| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/RelationServiceMBean.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/management/relation/RelationService.html)   [**NEXT CLASS**](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/management/relation/RelationServiceMBean.html)    [**NO FRAMES**](http://docs.google.com/RelationServiceMBean.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **javax.management.relation**

Interface RelationServiceMBean

**All Known Implementing Classes:** [RelationService](http://docs.google.com/javax/management/relation/RelationService.html)

public interface **RelationServiceMBean**

The Relation Service is in charge of creating and deleting relation types and relations, of handling the consistency and of providing query mechanisms.

**Since:** 1.5

| **Method Summary** | |
| --- | --- |
| void | [**addRelation**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#addRelation(javax.management.ObjectName))([ObjectName](http://docs.google.com/javax/management/ObjectName.html) relationObjectName)            Adds an MBean created by the user (and registered by him in the MBean Server) as a relation in the Relation Service. |
| void | [**addRelationType**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#addRelationType(javax.management.relation.RelationType))([RelationType](http://docs.google.com/javax/management/relation/RelationType.html) relationTypeObj)            Adds given object as a relation type. |
| [Integer](http://docs.google.com/java/lang/Integer.html) | [**checkRoleReading**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#checkRoleReading(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) roleName, [String](http://docs.google.com/java/lang/String.html) relationTypeName)            Checks if given Role can be read in a relation of the given type. |
| [Integer](http://docs.google.com/java/lang/Integer.html) | [**checkRoleWriting**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#checkRoleWriting(javax.management.relation.Role,%20java.lang.String,%20java.lang.Boolean))([Role](http://docs.google.com/javax/management/relation/Role.html) role, [String](http://docs.google.com/java/lang/String.html) relationTypeName, [Boolean](http://docs.google.com/java/lang/Boolean.html) initFlag)            Checks if given Role can be set in a relation of given type. |
| void | [**createRelation**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#createRelation(java.lang.String,%20java.lang.String,%20javax.management.relation.RoleList))([String](http://docs.google.com/java/lang/String.html) relationId, [String](http://docs.google.com/java/lang/String.html) relationTypeName, [RoleList](http://docs.google.com/javax/management/relation/RoleList.html) roleList)            Creates a simple relation (represented by a RelationSupport object) of given relation type, and adds it in the Relation Service. |
| void | [**createRelationType**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#createRelationType(java.lang.String,%20javax.management.relation.RoleInfo%5B%5D))([String](http://docs.google.com/java/lang/String.html) relationTypeName, [RoleInfo](http://docs.google.com/javax/management/relation/RoleInfo.html)[] roleInfoArray)            Creates a relation type (RelationTypeSupport object) with given role infos (provided by the RoleInfo objects), and adds it in the Relation Service. |
| [Map](http://docs.google.com/java/util/Map.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html),[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)>> | [**findAssociatedMBeans**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#findAssociatedMBeans(javax.management.ObjectName,%20java.lang.String,%20java.lang.String))([ObjectName](http://docs.google.com/javax/management/ObjectName.html) mbeanName, [String](http://docs.google.com/java/lang/String.html) relationTypeName, [String](http://docs.google.com/java/lang/String.html) roleName)            Retrieves the MBeans associated to given one in a relation. |
| [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)>> | [**findReferencingRelations**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#findReferencingRelations(javax.management.ObjectName,%20java.lang.String,%20java.lang.String))([ObjectName](http://docs.google.com/javax/management/ObjectName.html) mbeanName, [String](http://docs.google.com/java/lang/String.html) relationTypeName, [String](http://docs.google.com/java/lang/String.html) roleName)            Retrieves the relations where a given MBean is referenced. |
| [List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> | [**findRelationsOfType**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#findRelationsOfType(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationTypeName)            Returns the relation ids for relations of the given type. |
| [List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getAllRelationIds**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getAllRelationIds())()            Returns all the relation ids for all the relations handled by the Relation Service. |
| [List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getAllRelationTypeNames**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getAllRelationTypeNames())()            Retrieves names of all known relation types. |
| [RoleResult](http://docs.google.com/javax/management/relation/RoleResult.html) | [**getAllRoles**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getAllRoles(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            Returns all roles present in the relation. |
| boolean | [**getPurgeFlag**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getPurgeFlag())()            Returns the flag to indicate if when a notification is received for the unregistration of an MBean referenced in a relation, if an immediate "purge" of the relations (look for the relations no longer valid) has to be performed, or if that will be performed only when the purgeRelations method is explicitly called. |
| [Map](http://docs.google.com/java/util/Map.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html),[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)>> | [**getReferencedMBeans**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getReferencedMBeans(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            Retrieves MBeans referenced in the various roles of the relation. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRelationTypeName**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRelationTypeName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            Returns name of associated relation type for given relation. |
