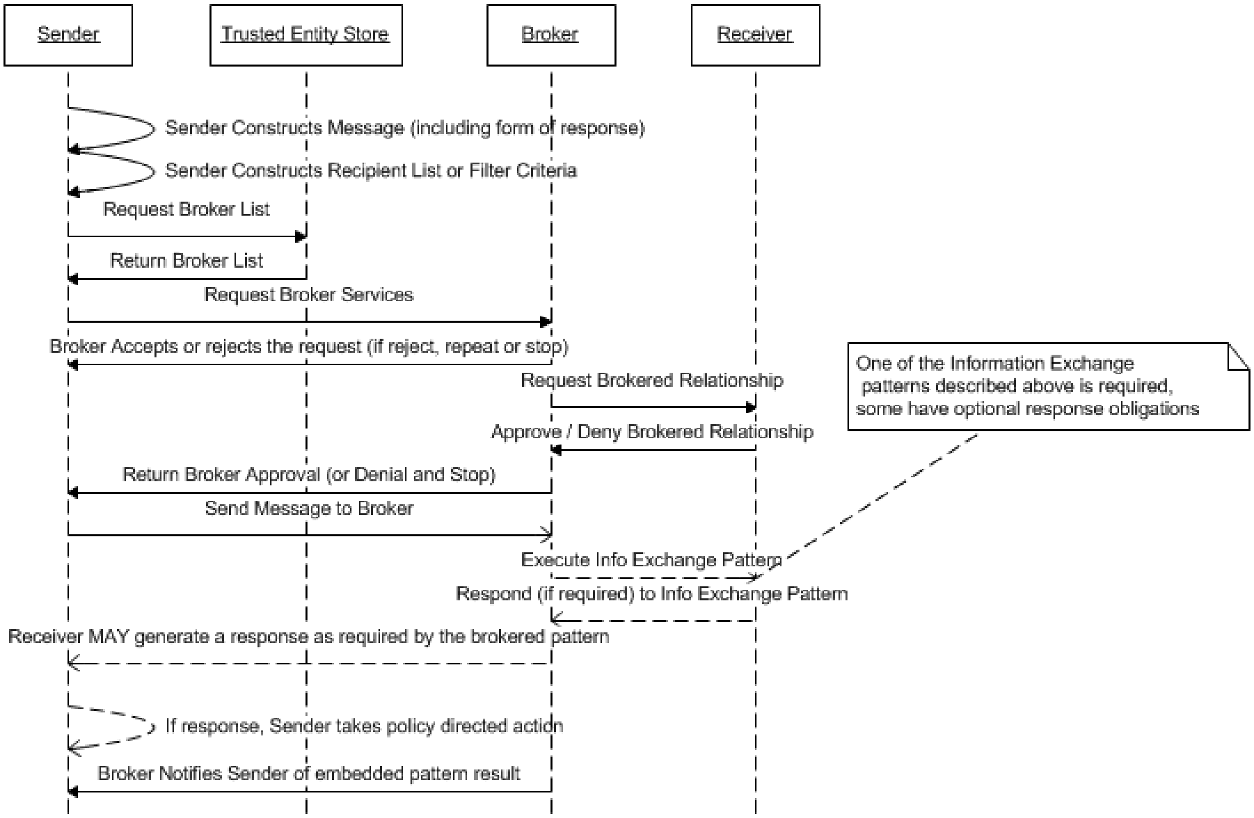
Selection Criteria

The brokered exchange pattern may be used to send information when the sender and receiver do not have a trusted context established. The message and recipient list is sent to a broker proxy, who then adds it to a trusted context between the broker and recipient(s) and forwards the message. The pattern should be used to encapsulate one of the other information exchange patterns, as it has no inherent information sharing pattern, except between the broker and senders and receivers.

Solution

The brokered exchange pattern consists of the following steps:

1. The **Sender** constructs the message to be sent, including any response obligation and the hash, signature and optional encryption of the message
2. The **Sender** identifies the **Receiver(s)** or the filter criteria for the **Broker** to determine the **Receiver(s)** as appropriate to the brokered exchange pattern**.**
3. The **Sender** requests a list of potential **Brokers** from the **Trusted Entity Store**
4. The **Sender** selects and confirms a broker
5. The **Sender** requests a trusted context with a list of **Receiver(s),** or filter criteria to derive the list, from the **Broker**
6. The **Broker** confirms delegates or declines the ability to broker the information exchange. If the request is rejected, the **Sender** can stop or try another **Broker**.
7. The **Sender** prepares the message to be sent, in accordance with the requirements of the selected information exchange pattern, notifies the **Broker** of the message and pattern to use
8. The **Sender** initiates the desired information exchange pattern with the recipients list, through the **Broker**
9. The **Broker** sends the sender an acknowledgement that the embedded pattern was executed.


Sender
Trusted Entity Store
Broker
Receiver
Sender Constructs Message (including form of response)
Sender Constructs Recipient List or Filter Criteria
Request Broker List
Return Broker List
Request Broker Services
Broker Accepts or rejects the request (if reject, repeat or stop)
Request Brokered Relationship
Approve / Deny Brokered Relationship
Return Broker Approval (or Denial and Stop)
Send Message to Broker
Execute Info Exchange Pattern
Respond (if required) to Info Exchange Pattern
Receiver MAY generate a response as required by the brokered pattern
If response, Sender takes policy directed action
Broker Notifies Sender of embedded pattern result
One of the Information Exchange patterns described above is required, some have optional response obligations

Implications

The brokered exchange pattern serves essentially as a wrapper to maintain the trusted context between parties in a TMI. It makes no assertions as to who that intermediary might be. It could be one of the providers or a totaly unrelated third party. As the intermediary can have access to any and all messages exchanged between the parties, it is important that both parties can rely on the integrity of the broker. Alternately, the messages could be encrypted by each party and signed by the broker.

Related Requirements

The brokered exchange pattern is one method of implementing the requirements regarding exchange of information between trusted parties. The selection of recipients and the signing of the message implement the requirement that information exchange between trusted parties should occur within a trusted context. The creation of the message hash and the optional encryption implement the requirement that the integrity of the information exchanged between trusted parties should be assured.

Related Patterns

All of the patterns in the section information Exchange between Trusted Parties address similar problems, and all are dependent on the patterns in the section establish a Trusted Context. The brokered exchange pattern should use one of the other information exchange patterns operating though the broker as proxy.

Related Use Cases

Brokered Exchange is one of several patterns implementing a core requirement for establishing a TMI. One or more of the patterns for information exchange between trusted parties is mandatory for TMI compliance and while not explicitly called out in one of the TMI use cases, is noted as a fundamental capability underlying all of the use cases.

3.4.2 Provision, Validate and Enforce Policies

Probably the most complex of the core functions, the requirement for policy determination, validation and enforcement within a trusted context ensures that controls are in place to protect the confidentiality, integrity and availability of information between parties in a multi-tenant ecosystem. These patterns are used to provision, manage, delegate decision authority and enforce policy and compliance requirements across a multi-tenant and multi-provider ecosystem.

The patterns for managing policy within the TMI are organized to decompose the process of policy provisioning, validation and enforcement:

Policy Administration. A policy is, in essence, a conditional expression followed by one or more declarative statements – essentially an if-then-else construct. This is generally populated with one or more attribute variables from a pre-defined dictionary of terms. Each of these variable terms is bound to a mechanism to resolve the value appropriate to the policy statement execution context. The administration of policy includes definition of policy statements. Policy definition also includes the rules for combining multiple policy statements into a combined rule or decision hierarchy, so that the resulting decisions will be unambiguous. Once the policy and combination rules are defined, they should be provisioned, or made available, to the Policy Management Controller (PMC).

Policy Validation. Once the policy has been defined and the rules for resolution of ambiguity are defined, the state of compliance should be tested. Within the trusted systems domain compliance validation could be driven by events, timed intervals or on request. Within the patterns in the TMI Reference Model, there are many references to policy validation within the patterns. This assures that the actions taken do not compromise the integrity of the trusted systems domain. Policy compliance is tested using a Policy Decision Point (PDP). The PDP is responsible for resolution of the policy statements into an executable rule, the resolution of variables (attributes) using the Policy Information Point (PIP) and the execution of the policy rule. A decision can be pass, fail or pass with obligations. An obligation is an additional step that should be taken in policy enforcement.

Policy Enforcement. The primary controller of policy within a trusted systems domain is a Policy Management Controller (PMC). This component serves as a controller for interaction between the PDP, PIP and the Policy Enforcement Point (PEP). The PMC is responsible to determine, from information in the Trusted Entity Store, which PDP’s need to be engaged in the resolution of policy within the context at hand. It determines the entities involved and determines the proper combination of PDP and

PEP to engage. Once a policy decision has been reached, the PEP takes the necessary action, based on the policy, in response to the policy decision.

The Policy Management patterns form the last element of the core functionality of the TMI Reference Model. All other functionality is dependent on the trusted context and compliance enforcement provided by policy enforcement capabilities within a trusted context.

3.4.2.1 Policy Administration

Synopsis

The ability to define policies and policy combination rules within the TMI is a key element of evaluating and enforcing configuration, separation and behavior as well as maintaining compliance within a multi-tenant environment. The domain owner establishes a policy or set of policies that appropriately asserts standards for operation of the domain but also accounts for key stakeholders and their policy needs. Policy Administration involves the ability for a domain owner to establish/modify policy, policy sets and policy resolution rules within their domain. The Policy Administration Point (PAP) is the interface for maintenance of the Policy Store.

Context

In order to operate in a trusted multi-tenant environment, policy should be established within each domain by the domain owner. This pattern describes the establishment of policy, policy sets and policy resolution rules within a domain to provide policy enforcement and decisions regulating access to resources. When a Trusted Systems Domain (TSD) is allocated, the Trusted Entity Store and the Policy Store are allocated. The default policy is to allow the TSD owner to manage policy but deny all other actions. The Domain Owner uses the Policy Administration pattern and the Policy Administration Point (PAP) to establish domain policy. The PAP serves as the Policy Enforcement Point (PEP) for the Policy Store.

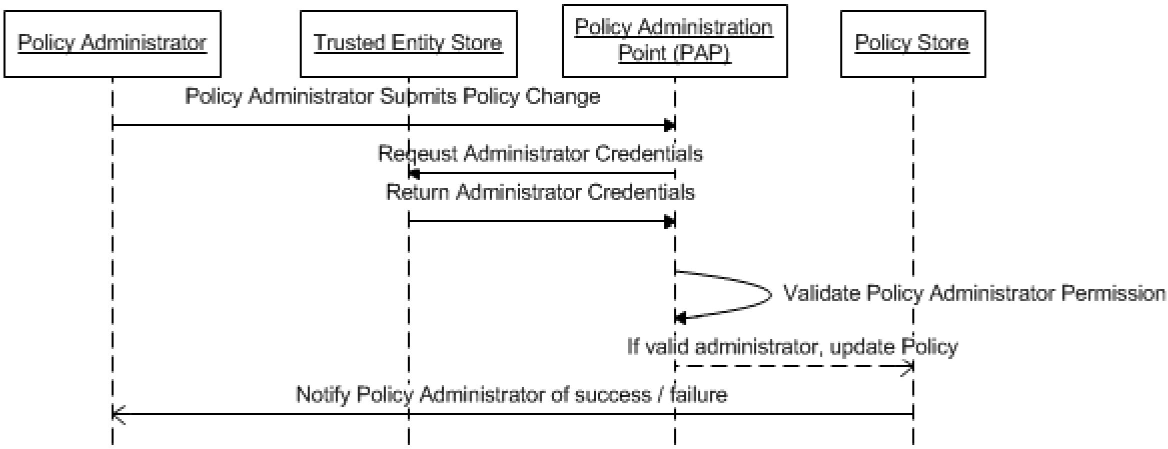
Selection Criteria

Policy Administration is used by domain owners to establish and maintain policy stores. This pattern allows the domain owner the ability to establish and modify their domain policy/policy sets to meet their specific policy compliance needs within the TMI.

Solution

1. A **Policy Administrator** submits a policy change to the **Policy Administration Point (PAP)**
2. The **Policy Administration Point** requests credentials from the **Trusted Entity Store** for the **Policy Administrator**
3. The **Trusted Entity Store** returns credentials for the **Policy Administrator**

4. The **Policy Administration Point** validates that the requestor is a valid **Policy Administrator**
5. If the Requestor is a valid **Policy Administrator** the **Policy Administration Point** updates the **Policy Store**.
6. The **Policy Administrator** is notified of the success or failure of the change


Policy Administrator
Trusted Entity Store
Policy Administration Point (PAP)
Policy Store
Policy Administrator Submits Policy Change
Reqeust Administrator Credentials
Return Administrator Credentials
Validate Policy Administrator Permission
If valid administrator, update Policy
Notify Policy Administrator of success / failure

Implications

Policy Administration maintains on-going policy compliance standards for resources in the domain but having large policies or multiple policy sets to verify can affect the performance within the domain.

