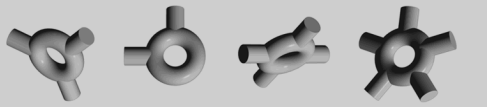
elements



Permissions Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

Contents

List of Figures

List of Tables

Package Description

This package is used to enable entities to have permission to perform restricted operations. In the field of permissions, such an entity is conventionally referred to as a “principal”, however it was decided to use the name “permission owner” to avoid ambiguity with the financial term “principal”. An example of a permission owner is a user. An example of an operation is the ability to trade a particular deal.

1 Use Cases

1.1 Authenticate Passphrase

Main flow of events:
The system prompts the user to enter their username and their passphrase. The user enters their username and passphrase via the keyboard and hits “enter” or clicks “accept”. The system verifies that the combination of user and passphrase is valid. If the combination of user and passphrase is valid then the system acknowledges this and the user becomes the current permission owner.
Exceptional flow of events:
The user can cancel the authentication process at any time by pressing the Cancel button, thus restarting the use case.
Exceptional flow of events:
If the user enters an invalid username and passphrase combination then the use case restarts. If this happens three times in a row, the system cancels the authentication procedure, and prevents user access for 60 seconds.

1.2 Perform Operation

Background flow of events: Upon opening a user interface, the program will inquire of the current PermissionOwner (in this case a user) about whether it is allowed to perform various operations relevant to the interface. The program will then disable those parts of the user interface that the user is not allowed to perform. This will help prevent attempts to perform operations that the current user does

not have privilege to perform. (Note: this mechanism is not the primary technique used to prevent unauthorized access.)

Main flow of events: The user interacts with the interface, instructing it to perform some operation. At the very beginning of the operation, the "attemptingToDo" method on the current PermissionOwner is called with the object that represents this operation. The operation is then performed.

Exceptional flow of events: The "attemptingToDo" method raises a AuthorizationException, and the execution of the restricted operation is aborted.

2 Actors

2.1 User

This actor is a user of the system. A user will be a permission owner.

2.1.1 Relationships

	Class	Description	Notes
↔	Authenticate Passphrase §1.1		→
↔	Perform Operation §1.2		→

↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3 Interfaces

3.1 AccessLogger

This interface is used to receive raw data and record this data appropriately.

3.1.1 Relationships

	Class	Description	Notes
↓	NullAccessLoggerModel §4.2		
↓	StandardAccessLoggerModel §4.11		
↔	PermissionOwnerModel §4.10	accessLogger	

↓:Realized by ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3.1.2 Operations

log(PermissionOwner permissionOwner, Operation operation, Boolean allowed) log

permissionOwner: PermissionOwner The permission owner who has attempted an operation.
operation: Operation The operation that was attempted.
allowed: Boolean A value saying whether the permission owner is allowed to perform the operation.

This method takes in a permissionOwner, an operation and an allowed flag, and determines what information should be sent to the associated auditor.

3.2 AuthenticationLogger

This interface defines a log method that is used to recieve raw data (regarding attempted authentication processes) and to possibly log this data.

3.2.1 Relationships

	Class	Description	Notes
↓	StandardAuthenticationLogger-Model §4.12		
↓	NullAuthenticationLoggerModel §4.3		
↔	PermissionOwnerModel §4.10	authentication-Logger	

↓:Realized by ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3.2.2 Operations

log(PermissionOwner permissionOwner, Boolean authenticated) log

permissionOwner: PermissionOwner The permission owner who an entity is claiming to be.
authenticated: Boolean Whether or not the entity claiming to be a permissionGrant was authenticated.

This method takes in a permissionOwner and an authorized flag, and determines what information should be sent to the associated auditor.

3.3 Authenticator

This interface captures the general concept of the authentication of permission owners. Different permission owners may require different authentication procedures.

The process of authenticating a software component is handled via cryptography. The combination of a digital signature and a certificate will authenticate a software component.

3.3.1 Relationships

	Class	Description	Notes
⇓	Passphrase §3.4		
↔	PermissionOwnerModel §4.10	authenticator	

⇓:Inherited by ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3.3.2 Operations

Boolean authenticate(Object authenticationObject) authenticate

authenticationObject: Object This is an object which will be used to determine if the alleged permission owner is really the permission owner it claims to be.

This method is used to determine if an entity can provide the appropriate information that will identify it as a legitimate permission owner.

3.4 Passphrase

This interface enables the verification of passphrases submitted by a permission owner, for example, a user typing in his passphrase. This is accomplished by comparing the submitted passphrase with the passphrase that is currently assigned to the permission owner.

During the initialization of a passphrase object, a random number is generated and stored. This number is set only once for the life of the object. This random number is used when the passphrase is set and during verification of passphrases.

3.4.1 Relationships

	Class	Description	Notes
⇑	Authenticator §3.3		
↓	PassphraseModel §4.7		

⇑:Inherits ↓:Realized by

3.4.2 Operations

Date passphraseLastChanged() passphrase-LastChanged

The date at which this passphrase was last changed.

setPassphrase(String passphrase) setPassphrase

passphrase: String The raw passphrase that will be set as the permission owner's passphrase.

This method calls passphraseIsHardToGuess with the given (raw) passphrase. If the given passphrase fails this test then an InvalidPassphraseException is raised. Otherwise this method encodes the given (raw) passphrase in an appropriate manner for storage.

This method takes a passphrase and combines it with the held randomNumber by appending the lowest order 8 hexadecimal digits (where hexadecimal letters are written in uppercase) of the random number to the front of the passphrase string. The combination of passphrase and the random number are then hashed using SHA-1 [2], and the result is stored in the encryptedPassprhase attribute. This method then updates the passphraseLastChanged variable to be the current date.

Boolean passphaseIsHardToGuess(String passphrase) passpha-seIsHard-ToGuess

passphrase: String The passphrase that will be checked to see if it is hard to guess.

This method tests the passphrase to ensure that it is not "too easy" to guess. Tests could include minimum length, inclusion of upper and lower case characters, inclusion of non-alphabetic characters, avoidance of repeated characters, avoidance of common names and difference from previous passphrases. The following reference gives guidence on what constitutes a passphrase being hard to guess. [1]. The number of these tests performed and the strictness with which they are enforced is determined by the adopted security policy. This method returns true if it passes these tests, otherwise false.