| [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> | [**getRole**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRole(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId, [String](http://docs.google.com/java/lang/String.html) roleName)            Retrieves role value for given role name in given relation. |
| [Integer](http://docs.google.com/java/lang/Integer.html) | [**getRoleCardinality**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRoleCardinality(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId, [String](http://docs.google.com/java/lang/String.html) roleName)            Retrieves the number of MBeans currently referenced in the given role. |
| [RoleInfo](http://docs.google.com/javax/management/relation/RoleInfo.html) | [**getRoleInfo**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRoleInfo(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationTypeName, [String](http://docs.google.com/java/lang/String.html) roleInfoName)            Retrieves role info for given role of a given relation type. |
| [List](http://docs.google.com/java/util/List.html)<[RoleInfo](http://docs.google.com/javax/management/relation/RoleInfo.html)> | [**getRoleInfos**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRoleInfos(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationTypeName)            Retrieves list of role infos (RoleInfo objects) of a given relation type. |
| [RoleResult](http://docs.google.com/javax/management/relation/RoleResult.html) | [**getRoles**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRoles(java.lang.String,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) relationId, [String](http://docs.google.com/java/lang/String.html)[] roleNameArray)            Retrieves values of roles with given names in given relation. |
| [Boolean](http://docs.google.com/java/lang/Boolean.html) | [**hasRelation**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#hasRelation(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            Checks if there is a relation identified in Relation Service with given relation id. |
| void | [**isActive**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#isActive())()            Checks if the Relation Service is active. |
| [String](http://docs.google.com/java/lang/String.html) | [**isRelation**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#isRelation(javax.management.ObjectName))([ObjectName](http://docs.google.com/javax/management/ObjectName.html) objectName)            Returns the relation id associated to the given ObjectName if the MBean has been added as a relation in the Relation Service. |
| [ObjectName](http://docs.google.com/javax/management/ObjectName.html) | [**isRelationMBean**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#isRelationMBean(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            If the relation is represented by an MBean (created by the user and added as a relation in the Relation Service), returns the ObjectName of the MBean. |
| void | [**purgeRelations**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#purgeRelations())()            Purges the relations. |
| void | [**removeRelation**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#removeRelation(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            Removes given relation from the Relation Service. |
| void | [**removeRelationType**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#removeRelationType(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationTypeName)            Removes given relation type from Relation Service. |
| void | [**sendRelationCreationNotification**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#sendRelationCreationNotification(java.lang.String))([String](http://docs.google.com/java/lang/String.html) relationId)            Sends a notification (RelationNotification) for a relation creation. |
| void | [**sendRelationRemovalNotification**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#sendRelationRemovalNotification(java.lang.String,%20java.util.List))([String](http://docs.google.com/java/lang/String.html) relationId, [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> unregMBeanList)            Sends a notification (RelationNotification) for a relation removal. |
| void | [**sendRoleUpdateNotification**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#sendRoleUpdateNotification(java.lang.String,%20javax.management.relation.Role,%20java.util.List))([String](http://docs.google.com/java/lang/String.html) relationId, [Role](http://docs.google.com/javax/management/relation/Role.html) newRole, [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> oldRoleValue)            Sends a notification (RelationNotification) for a role update in the given relation. |
| void | [**setPurgeFlag**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#setPurgeFlag(boolean))(boolean purgeFlag)            Sets the flag to indicate if when a notification is received for the unregistration of an MBean referenced in a relation, if an immediate "purge" of the relations (look for the relations no longer valid) has to be performed, or if that will be performed only when the purgeRelations method is explicitly called. |
| void | [**setRole**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#setRole(java.lang.String,%20javax.management.relation.Role))([String](http://docs.google.com/java/lang/String.html) relationId, [Role](http://docs.google.com/javax/management/relation/Role.html) role)            Sets the given role in given relation. |
| [RoleResult](http://docs.google.com/javax/management/relation/RoleResult.html) | [**setRoles**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#setRoles(java.lang.String,%20javax.management.relation.RoleList))([String](http://docs.google.com/java/lang/String.html) relationId, [RoleList](http://docs.google.com/javax/management/relation/RoleList.html) roleList)            Sets the given roles in given relation. |
| void | [**updateRoleMap**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#updateRoleMap(java.lang.String,%20javax.management.relation.Role,%20java.util.List))([String](http://docs.google.com/java/lang/String.html) relationId, [Role](http://docs.google.com/javax/management/relation/Role.html) newRole, [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> oldRoleValue)            Handles update of the Relation Service role map for the update of given role in given relation. |