Modifications to policy/policy sets can cause unforeseen side effects within the domain unintentionally restricting or creating unknown policy violations. It is vital that only trusted parties have access to the PAP and that policies that are established and modified go through a robust review process.

Policy Administrators require roles and access rights are validated against the Trusted Entity Store and the PDP associated with the Policy Store (part of the PAP) to determine which policies the administrator has access to create and modify.

Related Requirements

Trust Relationships should be established before a policy can be created/modified within the domain.

Policies should be established before conducting monitoring, reporting and provisioning within the domain.

Related Patterns

Establish Trust

Trusted Data Exchange

Monitoring Services

Reporting Services

Provisioning Services

Direct Policy Enforcement

Related Use Cases

Applies to all TMI Use Cases

3.4.2.2 Policy Decision Authority Resolution

Synopsis

The ability to orchestrate policy decisions within the TMI is a key element of resolving and enforcing appropriate policy as well as maintaining compliance within a multi-tenant environment. Domain owners establish a policy or set of policies that appropriately meet their standards but also account for key stakeholders and their policy needs. Depending on which assets and operators are involved in an action, a clear understanding of where the decision authority lies for enforcement of policy is a critical part of maintaining appropriate control and separation of duties within the TMI. Each asset and operator has policy enforcement information stored within the Trusted Entity Store. This includes the URI of the Policy Decision Point (PDP), scope of authority and acceptable policy decision configuration options. This information is retrieved by the Policy Management Controller (PMC) and used to make a determination of how a policy decision is to be orchestrated.

Context

Policy Decision Authority Resolution is the process by which information is gathered for each of the parties to a decision and the orchestration process is determined. Each party has the responsibility to assign, delegate or describe the policy enforcement mechanisms used for assets under its control. This separation of duties is an important concept within the multi-tenant, multi-provider world of the TMI. A provider is responsible for the physical assets or operational processes for managing the pools of resources it allocates to the various trusted systems domains. The consumer is responsible for managing the assets allocated to the trusted systems domain. It is quite possible that a single asset may be affected by multiple policy decision authorities. The key to understand is whether the action for which a decision is being sought affects one or more of these stakeholders. If only a single stakeholder is involved, then the PDP is assigned and no further action is needed. If there are multiple stakeholders, then one or more of the other Policy Decision Authority Resolution steps may be need to determine PDP priority or rule combination authority.

Selection Criteria

Within the various TMI patterns, there is often a need to identify, validate and enforce policy compliance. For every policy decision, it is imperative that the correct decision authorities are involved. This pattern is the base pattern that collects and determines the stakeholders in a policy resolution action. It is used whenever policy resolution is called for. The additional resolution steps can be used to further refine the situations where either there are multiple PDPs involved or resolution can be performed by a single PDP, but required policy input from multiple Policy Stores.

To simplify:

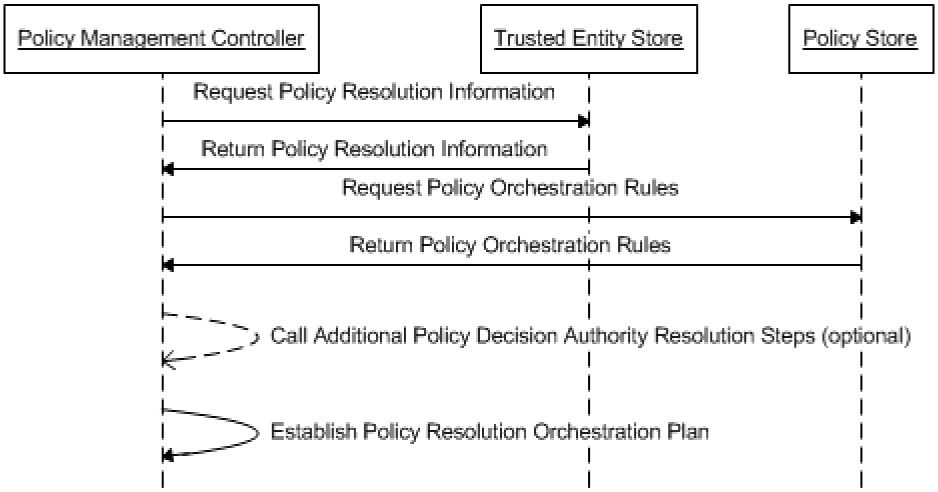
- The **Base Pattern** is always used to bring together the information about policy resolution stakeholders, PDPs and orchestration rules
- The **Rule Combination** steps are used if multiple decision authorities are identified, but the rules allow for a single PDP to gather policy from multiple policy stores and create an integrated policy using a rule combination algorithm
- The **PDP Hierarchy** steps are used if the action for which a policy decision is needed involves multiple PDP instances, each of which should be independently queried and policy is not shared between PDP instances.

It is possible to use all three of the pattern sections for a single decision.

Solution

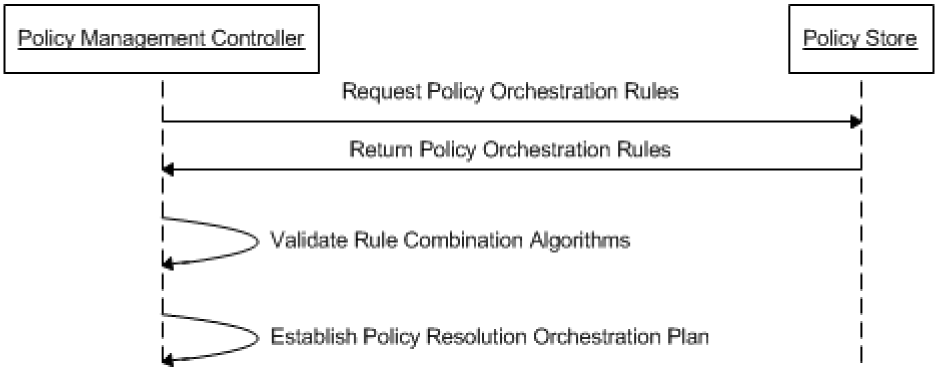
Base Pattern

1. The **Policy Management Controller** requests the Policy Resolution information from the **Trusted Entity Store(s)** of each asset or operator involved in an action.
2. The **Trusted Entity Store(s)** returns the Policy Resolution Information for the action to the **Policy Management Controller**.
3. The **Policy Management Controller** requests Policy Resolution Rules from the **Policy Store**
4. The **Policy Store** returns the Policy Resolution Rules
5. The **Policy Management Controller** may call additional policy resolution orchestration steps as specified in the **Trusted Entity Store(s)** for the assets/operator(s) involved.
6. The **Policy Management Controller** determines the policy resolution orchestrations steps necessary to reach a policy decision between the assets and/or operators involved.


Policy Management Controller
Trusted Entity Store
Policy Store
Request Policy Resolution Information
Return Policy Resolution Information
Request Policy Orchestration Rules
Return Policy Orchestration Rules
Call Additional Policy Decision Authority Resolution Steps (optional)
Establish Policy Resolution Orchestration Plan

Rule Combination

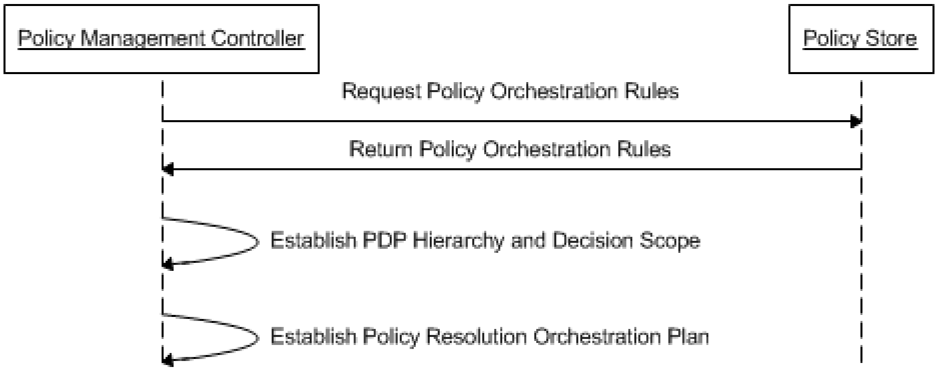
1. The **Policy Management Controller** requests Policy Combination Algorithm(s) from the **Policy Store(s)**
2. The **Policy Store** returns the Policy Combination Algorithm(s)
3. The **Policy Management Controller** validates that the Rule Combination Algorithms are executable
4. The **Policy Management Controller** establishes the orchestration steps needed to execute a policy decision


Policy Management Controller
Policy Store
Request Policy Orchestration Rules
Return Policy Orchestration Rules
Validate Rule Combination Algorithms
Establish Policy Resolution Orchestration Plan

PDP Hierarchy

1. The **Policy Management Controller** requests Policy Orchestration Rule(s) from the **Policy Store(s)**
2. The **Policy Store** returns the Policy Orchestration Rule(s) that govern PDP priority and Scope of Authority
3. The **Policy Management Controller** establishes the PDP hierarchy and validates that the scope of authority is clear

4. The **Policy Management Controller** establishes the orchestration steps needed to execute a policy decision


Policy Management Controller
Policy Store
Request Policy Orchestration Rules
Return Policy Orchestration Rules
Establish PDP Hierarchy and Decision Scope
Establish Policy Resolution Orchestration Plan

Implications

A **Policy Management Controller** that is interfacing with a large number of PDPs should maintain a proper prioritization amongst all the stakeholders.

Related Requirements

All interactions with protected resources require trusted information exchanges to make appropriate authorization decisions.

Trusted Information exchange relies on the establishment of policy in order to make appropriate access control decisions.

Related Patterns

Establish Trust

Trusted Data Exchange

Monitoring Services

Reporting Services

Provisioning Services

Related Use Cases

Applies to all TMI Use Cases.

3.4.2.3 Single PDP Decision

Synopsis

The ability to make policy decisions within the TMI is a key element of conducting critical authorization decisions as well as maintaining compliance within a multi-tenant environment. A policy decision is made by resolving a policy statement within the context of the action and environment in which the action is to take place. A policy statement is an IF-THEN-ELSE construct that contains dictionary references to variable attributes that are resolved, allowing the final statement to be evaluated and a decision returned. A Policy Management Controller handles the orchestration of the policy enforcement process, including interfacing with the PDP to make policy decisions. The dictionary is associated with the Policy Information Point (PIP) and handles resolution of attribute variables for the PDP. The Policy Store contains the policy statements and glossary information. There are a number of combinations of these elements possible. In this pattern a single policy store contributes policy statements and a single PDP makes policy decisions on behalf of all stakeholders.

Context

The Single PDP Decision pattern is able to make decisions based upon policy statements from a single policy store. If a Policy Enforcement Point intercepts an action that requires a decision and the Policy Decision Authority Resolution authority resolves to a single PDP, then a decision is requested and the result returned to the PEP. The PEP then allows the action, denies the action or allows the action with obligations. An obligation may reflect a pre or post condition to the action.

Selection Criteria

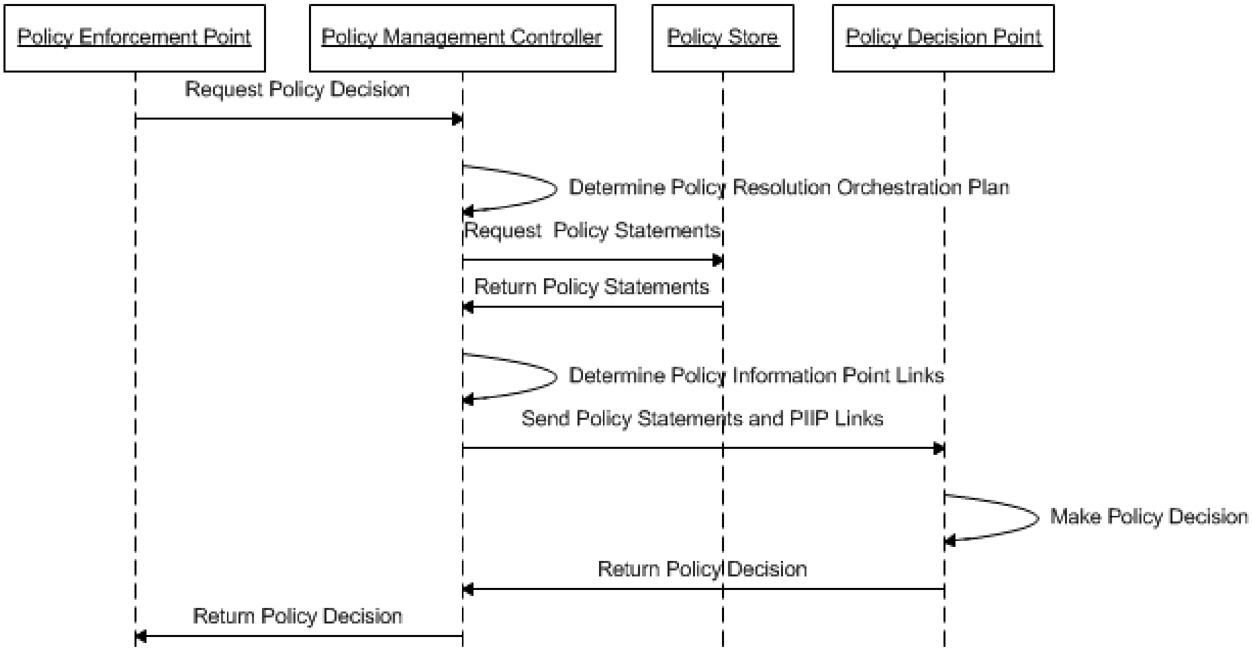
The Single PDP Decision pattern is selected when only one PDP is necessary to make policy decisions for an action. This can occur when either all of the assets in an action are under the policy control of the policy owner or all parties agree to delegate decision authority to the policy owner, resulting in a single Policy Decision Resolution Authority, or PDP.