3.5 Operation

This class is used to specify operations. An operation is defined as an action on a particular object. For example, changing a deal would be an operation whose action is “modify” and whose object is a deal.

In fact, we specify that an operation consists of three pieces of data: an “action”, a “type” and an “object”. The “action” is an action that is being applied to the object, for example to “open” a window, or “resize” a window. The “type” specifies the general class of the operation (such as operations on deals, operations on reference data or window operations), and is required when comparing two operations to ensure that their objects are comparable. The “object” specifies the particular object to which the operation applies (e.g. FX deals in a particular book).

Operations are used as part of the permission model: A permission owner will have the authority to perform certain specified operations. For example, a given user could have permission to create FX deals in a given book, modify loans in a different book, and create bond definitions in reference data.

3.5.1 Relationships

	Class	Description	Notes
↓	OperationModel §4.5		

↓:Realized by

3.5.2 Operations

OperationType type() type

The ‘type’ of an operation specifies the general category into which the operation falls. For example, operations may have types such as ‘operations on deals’, ‘operations on reference data’, ‘window operations’.

The ‘type’ is required when comparing two operations, to ensure that their objects have the same structure, and are thus comparable. For example an operation of type ‘DealOperationType’ will always have a object that is specified with the keys “Book”, “Counterparty”, “DealType”, etc.

OperationAction action() action

This returns an OperationAction, which is used to specify the action being applied to an object. For example, the action may be ‘enter’ or ‘modify’ (a deal), or ‘open’ or ‘move’ (a window). The available actions will be determined by the

operation's type, since, obviously, not all actions are applicable to every class of object.

Classifier object() object

Returns the associated classifier. This classifier will specify the objects to which this operation applies. For example, if the operation applies to FX and Loan deals in books bk1 or bk2, then the specifier would be:

'DealType' -> FXDeal, LoanDeal 'Book' -> bk1, bk2

3.6 OperationType

An OperationType is used to broadly characterize operations based on the "kind of object" to which they apply. For example, deals, reference data, windows etc.

The type of an operation will determine two things: The specific actions that can be performed (you can't modify a window or resize a deal), and the broad structure of the Classifier object that specifies the actual objects to which the operation applies. For example a deal operation can have the following kind of actions: "Create", "Read", "Update" and "Delete", applied to an object that can be specified with the following kind of values "book", "counterparty", "dealType", etc.

3.6.1 Relationships

	Class	Description	Notes
⇑	Identifiable		
↓	OperationTypeModel §4.6		
↔	OperationModel §4.5	type	
↔	PermissionGrantModel §4.8	operationType	

⇑:Inherits ↓:Realized by ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

3.6.2 Operations

Set<OperationAction> appropriateActions() appropriateActions

Returns a collection of OperationActions that are appropriate to the specific operation type.

3.7 PermissionGrant

A PermissionGrant is used to grant permission of a certain operation or group of operations to a permission owner.

3.7.1 Relationships

	Class	Description	Notes
⇑	Identifiable		
⇑	Validatable		
⇓	CompositePermissionGrant §3.8		
↓	PermissionGrantReferenceData §4.9		
↓	PermissionGrantModel §4.8		
↓	NullPermissionGrantModel §4.4		
↔	PermissionOwnerModel §4.10	permissionGrant	
↔	PermissionGrantReferenceData §4.9	model	
↔	CompositePermissionGrant-Model §4.1	collection	◊

⇑:Inherits ⇓:Inherited by ↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

3.7.2 Operations

Boolean allowedToDo(Operation operation) allowedToDo

operation: Operation The operation which may be allowed to be performed by this permission grant.

This method takes an operation and determines whether this permission grant allows it to be performed.

Collection<Object> allowedValues(Operation operation, String key) allowedValues

operation: Operation This is the operation that must be allowed by a permission grant before this method will return a list of values corresponding to the key parameter.

key: String This key specifies a particular aspect of an object. All allowed values of that aspect will be returned from this method.

This method returns a list of values that are allowed to be used in conjunction with the given operation. This list of values comes from the category that is determined by the given key. If the operation is not allowed then return an empty collection. If the key is not present then return an empty collection.

For example, when a user opens an FX deal entry screen we will want to know the list of counterparties which whom the user can trade. To obtain this list we create an operation object with the type “Deal Operation”, the action “create”, and a classifier that has the aspect “deal type” with a corresponding value “FX”.

We then ask the current permission owner what are your allowedValues given this operation, and the key “counterparties”. See also PermissionOwner §3.9 and PermissionGrantModel §4.8.

3.8 CompositePermissionGrant

This interface is a type of permission grant that is composed of a number of other permission grants. This permission grant allows an operation if any of the permission grants that it is composed of allows the operation. The act of grouping permission grants together enables groups of permissions to be allocated to people. Permission grants can be grouped together to form a particular role. Thus this class provides the ability to manage permission in a role-based manner.

3.8.1 Relationships

	Class	Description	Notes
⇑	PermissionGrant §3.7		
↓	CompositePermissionGrant-Model §4.1		

⇑:Inherits ↓:Realized by

3.8.2 Operations

Boolean allowedToDo(Operation operation) allowedToDo

operation: Operation The operation which may be allowed to be performed by this permission grant.

This method determines whether the given operation is allowed by this permission grant.

If any of the permission grants in the held collection return true when asked whether they are “allowToDo” the given operation then return true, otherwise false.

Collection<Object> allowedValues(Operation operation, String key) allowedValues

operation: Operation This is the operation that must be allowed by this permission grant before this method will return a list of allowed values.

key: String This key specifies a particular aspect of an object. All allowed values of that aspect will be returned from this method.

This method calls the same method with the same arguments on all of the PermissionGrants in its held collection and combines all the returned collections. This

method returns the combined collection.

Collection<PermissionGrant> collection() collection

The collection of permission grants that makes up this composite permission grant.

add(PermissionGrant permissionGrant) add

permissionGrant: PermissionGrant The permissionGrant to be added to the CompositePermissionGrant's collection.