| **Method Detail** |
| --- |

### isActive

void **isActive**()  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html)

Checks if the Relation Service is active. Current condition is that the Relation Service must be registered in the MBean Server

**Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if it is not registered

### getPurgeFlag

boolean **getPurgeFlag**()

Returns the flag to indicate if when a notification is received for the unregistration of an MBean referenced in a relation, if an immediate "purge" of the relations (look for the relations no longer valid) has to be performed, or if that will be performed only when the purgeRelations method is explicitly called.

true is immediate purge.

**Returns:**true if purges are immediate.**See Also:**[setPurgeFlag(boolean)](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#setPurgeFlag(boolean))

### setPurgeFlag

void **setPurgeFlag**(boolean purgeFlag)

Sets the flag to indicate if when a notification is received for the unregistration of an MBean referenced in a relation, if an immediate "purge" of the relations (look for the relations no longer valid) has to be performed, or if that will be performed only when the purgeRelations method is explicitly called.

true is immediate purge.

**Parameters:**purgeFlag - flag**See Also:**[getPurgeFlag()](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getPurgeFlag())

### createRelationType

void **createRelationType**([String](http://docs.google.com/java/lang/String.html) relationTypeName,  
 [RoleInfo](http://docs.google.com/javax/management/relation/RoleInfo.html)[] roleInfoArray)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [InvalidRelationTypeException](http://docs.google.com/javax/management/relation/InvalidRelationTypeException.html)

Creates a relation type (RelationTypeSupport object) with given role infos (provided by the RoleInfo objects), and adds it in the Relation Service.

**Parameters:**relationTypeName - name of the relation typeroleInfoArray - array of role infos **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [InvalidRelationTypeException](http://docs.google.com/javax/management/relation/InvalidRelationTypeException.html) - If:

- there is already a relation type with that name

- the same name has been used for two different role infos

- no role info provided

- one null role info provided

### addRelationType

void **addRelationType**([RelationType](http://docs.google.com/javax/management/relation/RelationType.html) relationTypeObj)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [InvalidRelationTypeException](http://docs.google.com/javax/management/relation/InvalidRelationTypeException.html)

Adds given object as a relation type. The object is expected to implement the RelationType interface.

**Parameters:**relationTypeObj - relation type object (implementing the RelationType interface) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter or if [relationTypeObj.getRelationTypeName()](http://docs.google.com/javax/management/relation/RelationType.html#getRelationTypeName()) returns null. [InvalidRelationTypeException](http://docs.google.com/javax/management/relation/InvalidRelationTypeException.html) - if there is already a relation type with that name

### getAllRelationTypeNames

[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> **getAllRelationTypeNames**()

Retrieves names of all known relation types.