Solution

1. The **Policy Enforcement Point** intercepts an action which requires a policy decision
2. The **Policy Management Controller** determines the Policy Resolution Orchestration Plan
3. The **Policy Management Controller** determines that the Policy Resolution Orchestration Plan contains a single Decision Authority (**Policy Decision Point**)
4. The **Policy Management Controller** pulls the policies from the **Policy Store**.
5. The **Policy Management Controller** determines the Policy Information Points for attribute resolution.
6. The **Policy Management Controller** passes control to the **Policy Decision Point** along with the policy statements and PIP links

7. The **Policy Decision Point** returns a policy decision to the **Policy Management Controller**

8. The **Policy Management Controller** returns the policy decision to the **Policy Enforcement Point**


Policy Enforcement Point
Policy Management Controller
Policy Store
Policy Decision Point
Request Policy Decision
Determine Policy Resolution Orchestration Plan
Request Policy Statements
Return Policy Statements
Determine Policy Information Point Links
Send Policy Statements and PIIP Links
Make Policy Decision
Return Policy Decision
Return Policy Decision

Implications

Implementation of this pattern maintains on-going policy compliance with resources in your domain but having large policies or multiple policy sets to verify can affect the performance within your domain.

Related Requirements

Trust Relationships should be established before a policy can be created within the domain.

All interactions with protected resources require trusted information exchanges to make appropriate authorization decisions.

Trusted Information exchange relies on the establishment of policy in order to make appropriate access control decisions.

Policies should be established before conducting monitoring, reporting and provisioning within the domain.

Related Patterns

Establish Trust

Trusted Data Exchange

Monitoring Services

Reporting Services

Provisioning Services

Related Use Cases

Applies to all TMI Use Cases.

3.4.2.4 Rule Combination Decision

Synopsis

The ability to make policy decisions within the TMI is a key element of conducting critical authorization decisions as well as maintaining compliance within a multi-tenant environment. A policy decision is made by resolving a policy statement within the context of the action and environment in which the action is to take place. A policy statement is an IF-THEN-ELSE construct that contains dictionary references to variable attributes that are resolved, allowing the final statement to be evaluated and a decision returned. A Policy Management Controller handles the orchestration of the policy enforcement process, including interfacing with the PDP to make policy decisions. The dictionary is associated with the Policy Information Point (PIP) and handles resolution of attribute variables for the PDP. The Policy Store contains the policy statements and glossary information. There are a number of combinations of these elements possible. In this pattern, multiple policy stores contribute policy statements that are combined such that a single PDP can make policy decisions on behalf of all stakeholders.

Context

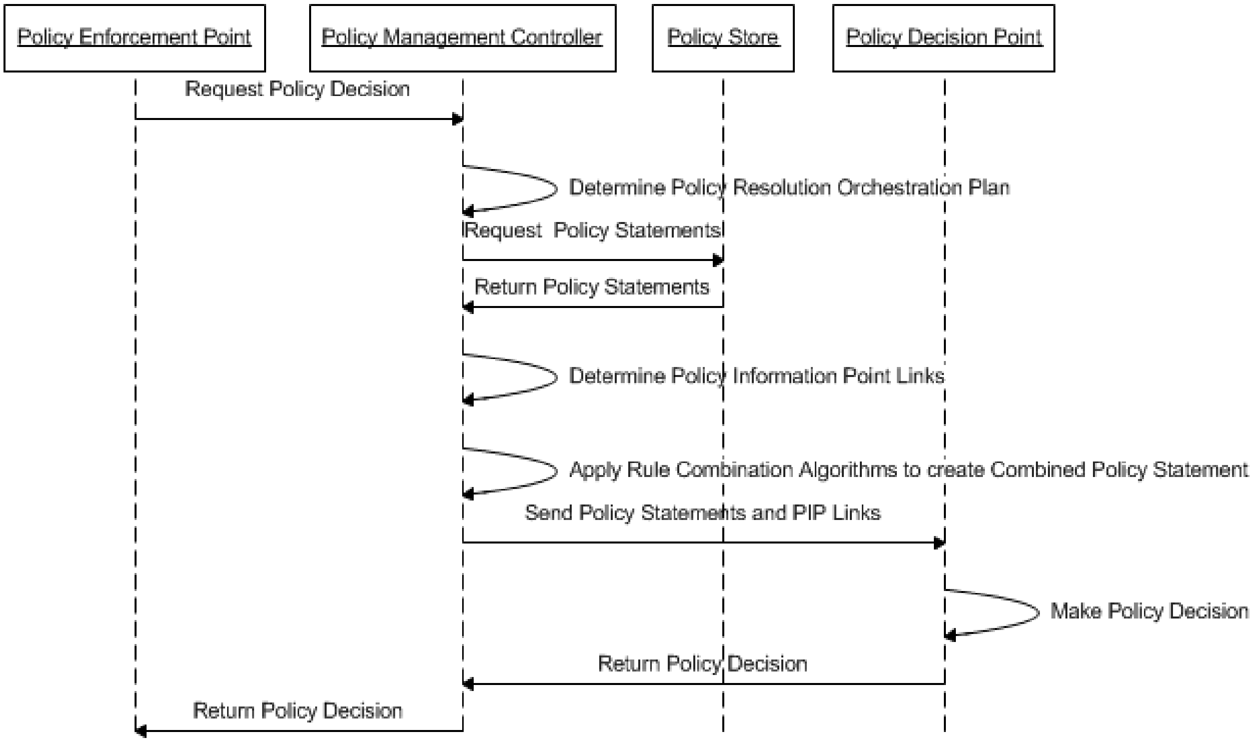
The Rule Combination Decision pattern describes how a single PDP is able to make decisions based upon policy statements from multiple policy stores. If a Policy Enforcement Point intercepts an action that requires a decision and the Policy Decision Authority Resolution authority resolves to a single PDP with multiple policy stores, then a decision is requested from the PDP, the policy statements are collected and the statements are combined or prioritized based upon an agreed Rule Combination Algorithm. The result is returned to the PEP. The PEP then allows the action, denies the action or allows the action with obligations. An obligation may reflect a pre or post condition to the action.

Selection Criteria

The Rule Combination Decision pattern is selected when only one PDP is necessary to make policy decisions for an action but multiple stakeholders have policy stores with relevant policy statements. This can occur when all parties agree to delegate decision authority to a single PDP and have agreed to a rule combination algorithm. It is critical that Policy Management Controllers properly prioritize policy sets and establish policy hierarchies that maintain policy compliance across all stakeholders involved.

Solution

1. The **Policy Enforcement Point** intercepts an action which requires a policy decision
2. The **Policy Management Controller** determines the Policy Resolution Orchestration Plan
3. The **Policy Management Controller** determines that the Policy Resolution Orchestration Plan contains multiple Decision Authorities but can leverage a single **Policy Decision Point** (PDP) using Rule Combination.
4. The **Policy Management Controller** requests the policies from the **Policy Stores**.
5. The **Policy Management Controller** determines the Policy Information Points (PIP) for attribute resolution.
6. The **Policy Management Controller** combines the rules into a combined policy set and sends it to the **Policy Decision Point** for resolution.
7. The **Policy Management Controller** passes control to the PDP along with the combined policy statements and PIP links
8. The **Policy Decision Point** returns a policy decision to the **Policy Management Controller**
9. The **Policy Management Controller** returns the policy decision to the **Policy Enforcement Point**


Policy Enforcement Point
Policy Management Controller
Policy Store
Policy Decision Point
Request Policy Decision
Determine Policy Resolution Orchestration Plan
Request Policy Statements
Return Policy Statements
Determine Policy Information Point Links
Apply Rule Combination Algorithms to create Combined Policy Statement
Send Policy Statements and PIP Links
Make Policy Decision
Return Policy Decision
Return Policy Decision

Implications

Implementation of this pattern maintains on-going policy compliance with resources in the domain but having large policies or multiple policy sets to verify can affect the performance within the domain.

Related Requirements

Trust Relationships should be established before a policy can be created within the domain.

All interactions with protected resources require trusted information exchanges to make appropriate authorization decisions.

Trusted Information exchange relies on the establishment of policy in order to make appropriate access control decisions.

Policies should be established before conducting monitoring, reporting and provisioning within the domain.

Related Patterns

Establish Trust

Trusted Data Exchange

Monitoring Services

Reporting Services

Provisioning Services

Related Use Cases

Applies to all TMI Use Cases.

3.4.2.5 PDP Hierarchy Decision

Synopsis

The ability to make policy decisions within the TMI is a key element of conducting critical authorization decisions as well as maintaining compliance within a multi-tenant environment. A policy decision is made by resolving a policy statement within the context of the action and environment in which the action is to take place. A policy statement is an IF-THEN-ELSE construct that contains dictionary references to variable attributes that are resolved, allowing the final statement to be evaluated and a decision returned. A Policy Management Controller handles the orchestration of the policy enforcement process, including interfacing with the PDP to make policy decisions. The dictionary is associated with the Policy Information Point (PIP) and handles resolution of attribute variables for the PDP. The Policy Store contains the policy statements and glossary information. There are a number of combinations of these elements possible. In this pattern, multiple decision authorities represent the various stakeholders and are not able or willing to delegate decision authority to a single PDP. Each PDP makes a discrete decision and then the Policy Management Controller uses an established hierarchy and prioritization rules to weight and evaluate the combined decisions.

Context

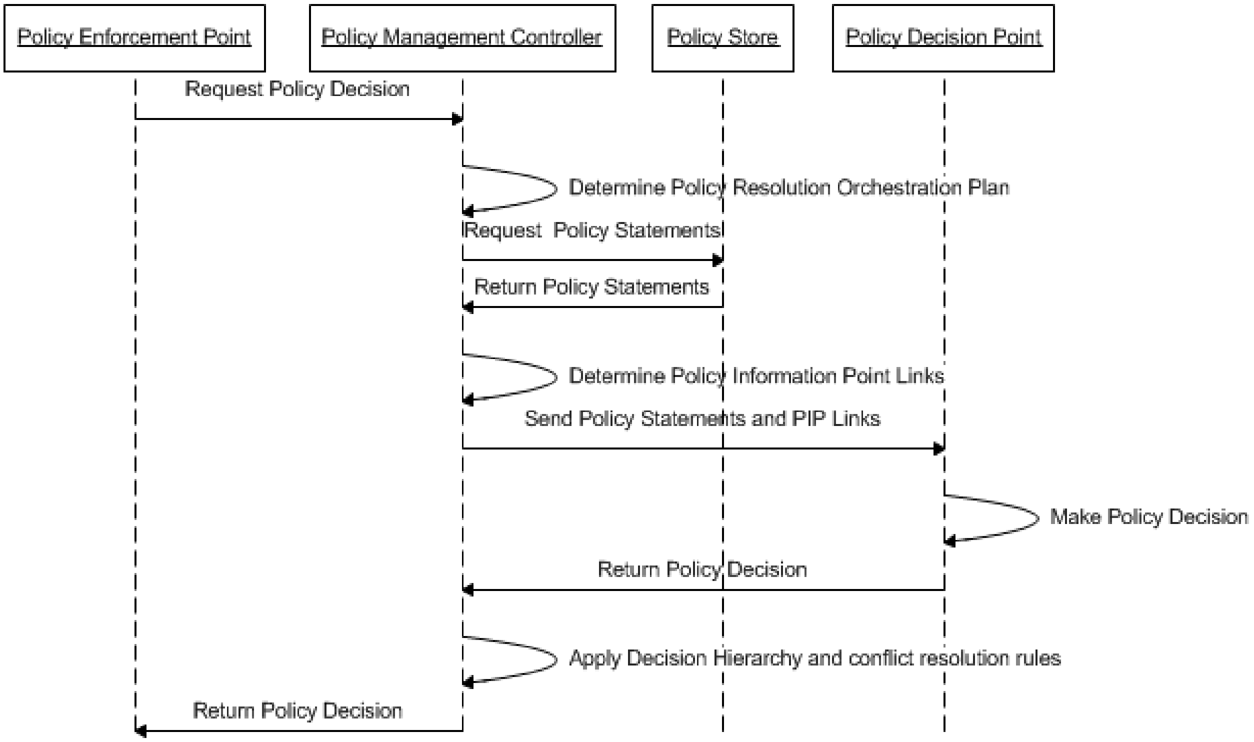
The PDP Hierarchy Decision pattern describes how multiple PDPs from various decision authorities can collaborate to make policy decisions. If a Policy Enforcement Point intercepts an action that requires a decision and the Policy Decision Authority Resolution authority resolves to multiple PDPs with multiple policy stores, then a decision is requested from each PDP, the decisions are collected and the decisions are combined or prioritized based upon an agreed Hierarchy and conflict resolution algorithm. The result is returned to the PEP. The PEP then allows the action, denies the action or allows the action with obligations. An obligation may reflect a pre or post condition to the action.