Adds the given permissionGrant to this object's collection.

Reportable validate() validate

This method returns a Reportable composed of all the Reportables returned from the calling the "validate" method on the permission grants in this interface's collection.

3.9 PermissionOwner

A permission owner is an entity that can have permission to perform a restricted operation. Such an entity is conventionally referred to as a "principal", however it was decided to use the name "permission owner" to avoid ambiguity with the financial term "principal". An example of a permission owner is a user. This interface provides authentication and authorization procedures.

At places where authorization is required before an operation can be performed, the current permission owner is asked whether it can perform the desired operation. This is done by sending the "allowedToDo" method to the current PermissionOwner with the desired operation as the parameter. See Operation §3.5.

Any appropriate object can become a permission owner. This can be achieved in two ways. Either the object can inherit from the PermissionOwnerModel, or it can realize the PermissionOwner interface and delegate to an associated realization of the PermissionOwner interface.

3.9.1 Relationships

	Class	Description	Notes
⇑	Identifiable		
⇑	Validatable		
↓	PermissionOwnerModel §4.10		

⇑:Inherits ↓:Realized by

3.9.2 Operations

Boolean allowedToDo(Operation operation) allowedToDo
operation: Operation

This method determines whether this permission owner is permitted to perform the given operation. The operation is permitted if this permission owner “isActive” and if any of the permission grants, that are returned from calling the permission grants method, allow the given operation. The allowedToDo method on the permission grant is called to determine whether it allows a given operation.

Collection<Object> allowedValues(Operation operation, String key) allowedValues
operation: Operation This is the operation that must be allowed by a permission grant before it will return a list of values corresponding to the key parameter.
key: String This is the key that specifies what kind of values are being sought from the permission grant.

This method is used to generate a list of allowed values (the nature of which is determined by the given key) that can be used to make a more specific operation of the same structure as the given one. For example if a user opens an FX deal screen, we will want a list of all the counterparties that the user can trade. To obtain this list we need to send the “allowableValues” method to the user with two parameters. The first parameter is an operation with the following three parts:
1) the dealOperationType (Which can be obtained by sending the dealOperationType method to the OperationTypeModel),
2) the “create” action. (Which can be obtained by sending the “create” method to the DealOperationAction class), and
3) a classifier containing a key that is the string “Deal types” and corresponds to a value that is the class FXTypeModel. The second parameter to be send with the “allowableValues” method is the string "Counterparty".

attemptingToDo(Operation operation) attemptingToDo
operation: Operation

This method is called when a permission owner is attempting to perform an operation. This method calls the allowedToDo method with the given operation. Then it calls the log method on the associated logger with itself, the operation, and the value returned from the allowedToDo method. If the allowedToDo method returns false then this method raises an AuthorizationException (after the logging has occurred).

authenticate(Object authenticationObject) authenticate

authenticationObject: Object This is an object that will be used to determine if the alleged permission owner is really the permission owner it claims to be.

This method is called when an entity is attempting to gain recognition as this permission owner. If this object's "isActive" method returns false, then the attempt to be recognized as this permission owner is considered false. If "isActive" is true then this method calls the "authenticate" method on the object returned from calling the "authenticator" method. After determining whether the authentication process was successful this method sends a "log" message to the authenticationLogger, with itself, and a boolean saying whether the authentication was successful or not. If the authentication was not successful then this method raises an AuthenticationException after the logging has occurred.

Boolean isActive() isActive

This method returns a boolean flag which determines whether this PermissionOwner is "active" or "non-active". An active PermissionOwner will function in a normal capacity. A non-active PermissionOwner prevents the represented entity from performing any restricted operations. No entity can authenticate itself as a non-active PermissionOwner.

Two examples of when a PermissionOwner would be deactivated are:
1) if it is suspected that its authentication procedure has been compromised, or
2) if the integrity of the entity represented by the PermissionOwner has come into question.

Authenticator authenticator() authenticator

Returns the associated Authenticator object.

AuthenticationLogger authenticationLogger() authentication-Logger

Returns the associated AuthenticationLogger, which will be able to provide various logging behavior for when an entity attempts to gain recognition as this permission owner.

AccessLogger accessLogger() accessLogger

Returns an associated accessLogger, which will be able to provide various logging behavior for when a PermissionOwner attempts to do something.

PermissionGrant permissionGrant() permission-Grant

This method returns the collection of permission grants that are used in deter-

mining whether a permission owner is allowed to perform an operation.

Reportable validate() validate

This method returns an object of type Reportable which will contain all the errors and warnings generated from this method. Below is a list of the axioms for an instance of a class that realizes this interface to be valid. Each statement is followed by the error or warning message that will be issued if the axiom is violated.

- This interface must have an authenticator. If not, then issue a "Permission owner has no authenticator" error.
- This interface must have an permission grant. If not, then issue a "Permission owner has no permission grant" error.
- This interface must have an authenticationLogger. If not, then issue a "Permission owner has no authenticationLogger" error.
- This interface must have an accessLogger. If not, then issue a "Permission owner has no accessLogger".

This method also calls the validate method on the permissionGrant.

4 Classes

4.1 CompositePermissionGrantModel

This class is a concrete realization of the CompositionPermissionGrant interface.

4.1.1 Relationships

	Class	Description	Notes
↑	CompositePermissionGrant §3.8		
↔	PermissionGrant §3.7	collection 0..n	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.1.2 Attributes

identifier: String The identifier of the composite permission grant.

4.2 NullAccessLoggerModel

This class provides the ability not to log information about attempted access to restricted operations. This class has only one instance.

4.2.1 Relationships

	Class	Description	Notes
↑	AccessLogger §3.1		

↑:Realizes

4.2.2 Operations

log(PermissionOwner permissionOwner, Operation operation, Boolean allowed) log

permissionOwner: PermissionOwner The permission owner who has attempted an operation.
operation: Operation The operation that was attempted.
allowed: Boolean A value saying whether the permission owner is allowed to perform the operation.

This method does nothing.