**Returns:**ArrayList of relation type names (Strings)

### getRoleInfos

[List](http://docs.google.com/java/util/List.html)<[RoleInfo](http://docs.google.com/javax/management/relation/RoleInfo.html)> **getRoleInfos**([String](http://docs.google.com/java/lang/String.html) relationTypeName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html)

Retrieves list of role infos (RoleInfo objects) of a given relation type.

**Parameters:**relationTypeName - name of relation type **Returns:**ArrayList of RoleInfo. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if there is no relation type with that name.

### getRoleInfo

[RoleInfo](http://docs.google.com/javax/management/relation/RoleInfo.html) **getRoleInfo**([String](http://docs.google.com/java/lang/String.html) relationTypeName,  
 [String](http://docs.google.com/java/lang/String.html) roleInfoName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html),  
 [RoleInfoNotFoundException](http://docs.google.com/javax/management/relation/RoleInfoNotFoundException.html)

Retrieves role info for given role of a given relation type.

**Parameters:**relationTypeName - name of relation typeroleInfoName - name of role **Returns:**RoleInfo object. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if the relation type is not known in the Relation Service [RoleInfoNotFoundException](http://docs.google.com/javax/management/relation/RoleInfoNotFoundException.html) - if the role is not part of the relation type.

### removeRelationType

void **removeRelationType**([String](http://docs.google.com/java/lang/String.html) relationTypeName)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html)

Removes given relation type from Relation Service.

The relation objects of that type will be removed from the Relation Service.

**Parameters:**relationTypeName - name of the relation type to be removed **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - If there is no relation type with that name

### createRelation

void **createRelation**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [String](http://docs.google.com/java/lang/String.html) relationTypeName,  
 [RoleList](http://docs.google.com/javax/management/relation/RoleList.html) roleList)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html),  
 [InvalidRelationIdException](http://docs.google.com/javax/management/relation/InvalidRelationIdException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html),  
 [InvalidRoleValueException](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html)

Creates a simple relation (represented by a RelationSupport object) of given relation type, and adds it in the Relation Service.

Roles are initialized according to the role list provided in parameter. The ones not initialized in this way are set to an empty ArrayList of ObjectNames.

A RelationNotification, with type RELATION\_BASIC\_CREATION, is sent.

**Parameters:**relationId - relation identifier, to identify uniquely the relation inside the Relation ServicerelationTypeName - name of the relation type (has to be created in the Relation Service)roleList - role list to initialize roles of the relation (can be null). **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html) - if a value is provided for a role that does not exist in the relation type [InvalidRelationIdException](http://docs.google.com/javax/management/relation/InvalidRelationIdException.html) - if relation id already used [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if relation type not known in Relation Service [InvalidRoleValueException](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html) - if:

- the same role name is used for two different roles

- the number of referenced MBeans in given value is less than expected minimum degree

- the number of referenced MBeans in provided value exceeds expected maximum degree

- one referenced MBean in the value is not an Object of the MBean class expected for that role

- an MBean provided for that role does not exist

### addRelation

void **addRelation**([ObjectName](http://docs.google.com/javax/management/ObjectName.html) relationObjectName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html),  
 [InvalidRelationIdException](http://docs.google.com/javax/management/relation/InvalidRelationIdException.html),  
 [InstanceNotFoundException](http://docs.google.com/javax/management/InstanceNotFoundException.html),  
 [InvalidRelationServiceException](http://docs.google.com/javax/management/relation/InvalidRelationServiceException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html),  
 [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html),  
 [InvalidRoleValueException](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html)

Adds an MBean created by the user (and registered by him in the MBean Server) as a relation in the Relation Service.

To be added as a relation, the MBean must conform to the following:

- implement the Relation interface

- have for RelationService ObjectName the ObjectName of current Relation Service

- have a relation id that is unique and unused in current Relation Service

- have for relation type a relation type created in the Relation Service

- have roles conforming to the role info provided in the relation type.