Selection Criteria

The Rule Combination Decision pattern is selected when multiple PDPs are necessary to make policy decisions for an action representing multiple stakeholders with relevant policy statements. The decisions are then aggregated and combined based upon a PDP hierarchy and conflict resolution policy. This can occur when parties are not able to delegate decision authority to a single PDP and have agreed to a decision hierarchy and conflict resolution policy. It is critical that Policy Management Controllers properly prioritize policy sets and establish policy hierarchies that maintain policy compliance across all stakeholders involved.

Solution

1. The **Policy Enforcement Point** intercepts an action which requires a policy decision
2. The **Policy Management Controller** determines the Policy Resolution Orchestration Plan
3. The **Policy Management Controller** determines that the Policy Resolution Orchestration Plan contains multiple Decision Authorities but can leverage a single **Policy Decision Point** (PDP) using Rule Combination.
4. The **Policy Management Controller** requests the policies from the **Policy Stores**.
5. The **Policy Management Controller** determines the **Policy Information Points** for attribute resolution.
6. The **Policy Management Controller** passes control to the **Policy Decision Points** along with the PIP links
7. The **Policy Decision Points** return policy decisions to the **Policy Management Controller**
8. The **Policy Management Controller** combines the decisions and resolves any conflicts.
9. The **Policy Management Controller** returns the policy decision to the **Policy Enforcement Point**


Policy Enforcement Point
Policy Management Controller
Policy Store
Policy Decision Point
Request Policy Decision
Determine Policy Resolution Orchestration Plan
Request Policy Statements
Return Policy Statements
Determine Policy Information Point Links
Send Policy Statements and PIP Links
Make Policy Decision
Return Policy Decision
Apply Decision Hierarchy and conflict resolution rules
Return Policy Decision

Implications

Implementation of this pattern maintains on-going policy compliance with resources in your domain but having large policies or multiple policy sets to verify can affect the performance within the domain.

Related Requirements

Trust Relationships should be established before a policy can be created within the domain.

All interactions with protected resources require trusted information exchanges to make appropriate authorization decisions.

Trusted Information exchange relies on the establishment of policy in order to make appropriate access control decisions.

Policies should be established before conducting monitoring, reporting and provisioning within the domain.

Related Patterns

Establish Trust

Trusted Data Exchange

Monitoring Services

Reporting Services

Provisioning Services

Related Use Cases

Applies to all TMI Use Cases.

3.4.2.6 Policy Enforcement

Synopsis

The ability to provide policy enforcement within the TMI is a key element of conducting critical authorization decisions as well as maintaining compliance and separation within a multi-tenant environment. The domain owner establishes a policy or set of policies that appropriately meets their standards but also accounts for key stakeholders and their policy needs. A Policy Enforcement Point (PEP) is associated with an action to be taken within the TMI context. It is often an agent or interface of the system that can engage the policy management services and then has the authority and ability to implement and enforce the policy decisions associated with the action. A PEP is therefore only rarely a generic construct, as it requires some level of integration into the system in order to effectively implement policy to modify the flow of the process.

Context

The Policy Enforcement pattern describes how a Policy Enforcement Point (PEP) serves as the agent of the stakeholders to enforce policy decisions associated with an action within the TMI. A Policy Enforcement Point intercepts an action that requires a decision and the Policy Decision Authority Resolution determines the Policy Decision Points (PDP) to be engaged. A decision is requested from the appropriate PDP(s). The result is returned to the PEP. The PEP then allows the action, denies the action or allows the action with obligations. An obligation may reflect a pre or post condition to the action.

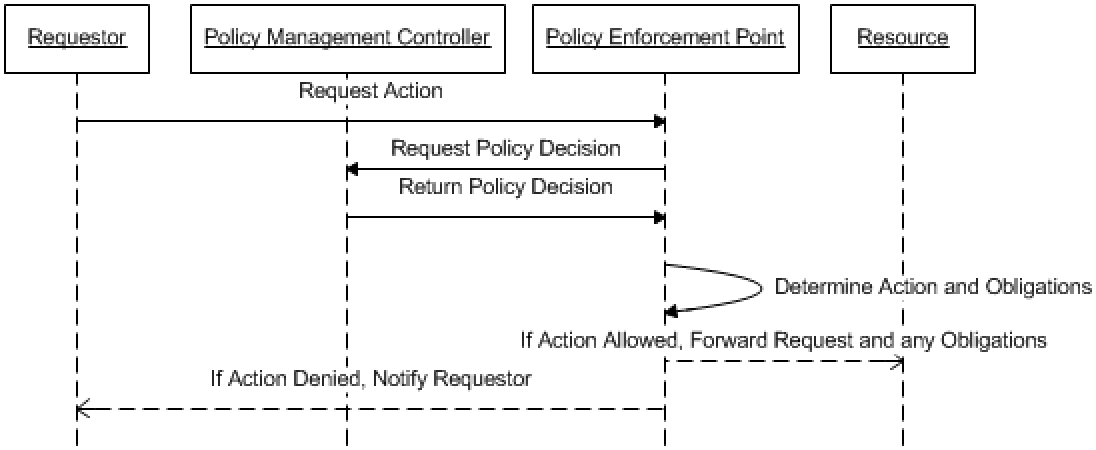
Selection Criteria

The Policy Enforcement pattern is selected when an action is attempted within the TMI that requires a policy decision. The enforcement of the policy is directly enabled and is not brokered through a third party. The policy decisions may be brokered or require interaction with multiple decision authorities, but the enforcement is not brokered for selection of this pattern.

Solution

1. A **Requestor** requests an action against a resource within the domain.
2. The **Policy Enforcement Point (PEP)** acting as the resource intercepts the **Requestor's** request.

3. The **Policy Enforcement Point** forwards the request to the **Policy Management Controller (PMC).**
4. After assessing the policy the **Policy Management Controller** sends back an authorization decision to the **Policy Enforcement Point** to either accept or deny the **Requestor's** request for an action against the resource.
5. The **Policy Management Controller** forwards the authorization decision back to the **Policy Enforcement Point** to either allow the request to the resource or deny the **Requestor's** request. Any obligations imposed by the decision authority are processed by the **Policy Enforcement Point**
6. If the authorization decision permits the action then the **Policy Enforcement Point** forwards the request to the **Resource**.
7. If the request is denied, the **Requestor** is notified


Requestor
Policy Management Controller
Policy Enforcement Point
Resource
Request Action
Request Policy Decision
Return Policy Decision
Determine Action and Obligations
If Action Allowed, Forward Request and any Obligations
If Action Denied, Notify Requestor

Implications

Implementation of this pattern maintains on-going policy compliance with resources in your domain but having large policies or multiple policy sets to verify can affect the performance within the domain.

Related Requirements

Trust Relationships should be established before a policy can be created within the domain.

All interactions with protected resources require trusted information exchanges to make appropriate authorization decisions.

Trusted Information exchange relies on the establishment of policy in order to make appropriate access control decisions.

Policies should be established before conducting monitoring, reporting and provisioning within the domain.

Related Patterns

Establish Trust

Trusted Data Exchange

Monitoring Services

Reporting Services

Provisioning Services

Related Use Cases

Applies to all TMI Use Cases.

3.5 Management Services

Management Services use TCG Technology and other appropriate industry standards to describe the foundational relationship between the various components in a Trusted Multi-tenant Infrastructure (TMI) and how they are managed. The ability to manage configuration of services, proactively monitoring assets, reporting compliance, and responding to events/audits provide the main implementation focus for Management Services within a cloud or shared infrastructure environment

A consumer can manage assets within the trusted systems domain environment and a provider can manage the provider environment as well as the various consumer domains within a cloud or shared infrastructure. All management in the TMI is done using policies. In terms of context – “management” means the ability to perform administrative functions against assets within the Consumer trusted systems domain and Provider environment in order to achieve and maintain policy compliance.

3.5.1 Monitoring Services

There are two basic Monitoring Services within the TMI, monitoring of events and monitoring of state. This service can be used to proactively monitor an assets’ audit, event, and state information to ensure policy adherence. The policies created (or configured) within the TMI determine how the monitoring services monitor activities on assets

Monitoring can be implemented in a variety of methods, including state based, agent based, agent-less, and event based. The TMI does not specify the specific approach to monitoring as long as all state and event can effectively be monitored in conformance with policy.

3.5.1.1 State Monitoring

Synopsis

State Monitoring is the process of utilizing sensors that actively collect information on the state of an asset within the TMI.

Context

In order to operate a TMI, state monitoring should be established for each party to ensure that policy compliance is maintained. This pattern describes the utilization of state monitoring within the platform to provide proactive attestation of platform assets against policy.

We tend to think in terms of monitoring assets. However, monitoring can be applied to any entity in the TMI. Components of the TMI itself, which are not necessarily assets, can be monitored. This can be applied to application code as well as physical assets. The monitoring infrastructure and the monitoring repository should be flexible enough to fulfill this objective.

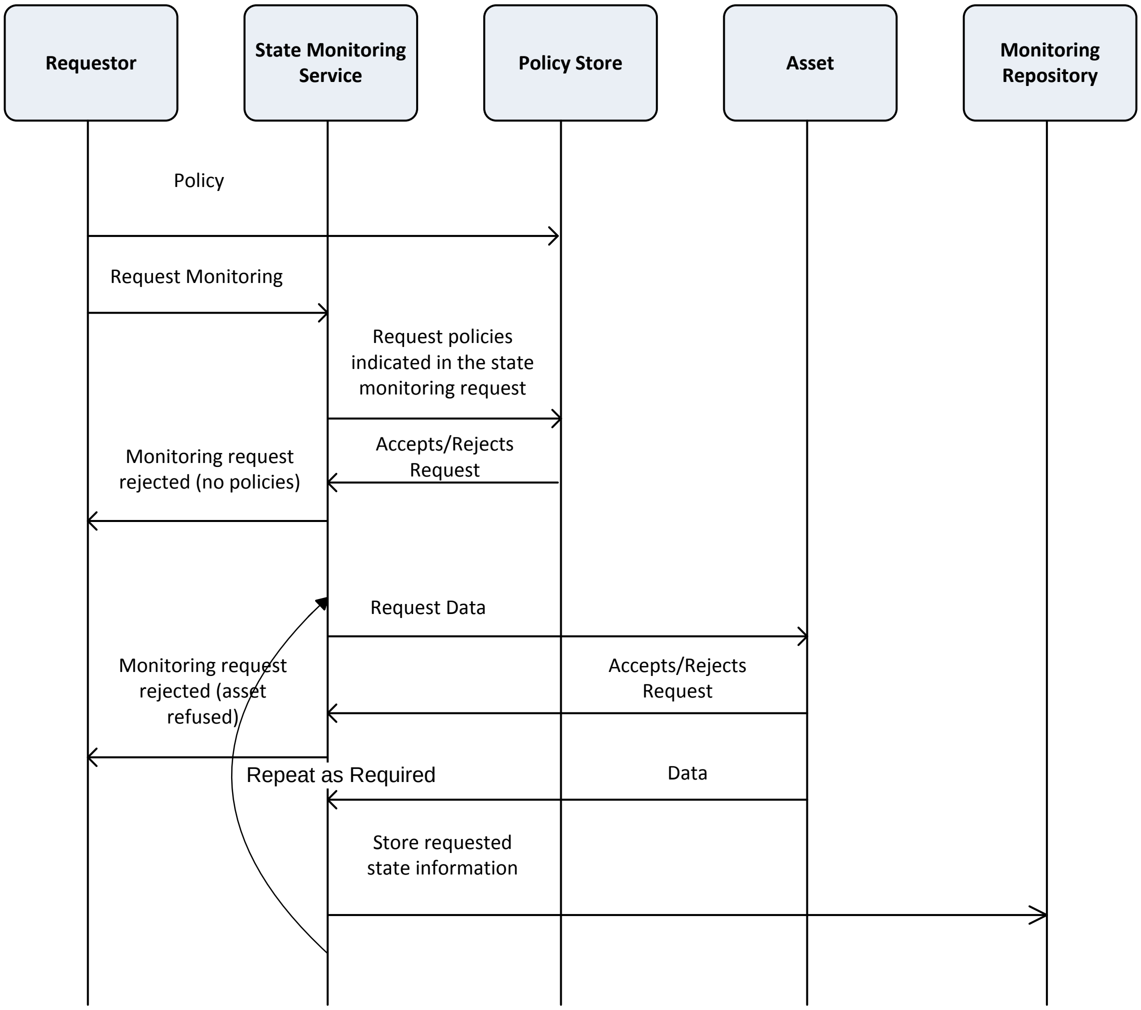
Selection Criteria

State monitoring is selected when the data is to be requested from the asset (or entity) by the monitoring service. The monitoring service can make a one-time request or repeated requests on a periodic basis. Each request is initiated by the monitoring service. Data received from an asset is trusted if the asset is equipped with a hardware base root of trust that support attestation, such as a TPM. Even if there is secure communications between the monitoring service and the asset, if the asset is not equipped with a hardware based root of trust, trust in the reported results has to be based upon other factors. Even with a TPM, for long running systems additional support should be available to assure the continued integrity of the system and its monitoring infrastructure.

