IdentityIQ Object Model and Usage

IdentityIQ Versions: 7.x through 8.x

This document describes the Objects in IdentityIQ, their attributes, and the objects' relationships to each other. Where appropriate, objects are groups by functionality within the IdentityIQ products, identifying which objects are used in each functional area. Refer to the JavaDocs that ship with the product for the methods available to access the objects' attributes. This document version pertains specifically to versions 7.0 through 8.3.

Document Revision History

Revision Date	Written/Edited By	Comments
Nov 2016	Jennifer Mitchell	Updated for 7.0; modified QuickLink object set – major changes in 7.0
Jan 2018	Jennifer Mitchell	Updated for 7.1 and 7.2; new Alert, AlertDefinition, Plugin, ProvisioningTransaction; further modifications to several other object sets, and addition of some new relationship diagram which were previously omitted
Aug 2018	Jennifer Mitchell	Updated for 7.3; new Environment Monitoring section and alterations to Request object set for RequestState addition
June 2022	Cathy Mallet	Updated for 8.3 and prior releases

Table of Contents

Introduction

The core objects in the IdentityIQ Object model are identified and described in this document, including their relationships to other objects in the model. All key attributes necessary to understanding object usage and relationships are listed and described for each object; some private attributes that don't contribute to interconnections between objects and that are not primary functional elements of the object may be omitted, particularly when they are low-usage fields that do not help clarify the model. Most of the listed (and non-listed) attributes are private attributes that can only be retrieved or modified through the classes' available public methods, which are listed in the JavaDocs that accompany the IdentityIQ product.

There is a high level of interconnection between objects in the IdentityIQ Object Model. To examine them all at once would prove overwhelming and minimally informative, at best. Instead, this document approaches the model by focusing on core objects and key areas of application functionality.

NOTE: Most object relationships shown here are direct references in one object to another object. Occasionally, the relationships are only managed through an ID or name value stored in the primary object that refers to the secondary object. In those cases, the relationships between the objects are shown with dashed lines to indicate that looser connection.

Objects' Shared Attributes

Many IdentityIQ database objects contain a shared set of attributes that provide basic identifying information and some core functionality. For clarity, these are listed only once here instead of repeating them on each object. These attributes are:

Attribute*	Description
id	Database ID; set when saved
name	user-defined name; though this is unique for most objects, that is not necessarily the case for all objects
lock	Contains lock information for record
owner	Owning Identity object
description	Verbose comments
created	Date object was created in persistent store
modified	Date object was last modified in persistent store
assignedScope	Scope object to which object is assigned (if scopes are in use); can be null
pendingWorkflow	Pending WorkflowCase object that will modify the object; only one can exist per object at a time
disabled	Flag marking object as disabled; does not apply to all objects; field only populated through methods on individual objects where this is relevant
extendedAttributeColumns	Array of the extended attributes (for objects that have extended attributes); stores the object's values for the extended attributes defined for the object type
attributeMetaData	List of attributeMetaData objects On Identity objects, attributeMetaData exists for any extended attribute copied from a link attribute, indicating the source application for the attribute

	value. If the attribute is marked as editable the username and lastValue fields will be set if the attribute value has been changed (through the UI) since it was copied from the link. On Link objects, attributeMetaData is only recorded if the attribute value has been edited through the UI. No other objects use this field.

* Any object which inherits from the sailpoint.object.SailPointObject class contains these common attributes. When these attributes are not applicable to a specific object type, a note to that effect will be shown below the attribute table for that object.

Identity

The Identity Object is the central object in IdentityIQ. Most Identities represent individual people within the company. The user's login credentials and access permissions for IdentityIQ itself are also stored in the Identity Object.

Some Identities represent workgroups – sets of Identities that can be assigned as a unit to activities within IdentityIQ. This is a special case of Identity whose attribute usage differs from other Identities; many fields that are normally populated for individual Identities are null for workgroup Identities.