**Parameters:**relationObjectName - ObjectName of the relation MBean to be added. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html) - If the MBean does not implement the Relation interface [InvalidRelationIdException](http://docs.google.com/javax/management/relation/InvalidRelationIdException.html) - if:

- no relation identifier in MBean

- the relation identifier is already used in the Relation Service

[InstanceNotFoundException](http://docs.google.com/javax/management/InstanceNotFoundException.html) - if the MBean for given ObjectName has not been registered [InvalidRelationServiceException](http://docs.google.com/javax/management/relation/InvalidRelationServiceException.html) - if:

- no Relation Service name in MBean

- the Relation Service name in the MBean is not the one of the current Relation Service

[RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if:

- no relation type name in MBean

- the relation type name in MBean does not correspond to a relation type created in the Relation Service

[InvalidRoleValueException](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html) - if:

- the number of referenced MBeans in a role is less than expected minimum degree

- the number of referenced MBeans in a role exceeds expected maximum degree

- one referenced MBean in the value is not an Object of the MBean class expected for that role

- an MBean provided for a role does not exist

[RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html) - if a value is provided for a role that does not exist in the relation type

### isRelationMBean

[ObjectName](http://docs.google.com/javax/management/ObjectName.html) **isRelationMBean**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

If the relation is represented by an MBean (created by the user and added as a relation in the Relation Service), returns the ObjectName of the MBean.

**Parameters:**relationId - relation id identifying the relation **Returns:**ObjectName of the corresponding relation MBean, or null if the relation is not an MBean. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - there is no relation associated to that id

### isRelation

[String](http://docs.google.com/java/lang/String.html) **isRelation**([ObjectName](http://docs.google.com/javax/management/ObjectName.html) objectName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html)

Returns the relation id associated to the given ObjectName if the MBean has been added as a relation in the Relation Service.

**Parameters:**objectName - ObjectName of supposed relation **Returns:**relation id (String) or null (if the ObjectName is not a relation handled by the Relation Service) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter

### hasRelation

[Boolean](http://docs.google.com/java/lang/Boolean.html) **hasRelation**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html)

Checks if there is a relation identified in Relation Service with given relation id.

**Parameters:**relationId - relation id identifying the relation **Returns:**boolean: true if there is a relation, false else **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter

### getAllRelationIds

[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> **getAllRelationIds**()

Returns all the relation ids for all the relations handled by the Relation Service.

**Returns:**ArrayList of String

### checkRoleReading

[Integer](http://docs.google.com/java/lang/Integer.html) **checkRoleReading**([String](http://docs.google.com/java/lang/String.html) roleName,  
 [String](http://docs.google.com/java/lang/String.html) relationTypeName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html)

Checks if given Role can be read in a relation of the given type.

**Parameters:**roleName - name of role to be checkedrelationTypeName - name of the relation type **Returns:**an Integer wrapping an integer corresponding to possible problems represented as constants in RoleUnresolved:

- 0 if role can be read

- integer corresponding to RoleStatus.NO\_ROLE\_WITH\_NAME

- integer corresponding to RoleStatus.ROLE\_NOT\_READABLE

**Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if the relation type is not known in the Relation Service

### checkRoleWriting

[Integer](http://docs.google.com/java/lang/Integer.html) **checkRoleWriting**([Role](http://docs.google.com/javax/management/relation/Role.html) role,  
 [String](http://docs.google.com/java/lang/String.html) relationTypeName,  
 [Boolean](http://docs.google.com/java/lang/Boolean.html) initFlag)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html)

Checks if given Role can be set in a relation of given type.