«Static Method» NullAccessLoggerModel getInstance() getInstance

This method returns the single instance of this class.

4.3 NullAuthenticationLoggerModel

This class provides the ability to log no information about attempted authentication. This class has only one instance.

4.3.1 Relationships

	Class	Description	Notes
↑	AuthenticationLogger §3.2		

↑:Realizes

4.3.2 Operations

log(PermissionOwner permissionOwner, Boolean authenticated) log

permissionOwner: PermissionOwner The permission owner who an entity is claiming to be.
authenticated: Boolean Whether the entity claiming to be a permissionGrant was authenticated.

This method does nothing.

«Static Method» NullAuthenticationLoggerModel getInstance() getInstance

This method returns the single instance of this class.

4.4 NullPermissionGrantModel

This class has a single instance, which is used when a permission owner has no permissions. This PermissionGrant grants no permissions.

4.4.1 Relationships

	Class	Description	Notes
↑	PermissionGrant §3.7		

↑:Realizes

4.4.2 Operations

String identifier() identifier

Returns the string “Null Permission Grant”.

Boolean allowedToDo(Operation operation) allowedToDo
operation: Operation The operation which may be allowed to be performed by this permission grant.

This method returns false. (This grant does not allow any permissions).

Collection<Object> allowedValues(Operation operation, String key) allowedValues
operation: Operation This is the operation that must be allowed by a permission grant before this method will return a list of values corresponding to the key parameter.
key: String This key specifies a particular aspect of an object. All allowed values of that aspect will be returned from this method.

As this permission grant does not allow any operation this method simply returns an empty collection.

«Static Method» NullPermissionGrant getInstance() getInstance

Returns the single instance of this class.

Reportable validate() validate

Returns the NullReportable.

4.5 OperationModel

This class is a concrete realization of the Operation interface. An operation is defined as an action applied to an object. In this implementation the object of the operation is specified by a classifier.

4.5.1 Relationships

	Class	Description	Notes
↑	Operation §3.5		
↑	Validatable		
↔	Classifier	object	→
↔	OperationType §3.6	type	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.5.2 Attributes

action: OperationAction

4.5.3 Operations

Reportable validate() validate

This method returns an object of type Reportable which will contain all the errors and warnings generated from this method. Below is a list of the axioms for an instance of this class to be valid. Each statement is followed by the error or warning message that will be issued if the axiom is violated.

- This operation's action must be contained within the operation's type's appropriateActions. If not, then this method creates the following error message: "Operation has an invalid action named "<id1>".", where <id1> is the identifier of the operation.

Note that additional validation could be performed if the OperationType was to hold onto a collection of keys that a collection of a given type can have. This

collection would contains the keys that a classifier of an operation of a particular type could have. Then this validation method could make sure that the keys of it classifier are allowed by the held operation type.

4.6 OperationTypeModel

This class is a concrete realization of the OperationType interface. This class has a fixed number of instances. Each instance will be accessible via its own static method on this class.

4.6.1 Relationships

	Class	Description	Notes
↑	OperationType §3.6		

↑:Realizes

4.6.2 Attributes

identifier: String This identifies the operation type.

appropriateActions: Set<OperationAction> Returns a set of OperationActions. Every instance of this class will have a unique collection of operation actions.

4.6.3 Operations

«Static Method» OperationTypeModel dealOperationType() dealOperationType

This method returns a particular instance of this class that has the identifier "Deal Operation Type", and a set of appropriateActions that consists of the objects returned from calling the following methods on the OperationAction class: "create", "browse", "modify" and "cancel". Classifiers associated with this type are likely to have the following keys: "books", "counterparties", "dealType", "dealPurpose".

«Static Method» OperationTypeModel screenOperationType() screenOperationType

This method returns a particular instance of this class that has the identifier "Screen Operation Type", and a set of appropriateActions that consists of the objects returned from calling the following methods on the OperationAction class:

"open". Classifiers associated with this type are likely to have the following keys: "screen name".

referenceDataOperationType

«Static Method» OperationTypeModel referenceDataOperationType()

This method returns a particular instance of this class that has the identifier "Reference Data Operation Type", and a set of appropriateActions that consists of the objects returned from calling the following methods on the OperationAction class: "create", "browse", "modify" and "delete". Classifiers associated with this type are likely to have the following keys: "reference data".

namedOperationType

«Static Method» OperationTypeModel namedOperationType()

This method returns a particular instance of this class that has the identifier "Named Operation Type", and a set of appropriateActions that consists of the objects returned from calling the following methods on the OperationAction class: perform. The named operation type is quite generic as it is used to specify operations based only upon its name. Classifiers associated with this type are likely to have the following keys: "operation name".

passwordOperationType

«Static Method» OperationType passwordOperationType()

This method returns a particular instance of this class that has the identifier "Password Operation Type", and a set of appropriateActions that consists of the object returned from calling the following method on the OperationAction class: "modify". Classifiers associated with this type are likely to have the following keys: "user".

4.7 PassphraseModel

This class is a concrete realization of the Passphrase interface. It includes some implementation detail on how the passphrases are stored.

4.7.1 Relationships

	Class	Description	Notes
↑	Passphrase §3.4		

↑:Realizes

4.7.2 Attributes

encryptedPassphrase: String Holds the encrypted passphrase of this user.

randomNumber: Integer Holds a number that was randomally generated upon instantiation of this object.

passphraseLastChanged: Date Holds the date at which this passphrase was last changed.

4.7.3 Operations

Boolean authenticate(Object authenticationObject) authenticate

authenticationObject: Object This is an object that will be used to determine if the alleged permission owner is really the permission owner it claims to be.

This method calls verifyPassphrase with the authenticationObject, and returns that result.

Boolean verifyPassphrase(String passphrase) verify-Passphrase

passphrase: String A string that has been entered by a human user and will be compared against the passphrase held by this interface to determine whether it matches.

Combine the given passphrase with the held random number (according to the method outlined in the setPassphrase documentation). Hash the combined string with SHA-1 and compare against the encryptedPassphrase. This method returns true if they match, otherwise it returns false.