Solution

The state monitoring service consists of several steps:

1. Policies that will govern the state monitoring are placed in the **Policy Store**.
2. The **Requestor** asks the **State Monitoring Service** to monitor an **Asset**.
3. The **State Monitoring Service** requests the indicated policies from the **Policy Store** in order to determine the state monitoring procedures for the TMI. The **Policy Store** contains information on monitoring repositories where the **State Monitoring Service** should store information that is collected. If the required policy information cannot be located, the **Policy Store** rejects the request and the **State Monitoring Service** rejects the request with a “no policy” indication.
4. Once policy is acquired the **State Monitoring Service** acts as a sensor requesting state data from the **Assets** within the TMI. If the **Asset** rejects the request for data, the **State Monitoring Service** rejects the request with an “asset refused” indication.
5. The asset returns the requested data to the **Monitoring Service**. Depending on the request the **Monitoring Service** may have to process the data before it is recorded in the repository.
6. The State **Monitoring Service** stores the data collected from the Assets in a **Monitoring Repository**.


Requestor
State Monitoring Service
Policy Store
Asset
Monitoring Repository
Policy
Request Monitoring
Request policies indicated in the state monitoring request
Accepts/Rejects Request
Monitoring request rejected (no policies)
Request Data
Monitoring request rejected (asset refused)
Accepts/Rejects Request
Repeat as Required
Data
Store requested state information

Implications

The use of state monitoring implies the existence of a baseline configuration or maximum and minimum threshold for acceptable configuration. It also implies some acceptable timeframe over which state is accepted before being revalidated. A change in state or the presence or absence of state information may trigger an event that requires evaluation against policy for the asset.

Related Patterns

- Reporting Pattern has a post processing relationship to the monitoring pattern(s) to provide policy compliance reporting that contain state information regarding the assets within the TMI.

- Provisioning/De-Provisioning of assets is required to establish the state monitoring service and Assets.
- Trusted Data Exchange to perform secure communication between the state monitoring service and policy store as well as assets.
- Policy should be applied to define monitoring procedures.
- The Correlation service can be used to analyze the state information populated into the state monitoring repository. Event correlation may subscribe to this information. The correlation engine has the ability to modify the TMI within the constraints specified by policy.
- State monitoring can cause a report to be generated.

Related Use Cases

UC-2 Provider: Modification of the established Provider Environment Policy

UC-5 Provider: Re-provision Trusted Systems Domain Assets based on changes to the Trusted Systems Domain Policy.

UC-6 Provider: Audit of policy within the Provider Environment Policy.

UC-6 Consumer: Audit of policy within the Trusted Systems Domain.

UC-1 Consumer: Modification of the established Trusted System Domain Policy

3.5.1.2 Event Monitoring

Synopsis

Event monitoring captures events within the TMI.

Context

Event monitoring is provided in a TMI to enhance it manageability. This pattern describes the utilization of event monitoring within the platform to capture event information. Event information can be used for policy compliance validation, billing, or other functions of the TMI. The policy governing the events to be monitored and who has access to the event monitoring data should be established before an event monitoring request exists or it will fail.

Selection Criteria

The event capturing infrastructure should be sufficiently flexible to capture events wherever they are generated, code or data. Event capture and logging is required to make the TMI flexible enough to enable self-monitoring. Events can come from components of the TMI as well as assets under management. All captureable events should have sufficient metadata associated with them so that the quality of the data can be assessed and access privileges can be enforces according to policy.

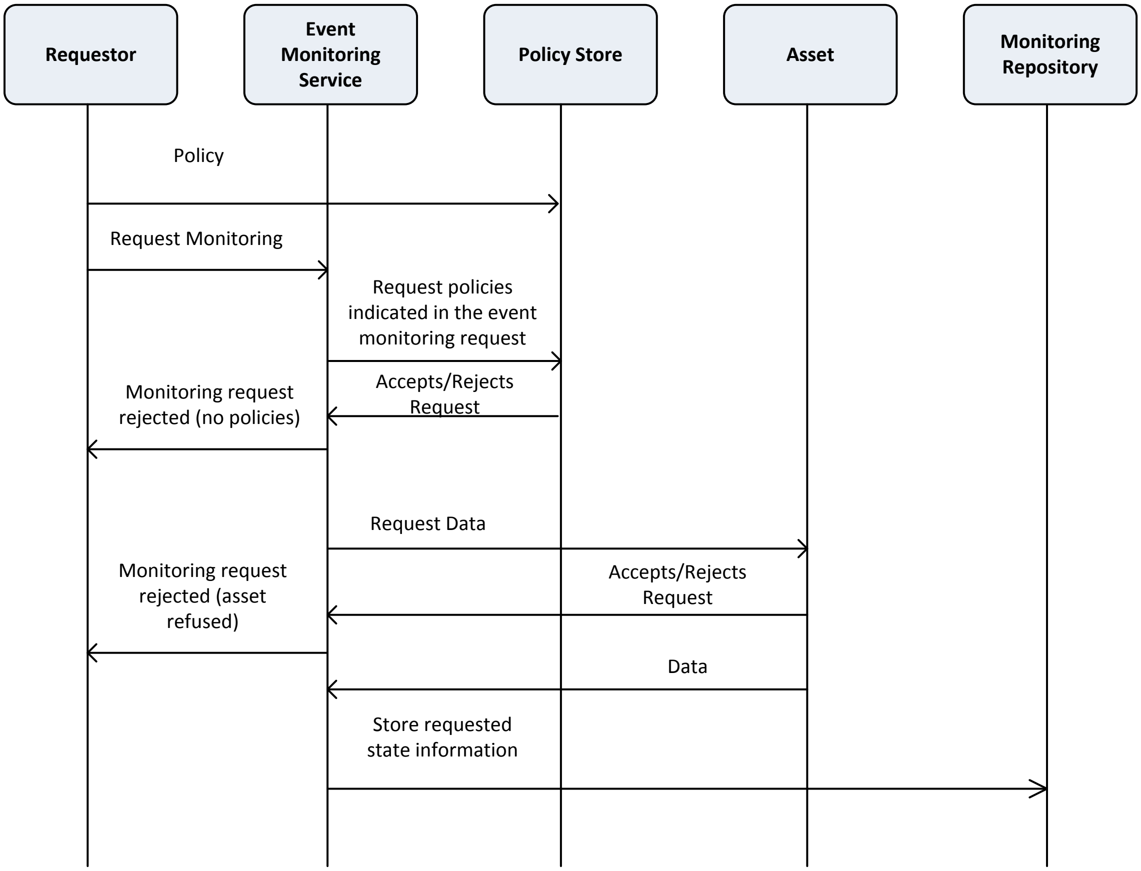
Event monitoring is requested when the desired data is expected to be generated by an asset and when the availability of the data cannot be predicted in advance. Event monitoring establishes a publish/subscribe framework between the asset and the monitoring service. The asset is the publisher of the information and the event monitor is a subscriber to the information. Event monitoring provides the ability to capture events that occur within and between assets in the TMI.

Solution

The event monitoring service consists of several steps:

1. The **Requestor** or other person, such as the operator of the TMI places the policies that will govern the events to be collected and user access to the event data in the **Policy Store**
2. A **Requestor** sends an event Monitoring request to the **Event Monitoring Service**.
3. The **Event Monitoring Service** requests that the events be monitored for the indicated **Requestor**. The request (policy) indicates the **Monitoring Repository** where the captured events will be stored.
4. The **Policy Store** accepts or rejects the request. The **Policy Store** will reject the request if the indicated **Requestor** is not authorized to access the requested events. The **Policy Store** will also reject the request if there are no policies that cover the indicated events. In both cases the rejection will say “no policy” because there is no policy that authorized the **Requestor** to access the events.
5. If the request is rejected the information will be passed onto the **Requestor**.
6. If the request is accepted, the **Event Monitoring Service** will request the data from the indicate **Asset**.
7. If the **Asset** rejects the request the **Event Monitoring Service** will pass the rejection along to the requestor with an “event request rejected” indication.
8. If the request is accepted, the event data will be published by the **Asset**, the **Event Monitoring Service** has become a subscriber to this data.
9. The **Event Monitoring Service** stores the data collected from the **Assets** in the indicated **Monitoring Repository**.

2071


Requestor
Event Monitoring Service
Policy Store
Asset
Monitoring Repository
Policy
Request Monitoring
Request policies indicated in the event monitoring request
Accepts/Rejects Request
Monitoring request rejected (no policies)
Request Data
Monitoring request rejected (asset refused)
Accepts/Rejects Request
Data
Store requested state information

2072
2073

2074 **Implications**

2075 The event monitoring service is used to monitor events for a variety of reasons. For
2076 example, events could be monitored so that policy compliance can be continuously
2077 performed within the TMI. The trustworthiness of the captured events depends each
2078 asset having a secure state (hardware root of trust combined with a trusted context)
2079 defined in order to perform the monitoring activities.

2080

2081 **Related Patterns**

2082 Reporting Pattern has a post processing relationship to the monitoring pattern(s) to
2083 provide policy compliance reporting that contain event information regarding the
2084 assets within the TMI.

2085 Provisioning/De-Provisioning of assets is required to establish the event monitoring
2086 service and Assets.

2087 Trusted Data Exchange to perform secure communication between the monitoring
2088 services and policy store as well as assets and the monitoring service.

Policy should be applied to define monitoring procedures.

The Correlation analyzes the event information populated into the event monitoring repository. The event monitoring service does not have the ability to modify the TMI. However, the correlation engine can modify the TMI in response to events within the constraints allowed by policy.

Event monitoring can cause a report to be generated.

Related Use Cases

UC-2 Provider: Modification of the established Provider Environment Policy

UC-5 Provider: Re-provision Trusted Systems Domain Assets based on changes to the Trusted Systems Domain Policy.

UC-1 Consumer: Modification of the established Trusted System Domain Policy

UC-6 Provider: Audit of policy within the Provider Environment Policy.

UC-6 Consumer: Audit of policy within the Trusted Systems Domain.

3.5.2 Monitoring Data and Policy Correlation

Synopsis

Monitoring data and policy correlation compares state and/or event information against the relevant compliance policies, trusted baselines, alone, or in combination. The policy store contains rules on when to run compliance audits and state reports, as well as how to respond to events passed from the monitoring services. Also defined are the rules for evaluation of events and state information, including thresholds and response actions. Once the monitor data has been evaluated a decision is made on whether to trigger an event that could lead to further data collection, an enhanced evaluation workflow, generation of a reporting action or a management action against the domain.

Context

In order to operate in a TMI, correlation should be established for each party to ensure that events are compared against a secure baseline and/or compliance policy. This pattern describes the utilization of correlation monitoring within the platform to provide correlations between secure baselines and compliance policies.