**Parameters:**role - role to be checkedrelationTypeName - name of relation typeinitFlag - flag to specify that the checking is done for the initialization of a role, write access shall not be verified. **Returns:**an Integer wrapping an integer corresponding to possible problems represented as constants in RoleUnresolved:

- 0 if role can be set

- integer corresponding to RoleStatus.NO\_ROLE\_WITH\_NAME

- integer for RoleStatus.ROLE\_NOT\_WRITABLE

- integer for RoleStatus.LESS\_THAN\_MIN\_ROLE\_DEGREE

- integer for RoleStatus.MORE\_THAN\_MAX\_ROLE\_DEGREE

- integer for RoleStatus.REF\_MBEAN\_OF\_INCORRECT\_CLASS

- integer for RoleStatus.REF\_MBEAN\_NOT\_REGISTERED

**Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if unknown relation type

### sendRelationCreationNotification

void **sendRelationCreationNotification**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Sends a notification (RelationNotification) for a relation creation. The notification type is:

- RelationNotification.RELATION\_BASIC\_CREATION if the relation is an object internal to the Relation Service

- RelationNotification.RELATION\_MBEAN\_CREATION if the relation is a MBean added as a relation.

The source object is the Relation Service itself.

It is called in Relation Service createRelation() and addRelation() methods.

**Parameters:**relationId - relation identifier of the updated relation **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if there is no relation for given relation id

### sendRoleUpdateNotification

void **sendRoleUpdateNotification**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [Role](http://docs.google.com/javax/management/relation/Role.html) newRole,  
 [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> oldRoleValue)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Sends a notification (RelationNotification) for a role update in the given relation. The notification type is:

- RelationNotification.RELATION\_BASIC\_UPDATE if the relation is an object internal to the Relation Service

- RelationNotification.RELATION\_MBEAN\_UPDATE if the relation is a MBean added as a relation.

The source object is the Relation Service itself.

It is called in relation MBean setRole() (for given role) and setRoles() (for each role) methods (implementation provided in RelationSupport class).

It is also called in Relation Service setRole() (for given role) and setRoles() (for each role) methods.

**Parameters:**relationId - relation identifier of the updated relationnewRole - new role (name and new value)oldRoleValue - old role value (List of ObjectName objects) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if there is no relation for given relation id

### sendRelationRemovalNotification

void **sendRelationRemovalNotification**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> unregMBeanList)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Sends a notification (RelationNotification) for a relation removal. The notification type is:

- RelationNotification.RELATION\_BASIC\_REMOVAL if the relation is an object internal to the Relation Service

- RelationNotification.RELATION\_MBEAN\_REMOVAL if the relation is a MBean added as a relation.

The source object is the Relation Service itself.

It is called in Relation Service removeRelation() method.

**Parameters:**relationId - relation identifier of the updated relationunregMBeanList - List of ObjectNames of MBeans expected to be unregistered due to relation removal (can be null) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if there is no relation for given relation id

### updateRoleMap

void **updateRoleMap**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [Role](http://docs.google.com/javax/management/relation/Role.html) newRole,  
 [List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> oldRoleValue)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Handles update of the Relation Service role map for the update of given role in given relation.

It is called in relation MBean setRole() (for given role) and setRoles() (for each role) methods (implementation provided in RelationSupport class).

It is also called in Relation Service setRole() (for given role) and setRoles() (for each role) methods.

To allow the Relation Service to maintain the consistency (in case of MBean unregistration) and to be able to perform queries, this method must be called when a role is updated.

**Parameters:**relationId - relation identifier of the updated relationnewRole - new role (name and new value)oldRoleValue - old role value (List of ObjectName objects) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation for given id.

### removeRelation

void **removeRelation**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Removes given relation from the Relation Service.

A RelationNotification notification is sent, its type being:

- RelationNotification.RELATION\_BASIC\_REMOVAL if the relation was only internal to the Relation Service

- RelationNotification.RELATION\_MBEAN\_REMOVAL if the relation is registered as an MBean.

For MBeans referenced in such relation, nothing will be done,

**Parameters:**relationId - relation id of the relation to be removed **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation corresponding to given relation id

### purgeRelations

void **purgeRelations**()  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html)

Purges the relations.

Depending on the purgeFlag value, this method is either called automatically when a notification is received for the unregistration of an MBean referenced in a relation (if the flag is set to true), or not (if the flag is set to false).