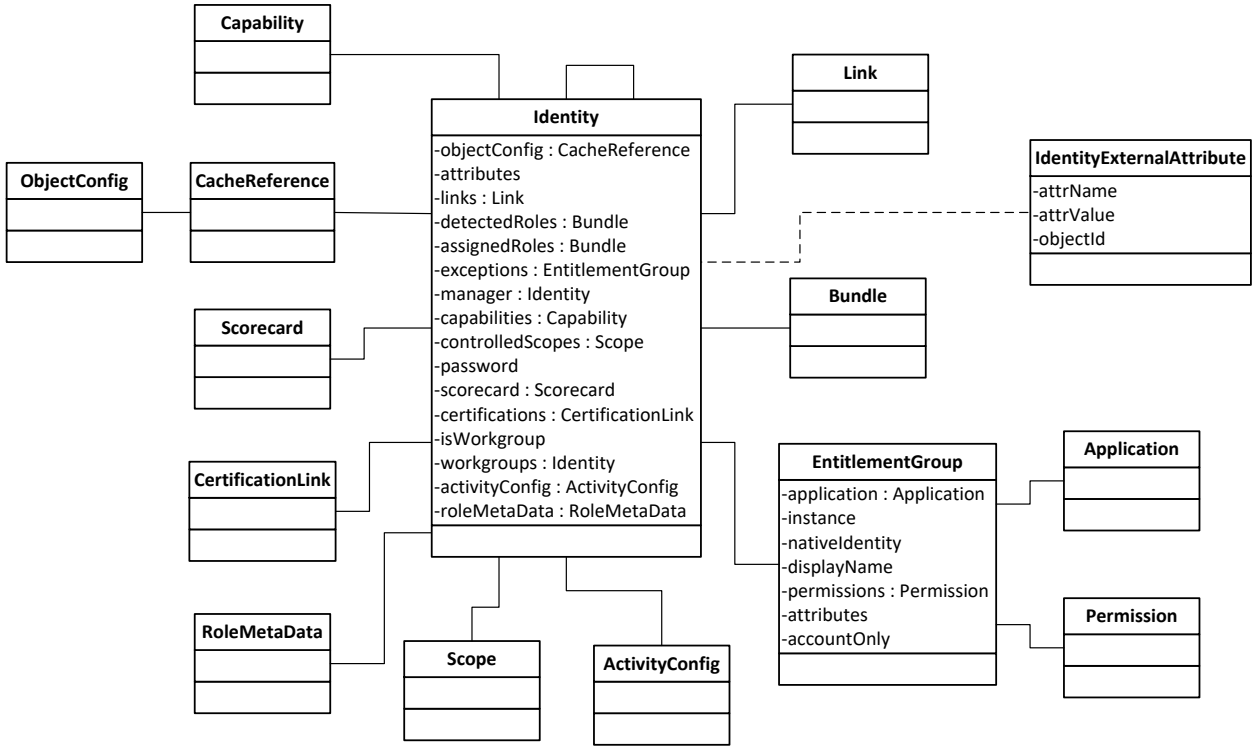


Figure 1: Identity Object and Its Attribute Relationships

The **Identity** object contains these attributes.

Attribute	Description
objectConfig	Searchable and extended attributes for the Identity (cached for frequent use by persistence layer, so accessed through cacheReference)
attributes	Extended attribute map (name/value pairs)

links	List of Link objects (represent accounts on an application correlated to Identity)
detectedRoles	List of Bundle objects representing the Identity's detected (IT) roles
assignedRoles	List of Bundle objects representing the Identity's assigned (business) roles
exceptions	List of EntitlementGroup objects that represent the Identity's system access outside of Roles (in UI, these are called Additional Entitlements)
manager	Identity object that is this Identity's manager
capabilities	List of Capability objects selected for the Identity (represent sets of IdentityIQ system functionality)
controlledScopes	List of Scope objects to which Identity is assigned
password	IdentityIQ password (passwords are not stored here when pass-through authentication is in use)
scorecard	Scorecard object for Identity (represents Identity risk score)
certifications	List of CertificationLink objects that represent past certifications in which Identity has been included
isWorkgroup	Boolean value that identifies the Identity as a Workgroup
workgroups	List of Workgroups (Identity objects) to which Identity is assigned
activityConfig	Activity monitoring configuration object
roleMetaData	List of RoleMetaData objects for the Identity, which contain information about the roles associated with the Identity

NOTE: Identity Objects which have the isWorkgroup flag set to true appear in the IdentityIQ Debug pages under the object type **Workgroup**, even though there is not actually a separate Workgroup object type. These are only managed separately for administrative ease.

An Entitlement Group is a set of "additional" Entitlements held by the Identity on a single application; Entitlements granted through role assignments (or encapsulated in a detected role) are not represented through this object. The table below describes key attributes on the **EntitlementGroup** object and their usages.

Attribute	Description
application	Application object on which the entitlements exist
instance	Instance identifier for the applications (used when a single application object represents multiple actual items (e.g. UNIX Workstations or an application that exists on multiple platforms) to uniquely identify each instance of the application)
nativeIdentity	User identity on account
displayName	User's display name on account
permissions	List of Permissions objects (indicate target and rights combinations)
attributes	Map of attributes (name-value pairs)
accountOnly	Flag indicating the user only has an account on the application, not specific entitlements

This table describes key attributes on the **IdentityExternalAttribute** object and their usages. This object models the multi-valued extended attributes for an Identity. It is connected to the Identity only through the objectId field, which contains the associated Identity's unique identifier in the IdentityIQ database. NOTE: This objectId is not the same as its name or unique identifier visible in the UI.

Attribute*	Description
attrName	Name of the multi-valued extended attribute
attrValue	Attribute value
objectId	ID value for Identity to which the attribute belongs

***** IdentityExternalAttribute does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The table below describes key attributes on the **RoleMetaData** object and their usages. This object is used to store information about a role that has been detected for or assigned to an identity.