String encryptedPassphrase() encrypted-Passphrase

Returns the encrypted passphrase of this user.

integer randomNumber() randomNumber

Returns a number that was randomly generated during the initialization of this object. This random number is generated only once for the life of this object. This random number should be between 0 and $2^{32} - 1$. A different random number is kept with each passphrase object in order to make brute force passphrase attacks harder.

4.8 PermissionGrantModel

This is a concrete realization of the PermissionGrant interface.

4.8.1 Relationships

	Class	Description	Notes
↑	PermissionGrant §3.7		
↔	Classifier	operationObject	→
↔	OperationType §3.6	operationType	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.8.2 Attributes

identifier: String This string identifies this permission grant.

operationActions: Collection<OperationAction> This attribute specifies the "actions" authorized by the permissionGrant, for example, "modify" (a deal) or "open" (a window). The attribute contains a collection of OperationActions, which are the actions of an operation that this PermissionGrant allows. See Operation §3.5.

4.8.3 Operations

Boolean allowedToDo(Operation operation) allowedToDo

operation: Operation The operation which may be allowed to be performed by this permission grant.

This method determines whether the given operation is allowed by this PermissionGrant.

For this method to return true the following three conditions must be met:
1) This object's operationType is the same as the given operation's "type",
2) The given operation's action exists within this object's operationActions set.
3) This object's operationObject "is a superset of" the given operation's object.

Collection<Object> allowedValues(Operation operation, String key) allowedValues

operation: Operation This is the operation that must be allowed by a permission grant before this method will return a list of values corresponding to the key parameter.

key: String This is the key that specifies what kind of values are being sought from the permission grant.

If this class is “allowedToDo” the given operation, then ask the classifier, returned from the “operationObject” method, for its value at the given key, and return that result. If the operation is not allowed then return an empty collection. If the key is not present then return an empty collection.

Classifier operationObject() operationObject

This method returns the associated classifier. An operation is defined as an action on an object. This classifier contains information on what kind of objects (of an operation) that this permission grant allows.

OperationType operationType() operationType

This method returns the associated OperationType, which is used to identify the “type” of the operation that this permission grant authorizes. For example, “deal operations” or “window operations”. See Operation §3.5 for an explanation of “Operations” and operation “types”.

Reportable validate() validate

This method returns an object of type Reportable which will contain all the errors and warnings generated from this method. Below is a list of the axioms for an instance of this class to be valid. Each statement is followed by the error or warning message that will be issued if the axiom is violated.

- This grant’s operationActions must be a subset of this grant’s type’s appropriateActions. For all invalid actions the following error message should be added to the Reportable that will be returned from this method: “Permission grant “<id1>” has an invalid action named “<id2>”.”, where id1 is the identifier of this permission grant, and id2 is the identification of the invalid action.

Note that additional validation could be performed if the OperationType was to hold on to a collection of keys that a collection of a given type can have. This collection would contain the keys that a classifier of an operation of a particular type could have. Then this validation method could make sure that the keys of its classifier are allowed by the held operation type.

4.9 PermissionGrantReferenceData

Implements the interface by delegating to the associated model.

4.9.1 Relationships

	Class	Description	Notes
⇑	ReferenceDataModel		
↑	PermissionGrant §3.7		
↔	PermissionGrant §3.7	model	→

⇑:Inherits ↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.10 PermissionOwnerModel

This class is a concrete realization of the PermissionOwner interface.

This class holds onto a single permission grant, which can be a composite permission grant.

4.10.1 Relationships

	Class	Description	Notes
↑	PermissionOwner §3.9		
↔	PermissionGrant §3.7	permissionGrant	→
↔	Authenticator §3.3	authenticator	→
↔	AccessLogger §3.1	accessLogger	→
↔	AuthenticationLogger §3.2	authentication-Logger	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.10.2 Attributes

identifier: String The identifier of this permission owner.

isActive: Boolean Is this permission owner active?

4.11 StandardAccessLoggerModel

This class provides an implementation of a particular auditing policy for information pertaining to attempted performing of restricted operations. The details of this class have not yet been defined due to its relationship with the currently undefined auditing package.

4.11.1 Relationships

	Class	Description	Notes
↑	AccessLogger §3.1		
↔	Auditor	auditor	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.11.2 Operations

log(PermissionOwner permissionOwner, Operation operation, Boolean allowed) log

permissionOwner: PermissionOwner The permission owner who is attempting to perform a restricted operation.
operation: Operation The operation that a permission owner is attempting to perform.
allowed: Boolean Whether the permission owner is permitted to perform the operation.

This method sends some information to the auditor. The information sent to the auditor is not defined because the auditing package has not yet been specified.

Auditor auditor() auditor

Returns the auditor that will have permission logging information sent to it. An Auditor is an object that receives and records arbitrary data.

4.12 StandardAuthenticationLoggerModel

This class provides an implementation of a particular auditing policy for information pertaining to attempted authentication. The details of this class have not yet been defined due to its relationship with the currently undefined auditing package.

4.12.1 Relationships

	Class	Description	Notes
↑	AuthenticationLogger §3.2		
↔	Auditor	auditor	→

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

4.12.2 Operations

log(PermissionOwner permissionOwner, Boolean authenticated) log
permissionOwner: PermissionOwner
authenticated: Boolean Whether the entity claiming to be a permissionGrant was authenticated.

This method sends some information to the auditor. The information sent to the auditor is not defined because the auditing package has not yet been specified.

Auditor auditor() auditor
Returns the associated auditor.

5 Exceptions

5.1 AuthenticationException

This exception is raised when a permission owner fails to correctly verify itself as the permission owner it claims to be.

5.2 AuthorizationException

This exception is raised when the current permission owner is not allowed to perform the operation being attempted.

5.3 InvalidPassphraseExeception

This exception is raised when a passphrase does not meet the requirements of a string that is sufficiently difficult to guess.

6 Enumerations

6.1 OperationAction

This class is an enumeration. It has a fixed number of instances each of which represent a particular action that can be applied to an object that together specify an operation. For example we would use the action “create” in conjuction with a particular deal classifier in order to construct a deal operation.