In that case it is up to the user to call it to maintain the consistency of the relations. To be kept in mind that if an MBean is unregistered and the purge not done immediately, if the ObjectName is reused and assigned to another MBean referenced in a relation, calling manually this purgeRelations() method will cause trouble, as will consider the ObjectName as corresponding to the unregistered MBean, not seeing the new one.

The behavior depends on the cardinality of the role where the unregistered MBean is referenced:

- if removing one MBean reference in the role makes its number of references less than the minimum degree, the relation has to be removed.

- if the remaining number of references after removing the MBean reference is still in the cardinality range, keep the relation and update it calling its handleMBeanUnregistration() callback.

**Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server.

### findReferencingRelations

[Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)>> **findReferencingRelations**([ObjectName](http://docs.google.com/javax/management/ObjectName.html) mbeanName,  
 [String](http://docs.google.com/java/lang/String.html) relationTypeName,  
 [String](http://docs.google.com/java/lang/String.html) roleName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html)

Retrieves the relations where a given MBean is referenced.

This corresponds to the CIM "References" and "ReferenceNames" operations.

**Parameters:**mbeanName - ObjectName of MBeanrelationTypeName - can be null; if specified, only the relations of that type will be considered in the search. Else all relation types are considered.roleName - can be null; if specified, only the relations where the MBean is referenced in that role will be returned. Else all roles are considered. **Returns:**an HashMap, where the keys are the relation ids of the relations where the MBean is referenced, and the value is, for each key, an ArrayList of role names (as an MBean can be referenced in several roles in the same relation). **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter

### findAssociatedMBeans

[Map](http://docs.google.com/java/util/Map.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html),[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)>> **findAssociatedMBeans**([ObjectName](http://docs.google.com/javax/management/ObjectName.html) mbeanName,  
 [String](http://docs.google.com/java/lang/String.html) relationTypeName,  
 [String](http://docs.google.com/java/lang/String.html) roleName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html)

Retrieves the MBeans associated to given one in a relation.

This corresponds to CIM Associators and AssociatorNames operations.

**Parameters:**mbeanName - ObjectName of MBeanrelationTypeName - can be null; if specified, only the relations of that type will be considered in the search. Else all relation types are considered.roleName - can be null; if specified, only the relations where the MBean is referenced in that role will be considered. Else all roles are considered. **Returns:**an HashMap, where the keys are the ObjectNames of the MBeans associated to given MBean, and the value is, for each key, an ArrayList of the relation ids of the relations where the key MBean is associated to given one (as they can be associated in several different relations). **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter

### findRelationsOfType

[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> **findRelationsOfType**([String](http://docs.google.com/java/lang/String.html) relationTypeName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html)

Returns the relation ids for relations of the given type.

**Parameters:**relationTypeName - relation type name **Returns:**an ArrayList of relation ids. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if there is no relation type with that name.

### getRole

[List](http://docs.google.com/java/util/List.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html)> **getRole**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [String](http://docs.google.com/java/lang/String.html) roleName)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html),  
 [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html)

Retrieves role value for given role name in given relation.

**Parameters:**relationId - relation idroleName - name of role **Returns:**the ArrayList of ObjectName objects being the role value **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation with given id [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html) - if:

- there is no role with given name

or

- the role is not readable.

**See Also:**[setRole(java.lang.String, javax.management.relation.Role)](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#setRole(java.lang.String,%20javax.management.relation.Role))

### getRoles

[RoleResult](http://docs.google.com/javax/management/relation/RoleResult.html) **getRoles**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [String](http://docs.google.com/java/lang/String.html)[] roleNameArray)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Retrieves values of roles with given names in given relation.

**Parameters:**relationId - relation idroleNameArray - array of names of roles to be retrieved **Returns:**a RoleResult object, including a RoleList (for roles successfully retrieved) and a RoleUnresolvedList (for roles not retrieved). **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation with given id**See Also:**[setRoles(java.lang.String, javax.management.relation.RoleList)](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#setRoles(java.lang.String,%20javax.management.relation.RoleList))

### getAllRoles

[RoleResult](http://docs.google.com/javax/management/relation/RoleResult.html) **getAllRoles**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html),  
 [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html)

Returns all roles present in the relation.