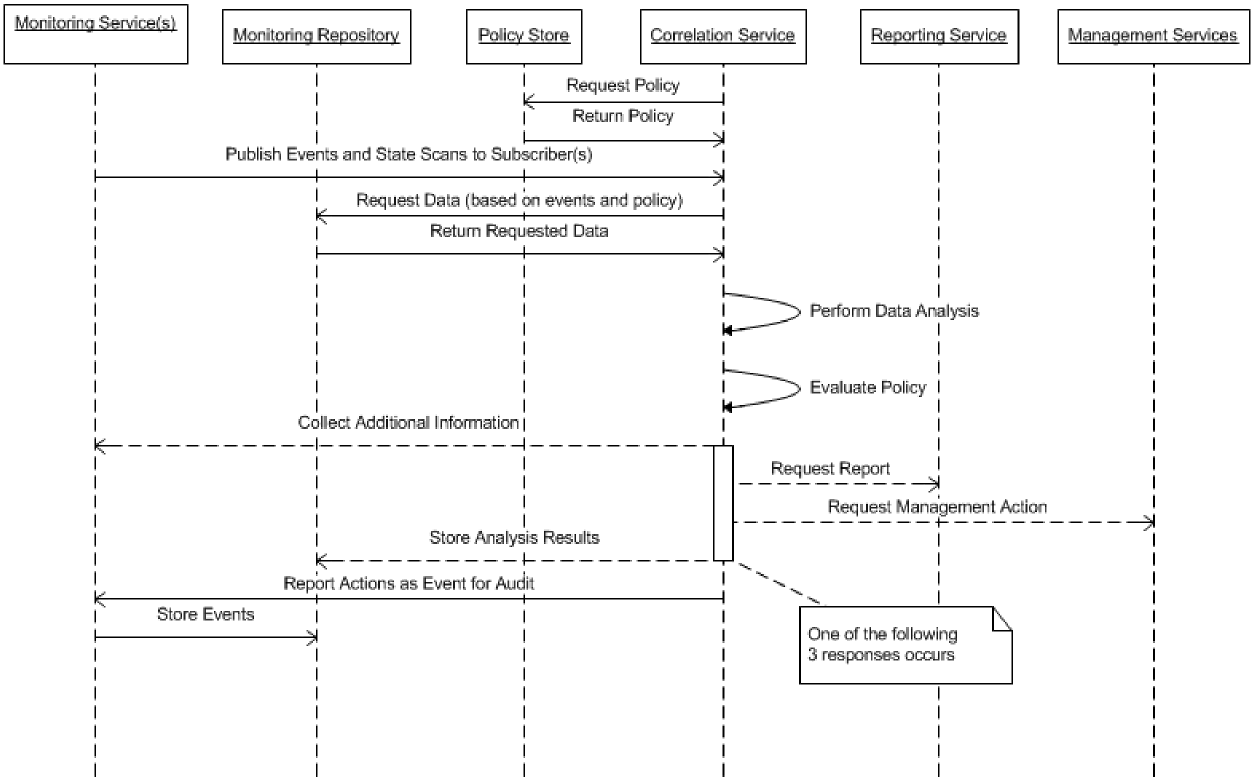
Selection Criteria

Correlation monitoring provides the ability to compare state or event information against a trusted baseline or defined compliance policy to validate integrity associated with the domain.

Solution

Monitoring correlation consists of several steps:

1. The **Correlation Service** retrieves policies from the **Policy Store** in order to determine the correlation monitoring procedures.
2. The **Correlation Service** is notified of new events or state scans based on subscriptions to topics published by the ***Monitoring Services***, as well as policy driven correlation triggers such as scheduled correlations to identify missing events.
3. The **Correlation Service** retrieves data from the **Monitoring Repository** that requires correlation based on policy.
4. The **Correlation Service** analyzes the information in accordance with policy. This may result in new derived monitoring information
5. The **Correlation Service** evaluates the information against the policy statements.
6. If further action is required based on the correlation of monitoring data against policy statements, one or more of the following **Correlation Service** actions can occur:
    a. An event is triggered to the ***Monitoring Service*** to collect additional information
    b. A request is forwarded to the **Reporting Service** provide information to administrators, users or other systems as needed
    c. An action is triggered on the ***Management Services*** to take some action on the domain (provision, configure, etc.)
    d. The results of data analysis are stored in the **Monitoring Repository**
7. The **Correlation Service** actions are reported as events to the **Monitoring Service**



Implications

Trusted state baselines have to be defined for each asset or specific policies are needed to allow the correlation service to compare the event results captured.

Related Patterns

Reporting Pattern has a post processing relationship to the correlation to provide policy compliance reporting that contain event information regarding the assets within the TMI.

Provisioning/De-Provisioning of assets is required to establish the event monitoring service and Assets.

Trusted Information Exchange patterns are used to perform secure communication between the monitoring services and policy store as well as assets.

Policy should be applied to define monitoring procedures.

The Correlation Service subscribes to events published by the event monitoring service using a publish/subscribe pattern

Agent-based, Agentless, and State Monitoring patterns populate the repositories that the Event Correlation Monitoring Service subscribes to validate policy compliance within the TMI.

Related Use Cases

UC-2 Provider: Modification of the established Provider Environment Policy

UC-5 Provider: Re-provision Trusted Systems Domain Assets based on changes to the Trusted Systems Domain Policy.

UC-1 Consumer: Modification of the established Trusted System Domain Policy

UC-6 Provider: Audit of policy within the Provider Environment Policy.

UC-6 Consumer: Audit of policy within the Trusted Systems Domain.

3.5.3 Reporting Service

Reporting Services within the TMI are intended to serve as a management service that reactively conduct reporting of the asset's audit, event, and state information to ensure policy adherence. Configuration of policies within the TMI drives how the reporting services within the TMI collect information on assets. All reporting is done on data that exist in the monitoring repository. The reporting service can subscribe to events that can be used to trigger a report. The reporting service can also use the correlation service to perform analytics on data from the event repository. The results of the analytics can be included in the report and recorded in the monitoring repository. The reporting service can ask the correlation service to subscribe to events that would cause the correlation service to periodically analyze data and record it in the monitoring repository. The activities of the correlation service are events that can be monitored.

The reporting service does not take any action that modifies the TMI. It generates reports that can be acted upon by other agents. Modifications to the TMI in response to reports, events, or state changes would have to be initiated by the correlation service.

Synopsis

Reports can be generated at any point in time. Reports may contain one or more of event data, state data, or correlated data. Data for reports is extracted from the monitoring repository. The reliability of the data in the monitoring repository depends on the trust model that has been established with the reporters. All data in the monitoring contains metadata recorded by the repository that indicates the trust model between the repository and the originator of the data. Signed data will have the same metadata, an indication of the trust model between the supplier and the repository and an indication of who sent the data.

Context

Reporting is a critical part of any complex infrastructure. There should be a mechanism to generate reports from data that is being monitored in a TMI. These reports can be used by the owner of the TMI to ascertain the state of their infrastructure, conduct/direct repairs, and validate billing and other infrastructure costs. Reports request are logged in the monitoring repository. Actual reports can be logged in the audit repository or the monitoring repository. Reporting never changes the state of assets. Reporting does not trigger changes of state to assets. The consumers of reports should take explicit actions to change the state of assets. Reports may exploit the correlation engine to generate data for the reports or to cause an event that initiates the creation of a report. In this pattern – an external supervisor/management entity (one of several possibilities) called the system management interface is requesting state data from assets within the TMI.

Selection Criteria

Reporting can be requested for any asset or entity. The quality of the report is dependent on the trust model between the asset and the monitoring repository

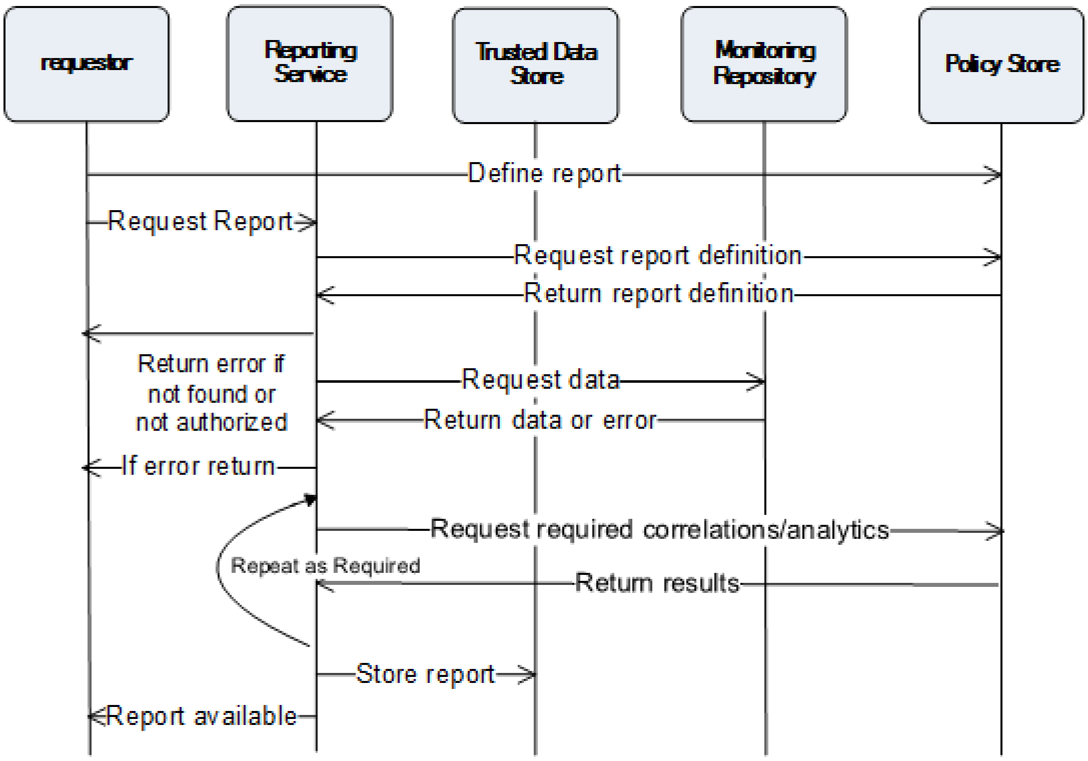
Solution

There are two fundamental elements of a reporting service: Report description and report generation. The Reporting service has to have a mechanism for reports to be defined. This definition includes a description of all the data required for the report, description of any requested analysis, report frequency, and the access permissions for the report. The TMI may optionally include some pre-defined reports. The data description refers to data that is in the monitoring repository. A report cannot be generated unless the requestor has permissions to all of the data that is not blinded by the report. The description of the data required by the report has to indicate whether the data is blinded by the report or in some sense transparent (or leaked) by the report. The analytics are assumed to be performed by the correlation engine. A reporting system has to allow both for predefined reports and for dynamically defined reports. For both cases the report description will be stored in the report repository along with a description of the requestor. Policy will determine how long dynamically generated report descriptions are retained. The report frequency should also include whether or not the report is automatically generated or generated only upon request. The access permissions for the report cannot override the requirement that the requestor of the report should have permission to all data not blinded by the report. Reports can be authored for a narrow or wide audience.

There is a concept of data being blinded by a report. The basic issue is that event data contains metadata that indicates who is allowed to see the event data. Requesting a report cannot enable an entity in the TMI (user or system) to gain access to data they are not authorized to see. However, a report may consume data and produce a report that does not allow the reader to derive some of the input data that was generated by the report. Data that is used to generate a report but cannot be derived by reading or processing the resulting report is considered to be blinded by the report. When data is blinded by a report the access authorizations associated with the data do not flow to the resulting report. If the data used to generate a report is not blinded by a report the access authorization that are associated with the data flow to the resulting report.

1. Someone or some process defines a report. The report definition is placed in the **Policy Store**.

2. A **Requestor** requests a report. The **Reporting Service** request the report from the indicated **Policy Store**.
3. The **Policy Store** returns the report definition or an error if it does not exist
4. The **Reporting Service** confirms that the **Requestor** is authorized for all non-blinded data. If not authorized, the **Reporting Service** returns an error.
5. The **Reporting Service** requests the indicate data from the **Monitoring Repository**.
6. The **Monitoring Repository** returns the requested data or an error if it does not exist.
7. The **Reporting Service** returns an error to the **Requestor** if one is indicated on the data request.
8. The **Reporting Service** generates the requested report. This could involve multiple calls to the Correlation Service to perform analytics on the requested data
9. The **Reporting Service** stores the report in a **Trusted Data Store** and indicates its location to the **Requestor**.



Implications

The reporting service when combined with the monitoring service and the correlation service ensures that on-going policy compliance is performed actively within the TMI.

These services rely on each asset having the ability to accurately report to the monitoring service.

Related Patterns

Monitoring Pattern has a precursor relationship to the reporting pattern to enable policy compliance reporting that contain state information regarding the assets within the TMI.

Provisioning/De-Provisioning of assets is required to establish the reporting service and Assets.

Trusted Data Exchange to perform secure communication between the reporting service, monitoring service and policy store as well as assets.

Policy should be applied to define prior to report generation.

The Correlation Service will perform all auditing processes and take any required actions within the limits specified by policy.

Related Use Cases

UC-2 Provider: Modification of the established Provider Environment Policy

UC-5 Provider: Re-provision Trusted Systems Domain Assets based on changes to the Trusted Systems Domain Policy.

UC-6 Provider: Audit of policy within the Provider Environment Policy.

UC-6 Consumer: Audit of policy within the Trusted Systems Domain.

UC-1 Consumer: Modification of the established Trusted System Domain Policy

3.5.4 Management/Control Services

Synopsis

Management/Control Services within the TMI are intended to serve as a management service that provides service initiation/decommission asset adjustment, and administrative sustainment of assets.

Context

In order to operate in a TMI, Management/Control Services should be established for each party to ensure that administrative functions within the TMI allow for Asset policy compliance. This pattern describes the utilization of Management/Control Services within the platform to provide service initiation/decommission asset adjustments, and administrative sustainment of the assets.