6.1.1 Relationships

	Class	Description	Notes
⇑	Enum		

⇑:Inherits

6.1.2 Operations

«Static Method» OperationAction create() create

This method returns the instance of this class that has the name “create”.

«Static Method» OperationAction browse() browse

This method returns the instance of this class that has the name “browse”.

«Static Method» OperationAction modify() modify

This method returns the instance of this class that has the name “modify”.

«Static Method» OperationAction cancel() cancel

This method returns the instance of this class that has the name “cancel”.

«Static Method» OperationAction delete() delete

This method returns the instance of this class that has the name “delete”.

«Static Method» OperationAction open() open

This method returns the instance of this class that has the name “open”.

«Static Method» OperationAction perform() perform

This method returns the instance of this class that has the name “perform”.

«Static Method» Collection<OperationAction> elements() elements

This method returns all the instances of this enumeration class. This will be a list of all the possible actions that can be applied to an object in order to form an operation, however not all of these actions can be applied to every object.

7 Associations

Table 1: Permissions— Associations

Association Role	Class	Card.	Notes
permissionGrant			
	PermissionGrant §3.7		→
	PermissionOwnerModel §4.10		
authenticator			
	Authenticator §3.3		→
	PermissionOwnerModel §4.10		
model			
	PermissionGrant §3.7		→
	PermissionGrantReferenceData §4.9		
collection			
	PermissionGrant §3.7	0..n	→
	CompositePermissionGrant-Model §4.1		◊
accessLogger			
	AccessLogger §3.1		→
	PermissionOwnerModel §4.10		
auditor			
	Auditor		→
	StandardAccessLoggerModel §4.11		
authenticationLogger			
	AuthenticationLogger §3.2		→
	PermissionOwnerModel §4.10		
auditor			
	Auditor		→
	StandardAuthenticationLogger-Model §4.12		
operationObject			
	Classifier		→
	PermissionGrantModel §4.8		
object			
	Classifier		→
	OperationModel §4.5		
type			
	OperationType §3.6		→
	OperationModel §4.5		

Table 1: . . . continued

Association Role	Class	Card.	Notes
operationType			
	OperationType §3.6		→
	PermissionGrantModel §4.8		

→:Navigable ◊:Aggregate ♦:Composite

7.1 permissionGrant

Role: *Navigable* PermissionGrant.
Role: PermissionOwnerModel.

7.2 authenticator

Role: *Navigable* Authenticator.
Role: PermissionOwnerModel.

7.3 model

Role: *Navigable* PermissionGrant.
Role: PermissionGrantReferenceData.

7.4 collection

Role: *Navigable* PermissionGrant, 0..n.
Role: *Aggregate* CompositePermissionGrantModel.

7.5 accessLogger

Role: *Navigable* AccessLogger.
Role: PermissionOwnerModel.

7.6 auditor

Role: *Navigable* Auditor.
Role: StandardAccessLoggerModel.

7.7 authenticationLogger

Role: *Navigable* AuthenticationLogger.
Role: PermissionOwnerModel.

7.8 auditor

Role: *Navigable* Auditor.
Role: StandardAuthenticationLoggerModel.

7.9 operationObject

Role: *Navigable* Classifier.
Role: PermissionGrantModel.

7.10 object

Role: *Navigable* Classifier.
Role: OperationModel.

7.11 type

Role: *Navigable* OperationType.
Role: OperationModel.

7.12 operationType

Role: *Navigable* OperationType.
Role: PermissionGrantModel.