**Parameters:**relationId - relation id **Returns:**a RoleResult object, including a RoleList (for roles successfully retrieved) and a RoleUnresolvedList (for roles not readable). **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation for given id [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server

### getRoleCardinality

[Integer](http://docs.google.com/java/lang/Integer.html) **getRoleCardinality**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [String](http://docs.google.com/java/lang/String.html) roleName)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html),  
 [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html)

Retrieves the number of MBeans currently referenced in the given role.

**Parameters:**relationId - relation idroleName - name of role **Returns:**the number of currently referenced MBeans in that role **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation with given id [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html) - if there is no role with given name

### setRole

void **setRole**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [Role](http://docs.google.com/javax/management/relation/Role.html) role)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html),  
 [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html),  
 [InvalidRoleValueException](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html),  
 [RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html)

Sets the given role in given relation.

Will check the role according to its corresponding role definition provided in relation's relation type

The Relation Service will keep track of the change to keep the consistency of relations by handling referenced MBean unregistrations.

**Parameters:**relationId - relation idrole - role to be set (name and new value) **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation with given id [RoleNotFoundException](http://docs.google.com/javax/management/relation/RoleNotFoundException.html) - if:

- internal relation

and

- the role does not exist or is not writable

[InvalidRoleValueException](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html) - if internal relation and value provided for role is not valid:

- the number of referenced MBeans in given value is less than expected minimum degree

or

- the number of referenced MBeans in provided value exceeds expected maximum degree

or

- one referenced MBean in the value is not an Object of the MBean class expected for that role

or

- an MBean provided for that role does not exist

[RelationTypeNotFoundException](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) - if unknown relation type**See Also:**[getRole(java.lang.String, java.lang.String)](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRole(java.lang.String,%20java.lang.String))

### setRoles

[RoleResult](http://docs.google.com/javax/management/relation/RoleResult.html) **setRoles**([String](http://docs.google.com/java/lang/String.html) relationId,  
 [RoleList](http://docs.google.com/javax/management/relation/RoleList.html) roleList)  
 throws [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html),  
 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Sets the given roles in given relation.

Will check the role according to its corresponding role definition provided in relation's relation type

The Relation Service keeps track of the changes to keep the consistency of relations by handling referenced MBean unregistrations.

**Parameters:**relationId - relation idroleList - list of roles to be set **Returns:**a RoleResult object, including a RoleList (for roles successfully set) and a RoleUnresolvedList (for roles not set). **Throws:** [RelationServiceNotRegisteredException](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) - if the Relation Service is not registered in the MBean Server [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation with given id**See Also:**[getRoles(java.lang.String, java.lang.String[])](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#getRoles(java.lang.String,%20java.lang.String%5B%5D))

### getReferencedMBeans

[Map](http://docs.google.com/java/util/Map.html)<[ObjectName](http://docs.google.com/javax/management/ObjectName.html),[List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)>> **getReferencedMBeans**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Retrieves MBeans referenced in the various roles of the relation.

**Parameters:**relationId - relation id **Returns:**a HashMap mapping:

ObjectName -> ArrayList of String (role names)

**Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation for given relation id

### getRelationTypeName

[String](http://docs.google.com/java/lang/String.html) **getRelationTypeName**([String](http://docs.google.com/java/lang/String.html) relationId)  
 throws [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html),  
 [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html)

Returns name of associated relation type for given relation.

**Parameters:**relationId - relation id **Returns:**the name of the associated relation type. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if null parameter [RelationNotFoundException](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) - if no relation for given relation id

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/RelationServiceMBean.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/management/relation/RelationService.html)   [**NEXT CLASS**](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/management/relation/RelationServiceMBean.html)    [**NO FRAMES**](http://docs.google.com/RelationServiceMBean.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).