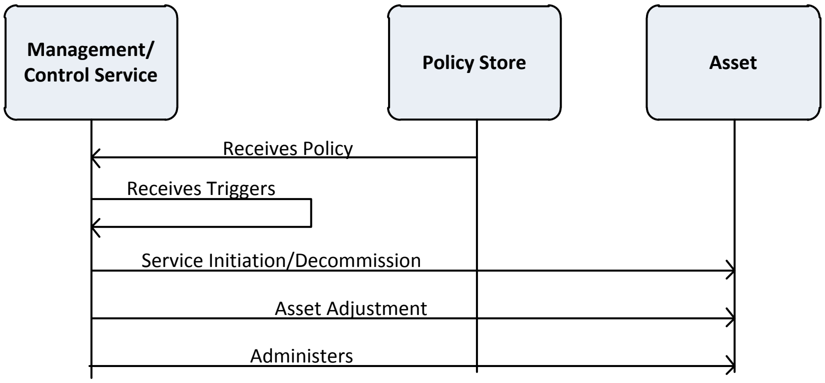
Selection Criteria

Management/Control Service is utilized when the administration of Assets includes initiation/decommission of Asset services, making adjustments to the Assets, and performing administrative sustainment activities on Assets driven by Management/Control policies.

Solution

Event correlation monitoring consists of several steps:

1. The **Management/Control Service** retrieves policies from the **Policy Store** in order to conduct administrative activities.
2. Once policy is applied the **Management/Control Service** can respond to triggers to take required actions including initiating/decommissioning services, making **Asset** adjustments, and administering **Assets** within the TMI.



Implications

The event monitoring service ensures that on-going policy compliance is performed actively within the TMI but it relies on each asset having a defined secure baseline or having specific policies to allow for event correlation.

The Management/Control Services allows for the adjustments of assets, sustainment of asset configurations, and initiation/decommission of services to maintain proper management of assets against defined TMI policies.

Related Patterns

Reporting Pattern has a post processing relationship to management and control to provide policy compliance reporting that contain event information regarding the assets within the TMI.

Provisioning/De-Provisioning of assets is required to establish the event monitoring service and Assets.

Trusted Data Exchange to perform secure communication between the monitoring services and policy store as well as assets.

Policy should be applied to define monitoring procedures.

Agent-based and Agent-less event monitoring to collect the events from the assets.

State Monitoring services to determine the current state of the asset and verify compliance against baselines and policies.

Related Use Cases

UC-1 Consumer: Modification of the established Trusted System Domain Policy.

UC-2 Consumer: Use of the Consumer Management Agent to manage resources within the Trusted System Domain
UC-3 Consumer: Use of the Consumer Management Agent after deviation from Trusted Systems Domain steady state after modification of Platform Environment hardware/software.

UC-5 Consumer: The retirement of the Asset within the Trusted Systems Domain

UC-2 Provider: Modification of the established Provider Environment Policy.

UC-10 Generic: Provision application components within the Trusted Systems Domain

3.6 Provisioning Services

Provisioning services are used to create, change, or destroy resources within a multi-tenant infrastructure. The provisioning agent acts on behalf of the requestor. The provisioning agent may be acquiring or acting on a resource or set of resources. If there is a policy store associated with an item, there should be policy allowing the request in the policy store or the request will fail. For every request the credentials of the requestor should be validated.

3.6.1 Provisioning a Trusted Systems Domain

Synopsis

A Trusted Systems Domain should be provisioned before any other action can be taken on it or for it. This service is used to create a trusted systems domain with an empty policy store.

Context

When a consumer desires to create and start using a multitenant infrastructure. The consumer should first establish trusted communication with the multitenant infrastructure and use this trusted channel to create a Trusted Systems Domain. After the trusted systems domain is created the Trusted Systems Domain Policy Store should be populated with the default policies for the Trusted Systems domain. Provisioning Services, **Error! Reference source not found.** are used to place policies in the policy store.

Selection Criteria

It is assumed that a trusted channel has been established between the consumer and the provider. A new Trusted Systems Domain is created if allowed. The Trusted Systems Domain, an empty Trusted Systems Domain Policy Store, and a Trusted

Identity Store with the credentials for the existing trusted context between the consumer and provider are returned to the consumer.

Solution

1. The **Consumer Management Agent** requests that the **Provider Management Agent** create a new Trusted Systems Domain.
2. The provider checks the **Provider Systems Domain Policy Store** to see if it can allocate a Trusted Systems Domain.
3. If the provider is allowed to fulfill the request

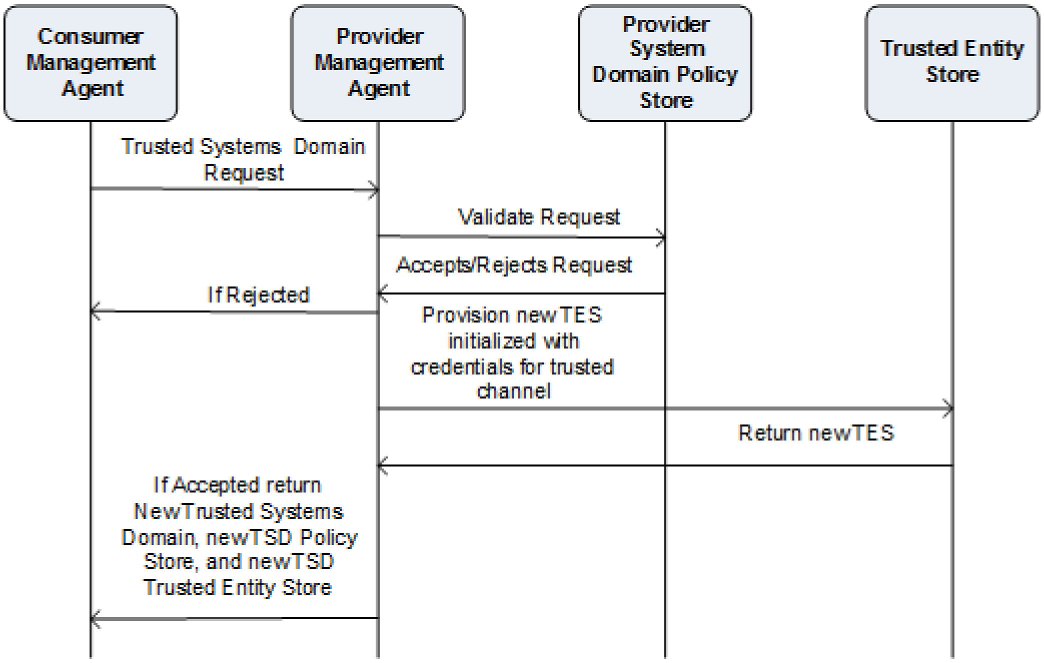
    a. The provider allocates a Trusted Systems Domain for the **Consumer Management Agent**

    b. The provider allocates an empty **Policy Store** associated with the Trusted Systems Domain

    c. The provider creates a **Trusted Entity Store** that is part of the Trusted Systems Domain. The **Trusted Entity Store** is initialized with the credentials associated with the trusted context that exist between the **Consumer Management Agent** and the **Provider Management Agent**.

    d. The provider returns to the consumer the new Trusted Systems Domain, the empty Trusted Systems Domain Policy Store, and the **Trusted Entity Store.**

4. If it is not allowed, the **Provider Management Agent** indicates to the **Consumer Management Agent** that the request cannot be fulfilled.


Consumer Management Agent
Provider Management Agent
Provider System Domain Policy Store
Trusted Entity Store
Trusted Systems Domain Request
Validate Request
Accepts/Rejects Request
If Rejected
Provision new TES initialized with credentials for trusted channel
Return new TES
If Accepted return New Trusted Systems Domain, new TSD Policy Store, and new TSD Trusted Entity Store

Implications

If the Trusted Systems Domain is provisioned then the consumer **should** populate the Trusted Systems Domain policy store before any other actions can be completed. Inability to provision a new trusted systems domain can be caused by a number of factors including the consumer not being allowed to add another domain. An appropriate message will be given to the Consumer Provisioning Agent if the request cannot be granted. If the consumer is not allowed to provision another Trusted Systems domain, the consumer will have to correct the underlying issue(s) which could require renegotiating their contract with the provider or selecting another provider.

Related Patterns

The consumer will have to establish a trusted context with the provider and exchange information between trusted parties, the provider and the consumer, in order to provision a Trusted Systems Domain. Once the domain is provisioned, the consumer will have to establish a trusted context with the newly provisioned Trusted Systems Domain in order to operate on it and use the TSD. Once the Trusted Context is established the consumer can use the other patterns in the TMI to manage and exploit the Trusted Systems Domain

Related Use Cases

The following use cases are directly related to provisioning a Trusted Systems Domain:

Generic: UC-2

Provider:

Consumer: UC-5

The following use cases are indirectly related to provisioning of Trusted Systems Domains:

Generic: UC-1, UC-3, UC-4, UC-5, UC-6, UC-7, UC-8, UC-10

Provider: UC-3, UC-4, and UC-5

Consumer: UC-3 and UC-4

3.6.2 Provisioning a dedicated Asset

Synopsis

From the use cases some examples of dedicated assets that can be provisioned are the: Consumer Management Agent, Server, Storage volume, Peripheral Device, Application Components, Consumer Audit Agent, and Consumer Centralized Audit Collection Environment. Provisioning services are used to create, operate on, or destroy assets associated with Trusted Systems Domains. The consumer should first provision a Trusted Systems Domain. Next, a trusted channel should be established with the new Trusted Systems Domain, finally, the policy store of the Trusted Systems Domain should be populated before any other assets can be provisioned. Once these steps have been completed other assets can be provisioned to the Trusted Systems Domain. The Trusted Systems Domain which is to contain the new dedicated asset should be indicated in the request.

Context

After a Trusted Systems Domain is created, any assets that are required for the TSD to function properly should be provisioned. While operating a Trusted Systems Domain may discover that it needs additional assets or that it no longer needs assets. When a Trusted Systems Domain is no longer needed, the remaining assets should be returned to the provider. The requestor can be the consumer or an agent acting on behalf of the consumer.

Selection Criteria

This pattern will be used when an asset should be provisioned that will not be shared; the new asset will be completely under the control of the trusted systems domain. It is assumed that a trusted channel has been established between the requestor and the provider. The Trusted Systems Domain that is to contain the new dedicated asset should be indicated on the request. The policies associated with the Trusted Systems Domain should allow the creation of the requested asset. The provider cannot check that the requestor's policy allows the allocation of the asset. Ideally, if there are no policies governing this type of asset in the Trusted Systems Domain Policy Store, the request should fail (the provider cannot enforce this). If the provider's policy does not allow the allocation, the request will fail. The metadata associated with the Trusted Systems Domain is updated to contain the new asset if the request is successful. Once the asset is provisioned the requestor should establish a trusted context with the new asset to validate and manage it.

Solution

1. The **Requestor** checks that their policy allows creation of the requested **Asset**. If it does not allow creation of the **Asset**, the request fails.
2. If it is allowed to create the **Asset**, the **Requestor** requests that the **Provider Management Agent** allocates a new **Asset** in the indicated **Trusted Systems Domain**.
3. The **Provider Management Agent** checks the **Provider Policy Store**
    a. To see if there are policies governing this asset type.
    b. To check that an additional **Asset** of this type is allowed.

   If either of these conditions fails, the **Requestor** is notified.
4. The **Provider Management Agent** checks the **Provider Asset Repository** to see if an **Asset** of this type which meets the required policies is available.
5. If the **Provider Management Agent** is allowed to fulfill the request
    a. The **Provider Management Agent** provisions the asset to the indicated Trusted Systems Domain.