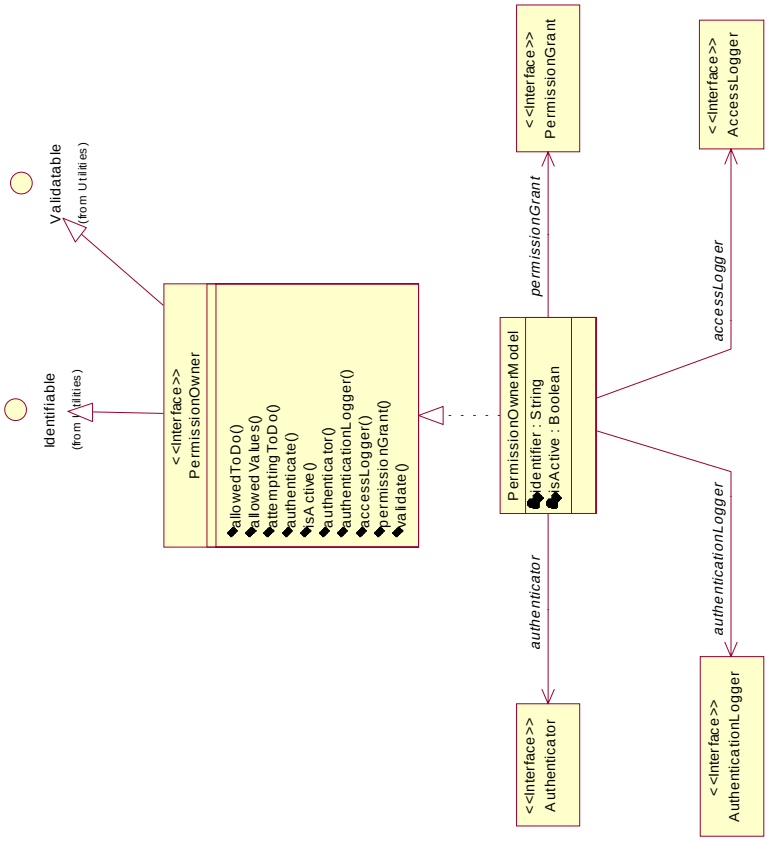


Figure 1: Class Diagram— Permission Owner

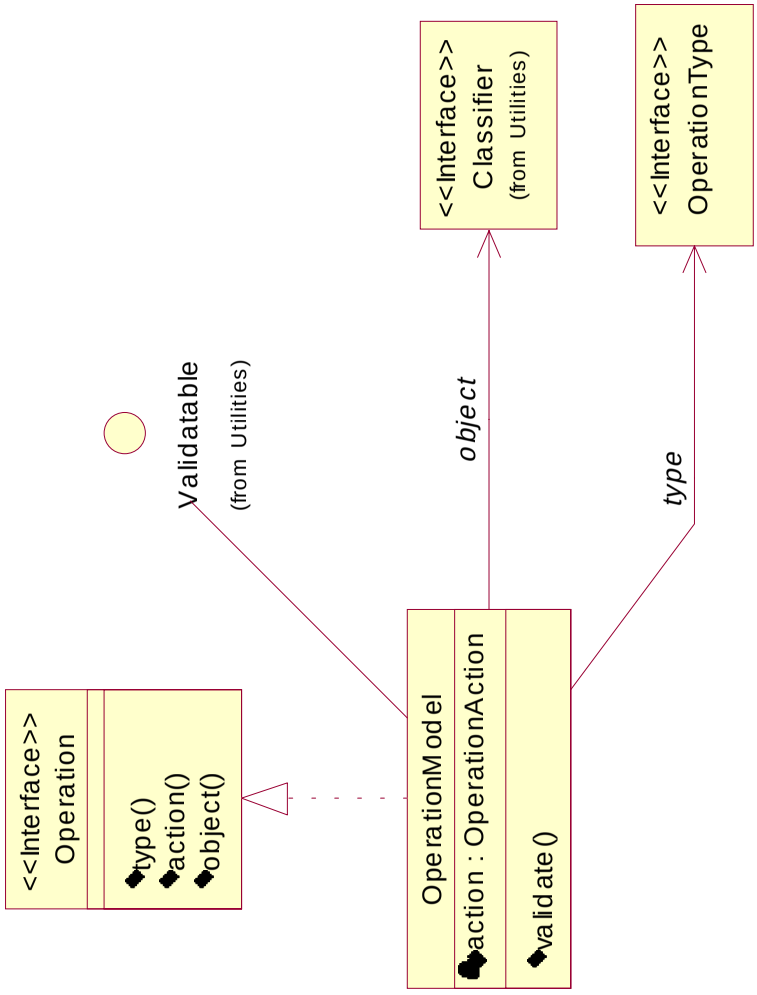


Figure 2: Class Diagram— Operations

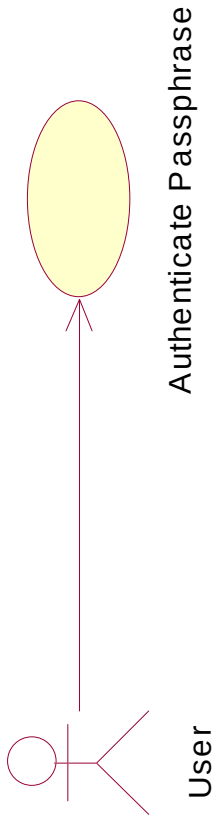


Figure 3: Class Diagram— Authentication

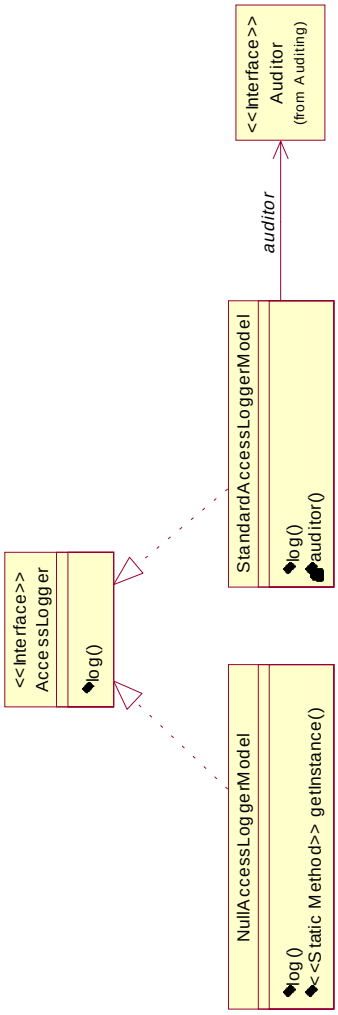


Figure 4: Class Diagram— Access Logging

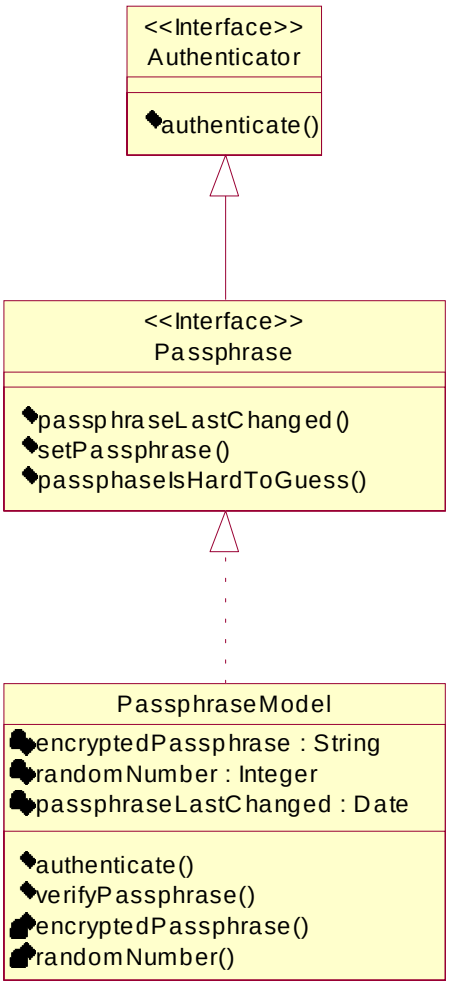


Figure 5: Class Diagram— Authentication

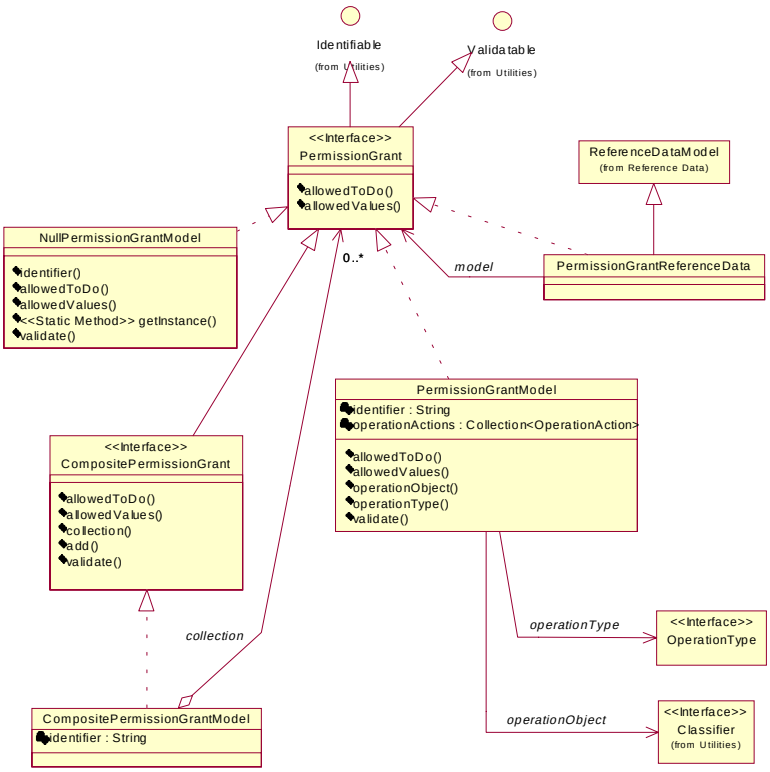


Figure 6: Class Diagram— Permission Grant

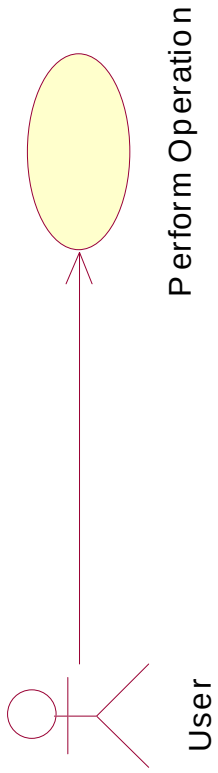


Figure 7: Class Diagram— Attempting an operation

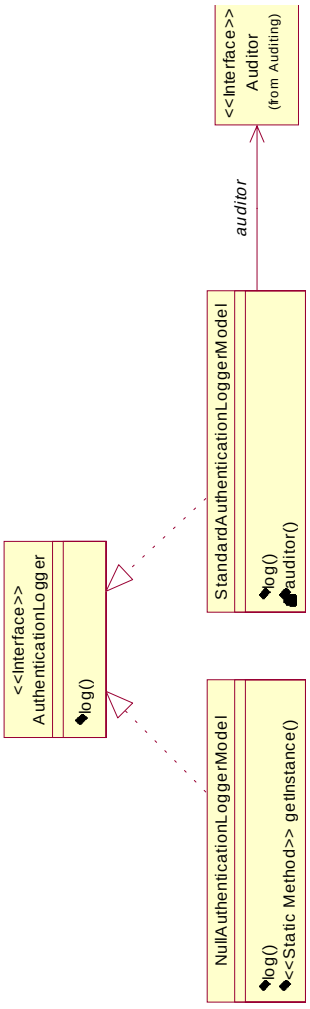


Figure 8: Class Diagram— Authentication Logging

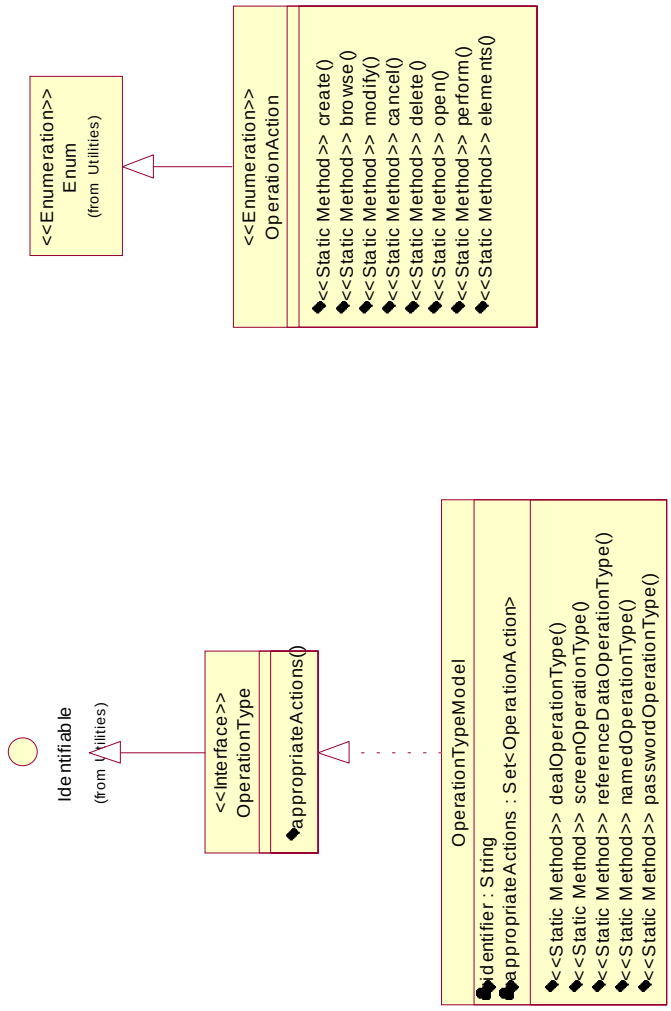


Figure 9: Class Diagram— Operation Type and Action

8 Extensions to the Auditing Package

8.1 Auditor

An Auditor is an object that receives and records arbitrary data. Auditing has not yet been defined.

8.1.1 Relationships

	Class	Description	Notes
↔	StandardAccessLoggerModel §4.11	auditor	
↔	StandardAuthenticationLogger-Model §4.12	auditor	

↔:Association →:Navigable ◊:Aggregate ♦:Composite

References

[1] *The Passphrase FAQ*. version 1.04.
http://www.stack.nl/ galactus/remailers/passphrase-faq.html.

[2] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, second edition, 1996.