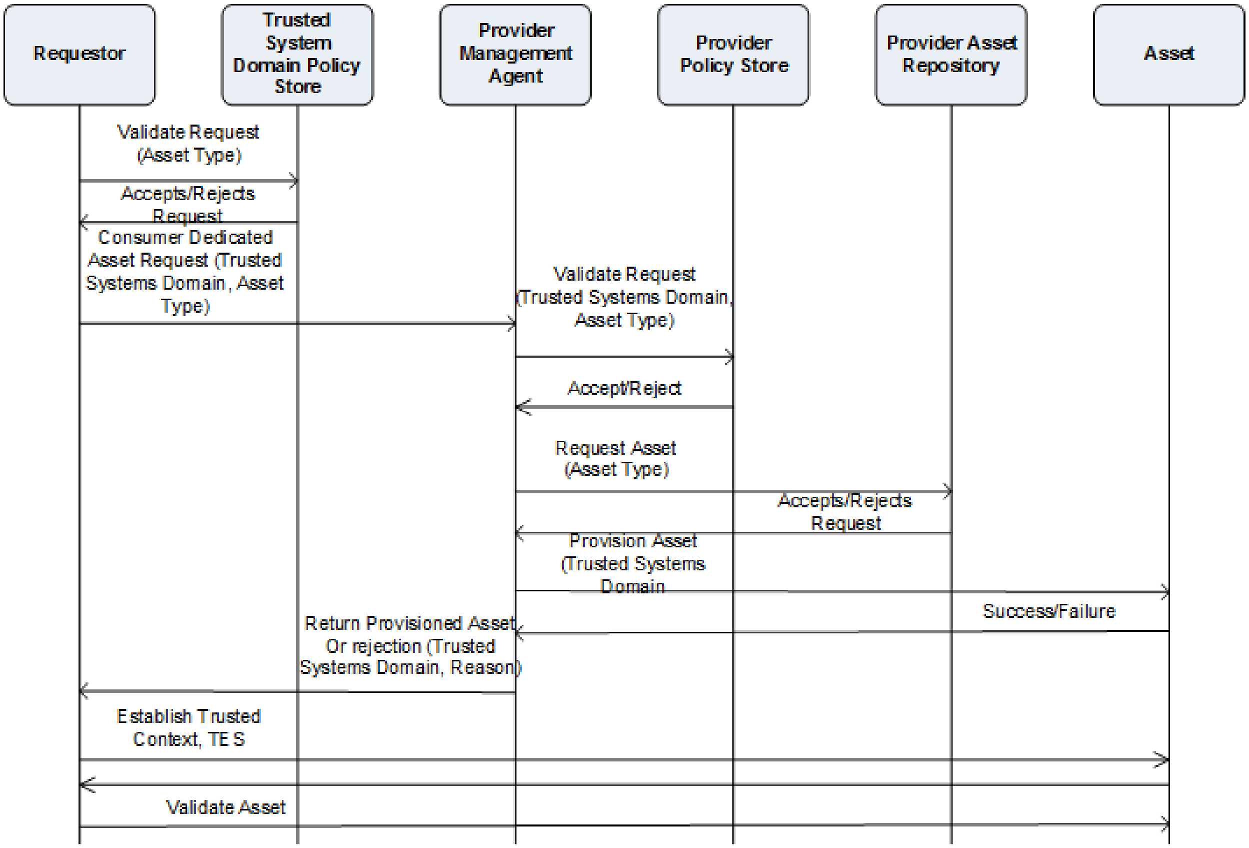
   b. The **Provider Management Agent** establishes trusted context for the new **Asset** the Provider Trusted Entity Store is updated to indicate the assignment of the **Asset**.

5. If the request is not fulfilled for any reason the **Provider Management Agent** notifies the **Requestor**

6. If the request is fulfilled the **Provider Management Agent** notifies the **Requester** and returns the **Asset**.

7. If the request was fulfilled, the *Requestor* Management Agent establishes trusted context for the new **Asset** and the *Requestor* Trusted Entity Store is updated to indicate the presence of the **Asset**

8. If the **Requestor** finds a problem with the **Asset**, it is returned to the **Provider Management Agent**.


Requestor
Trusted System Domain Policy Store
Provider Management Agent
Provider Policy Store
Provider Asset Repository
Asset
Validate Request (Asset Type)
Accepts/Rejects Request
Consumer Dedicated Asset Request (Trusted Systems Domain, Asset Type)
Validate Request (Trusted Systems Domain, Asset Type)
Accept/Reject
Request Asset (Asset Type)
Accepts/Rejects Request
Provision Asset (Trusted Systems Domain
Success/Failure
Return Provisioned Asset Or rejection (Trusted Systems Domain, Reason)
Establish Trusted Context, TES
Validate Asset

Implications

The consumer should populate the Trusted Systems Domain Policy Store with policies governing all the assets that will be provisioned to the Systems Domain before those assets are provisioned. (This is a self-enforced constraint.) The provider's policy should allow provisioning of the asset to the Trusted Systems Domain indicated by the consumer.

The asset type should be one of: Consumer Management Agent, Server, Storage Volume, Peripheral Device, Application Components, Consumer Audit Agent, or Consumer Centralized Audit Collection Environment

Related Patterns

The consumer should have established a trusted context with the provider and enabled the exchange of information between trusted parties, the provider and the consumer, in order to initially provision the Trusted Systems Domain. Once the domain is provisioned, the consumer will have to establish a trusted context with the newly provisioned Trusted Systems Domain in order to operate on and use the TSD. Once the Trusted Context is established the consumer can use the other patterns in the TMI to manage and exploit the Trusted Systems Domain. The consumer will have to assure through the use of Management and Monitoring services that the provisioning action will not disrupt the function of the Trusted Systems Domain.

Related Use Cases

The following use cases are directly related to provisioning a dedicated asset:

Generic: UC-1, UC-4, UC-5, UC-8, and UC-10

Consumer: UC-5

The following use cases are indirectly related to provisioning a dedicated asset:

Generic: UC-2, UC-3,

Provider: UC-3, UC-4, and UC-5

Consumer: UC-3, UC-4, and UC-5

3.6.3 Provisioning a Shared Asset

Synopsis

Examples of shared assets that can be provisioned include a Communications Channel and a Data Exchange Gateway. For shared assets, both parties should provision the asset and the policies governing the asset should be consistent (or match) in order for the asset to function properly. The asset will not become active until the second party provisions the asset. The asset only operates within the scope of each party's policies.

Context

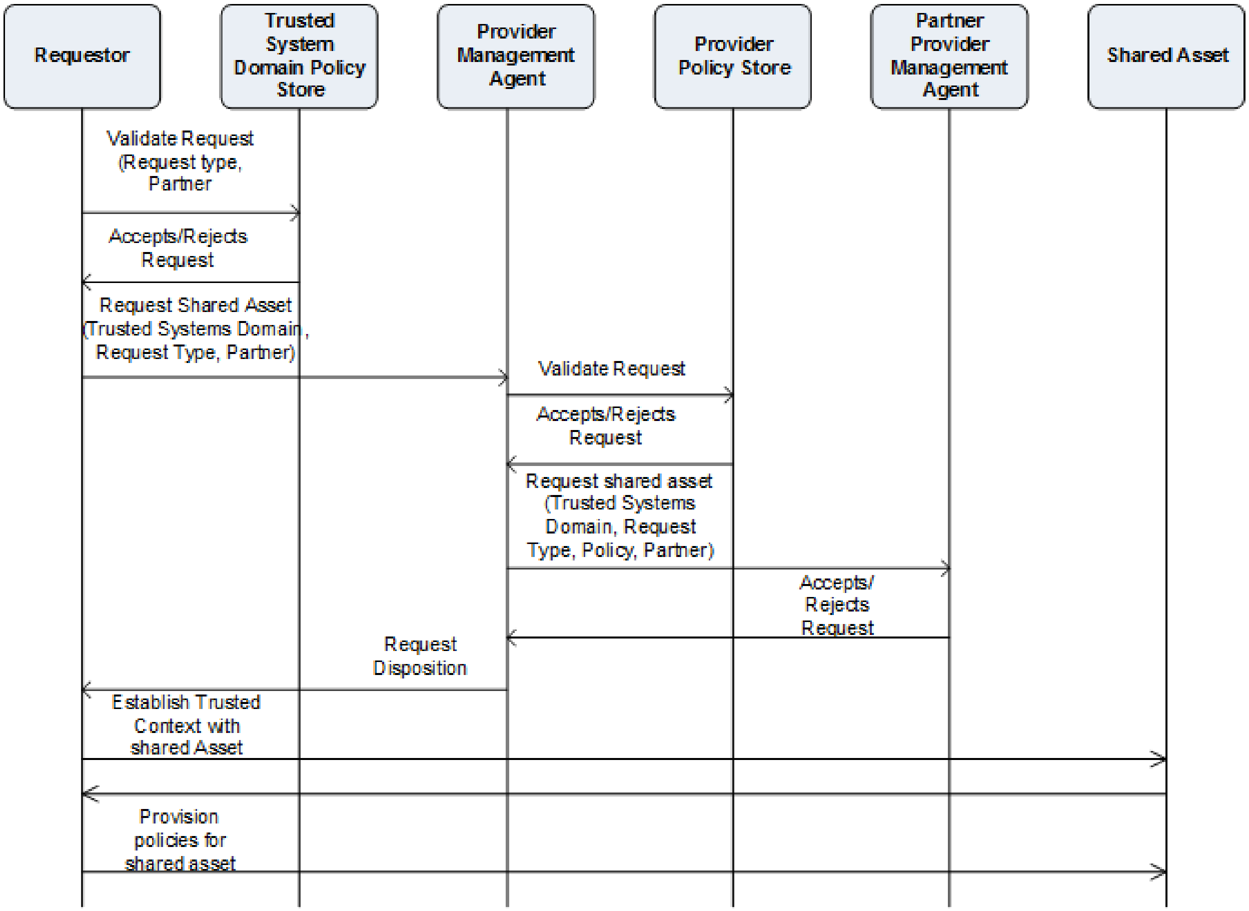
Each party provisioning a shared asset is authorizing their Trusted Systems Domain to share the asset with another party within the scope of its policies. The party could be another Trusted Systems Domain, or some arbitrary system such as one represented by a URL/UUID. If the other party is another Trusted Systems Domain, then that domain should also provision the asset for sharing to occur. If the indicated partner is outside the auspices of the provider, then provisioning the shared asset explicitly authorizes communications to/from that partner.

Selection Criteria

This pattern is used when there is a need to share an asset with another party. Sharing will not occur unless both parties "provision" the asset. It is assumed that a trusted channel has been established between the requestor and the provider. The Trusted Systems Domain that is provisioning the new shared asset should be indicated on the request. The policies associated with the Trusted Systems Domain should allow the creation of the requested asset. The provider cannot check that the requestor's policy allows the allocation of the asset. Ideally, if there are no policies governing this type of asset in the Trusted Systems Domain Policy Store, the request should fail (the provider cannot enforce this). If the provider's policy does not allow the creation of the shared asset, the request will fail and the requestor will be notified. The metadata associated with the Trusted Systems Domain is updated to contain the new asset if the request is successful.

Solution

1. The **Requestor** checks the **Trusted Systems Domain Policy Store** to see if the shared asset is allowed.
    a. If the shared request is not allowed the **Requestor** notifies the owner of the Trusted Systems Domain (this check is self-enforcing).
2. The **Requestor** provisions the policies that will govern the shared Asset.
3. The **Requestor** asks the **Provider Management Agent** to provision the shared Asset
    a. The **Requestor** should indicate the Trusted Systems Domain, the request type, and the partner or partners that will share the Asset.
4. The **Provider Management Agent** validates the request against the policies in the **Provider Policy Store**.
    a. If the request is not valid the **Requestor** will be notified.
5. If policies allow the shared Asset to be provisioned the **Provider Management Agent** requests that the ***Partner*** **Provider Management Agent** also provision the shared asset.
6. The ***Partner*** **Provider Management Agent** accepts or rejects the request
7. The **Provider Management Agent** notifies the requestor of the disposition of the request.
8. If the request was accepted the *Requestor* Management Agent establishes a trusted context with the shared Asset and updates the *Requestor* Trusted Entity Store.
9. The **Requestor** provisions polices associated with the shared Asset.


Requestor
Trusted System Domain Policy Store
Provider Management Agent
Provider Policy Store
Partner Provider Management Agent
Shared Asset
Validate Request (Request type, Partner
Accepts/Rejects Request
Request Shared Asset (Trusted Systems Domain, Request Type, Partner)
Validate Request
Accepts/Rejects Request
Request shared asset (Trusted Systems Domain, Request Type, Policy, Partner)
Accepts/ Rejects Request
Request Disposition
Establish Trusted Context with shared Asset
Provision policies for shared asset

Implications

If a shared asset is provisioned the requestor can start using it immediately. However, some shared assets, such as a communications channel, may not properly work until at least one other party provisions the asset. Provisioning a shared asset with a party that is outside the auspices of the provider explicitly allows communication with that partner. Communications channels are shared objects. Provisioning of a communications channel configures the providers systems so that they will permit communications with the indicated partner(s) within the scope of each party's policies. This allows the requestor to initiate communications or wait for the partner(s) to initiate. The communications policies established when the Systems Domain and the Trusted Systems Domain were provisioned, or subsequent modifications to those policies will determine whether a communications channel can be established.

A multi-party asset may not be deprovisioned until the last party deprovisions the asset. The shared asset will only function if there are no conflicts between the policies associated with the shared asset.

Related Patterns

The consumer should have established a trusted context with the provider and enabled the exchange of information between trusted parties, the provider and the consumer, in provision the shared asset. The requestor should have already provisioned the Trusted Systems domain which is to contain the shared asset. Once the shared asset is provisioned, the consumer will have to establish a trusted context with the newly provisioned shared asset in order to set it policies and use the TSD. Once the Trusted Context is established the consumer can use the other patterns in the TMI to manage and exploit the shared asset. The consumer will have to assure through the use of Management and Monitoring services that the provisioning action will not disrupt the function of the Trusted Systems Domain.

Related Use Cases

The following use cases are directly related to provisioning a shared asset (communications channel:

Generic: UC-1 and UC-6

Provider: UC-2

Consumer: UC-5

The following use cases are indirectly related to provisioning a communications channel:

Generic: UC-2, UC-3,

Provider: UC-3, UC-4, and UC-5

Consumer: UC-3, and UC-4