Attribute	Description
role	Reference to the Bundle object (role) to which this meta data relates
additionalEntitlements	Only for assigned roles; Flag indicating that the user has one or more entitlements which are not part of a permitted or required role definition
missingRequired	Only for assigned roles; Flag indicating that the user is missing some of the required roles under this assigned role because they are missing some entitlements which would allow the required role to be detected
assigned	Flag indicating whether this role was assigned to the user
detected	Flag indicating whether this role was detected for the user based on the entitlement profile in the Bundle definition
detectedException	Only applies to detected roles; Flag indicating whether this role is a detected role which is not part of any required or permitted roles list in an assigned role

Role and Attribute Assignment

This set of objects represents roles and entitlements assigned to (and revoked from) an Identity. The RoleAssignment object existed before version 6.0, but these objects were refactored beginning in version 6.0 to implement attribute assignment and to allow both RoleAssignment and AttributeAssignment to inherit from a common Assignment class for their shared attributes. The RoleAssignment and AttributeAssignment classes contain attributes that pertain only to role and entitlement assignments respective.

Entitlement and Role assignments are typically made through the LCM interface. Additionally, role assignment can be done through the administrator view of an Identity cube if that option has been enabled for the installation, and a certification option allows certified roles and entitlements to be marked as assigned as well. Assignment makes the role or entitlement "sticky" so it can be reprovisioned if any automated process deprovisions it. Roles and Entitlements can also be explicitly revoked and marked as "negative" assignments so they are not automatically reprovisioned by an automated process.

Each Identity object contains lists of AttributeAssignment and RoleAssignment objects inside its Attributes map, under the entry keys AttributeAssignments and RoleAssignments. Loose connections to Application, Link, and Bundle are also shown in this diagram since these Assignment objects reference those other objects through ID and name values.

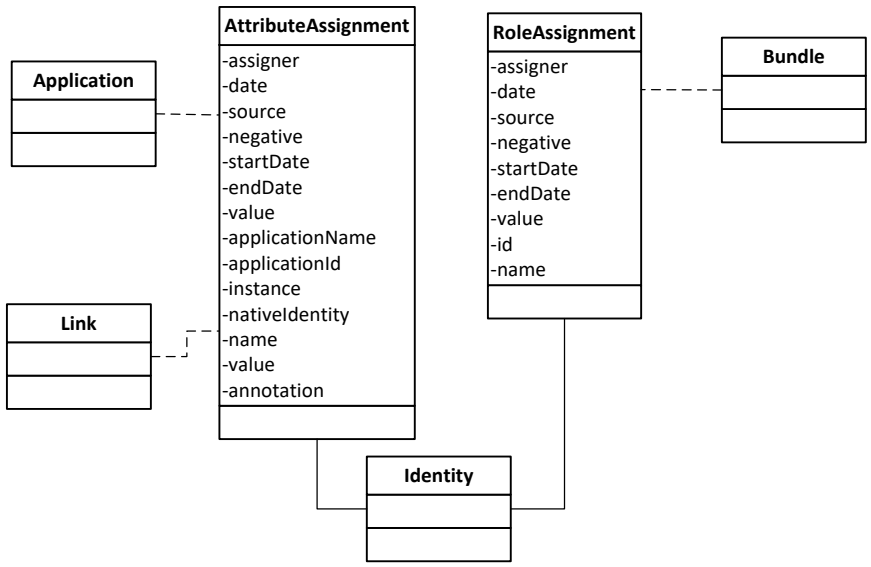


Figure 2: AttributeAssignment and RoleAssignment Objects and their Attribute Relationships

Attributes in the **Assignment** class, and their usages, are shown in this table. They are shown in the diagram above as though they are defined in the AttributeAssignment and RoleAssignment objects, just for simplicity.

Attribute*	Description
assigner	Name of the Identity performing the assignment (can be an abstract name like "system" where an Identity is not directly connected)
date	Date the assignment was made
source	Optional identifier of the source of the assignment – should usually be the string representation of the appropriate sailpoint.object.Source enumeration value (e.g. UI, Workflow, Aggregation, LCM, etc.)
negative	Flag indicating this is assignment has been revoked and should not be reassigned through an automatic assignment rule (though it can still be manually reassigned); this is done to allow a certification decision to override an automatic assignment rule; this is only visible (and can only be turned off once activated) through the Identity XML
startDate	Date on which the assignment starts (sunrise)
endDate	Date on which the assignment terminates (sunset)

This table describes the attributes in the **RoleAssignment** object. (The attributes in Assignment also apply.)

Attribute*	Description
id	ID of the assigned role (bundle)
name	Name of the assigned role

***** RoleAssignment does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **AttributeAssignment** object contains the attributes shown below (in addition to the attributes in the Assignment class).

Attribute*	Description
applicationName	Application name on which the entitlement exists (null for Identity-level attributes)
applicationId	Application ID (allows resolution of application when the application name has changed)
instance	Instance identifier for template applications
nativeIdentity	nativeIdentity on the account link
name	Name of attribute or target of permission
value	Value of attribute or rights of permission (expressed as CSV list)
annotation	Annotation of a permission (undefined for attributes)

***** AttributeAssignment does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Identity Entitlement

The IdentityEntitlement class is used to manage entitlement connections and record certification and request meta data for an Identity. Primarily, this object records how entitlement data has been associated to the Identity – whether it came from an aggregation or from an assignment or detection. This object was new in version 6.0.

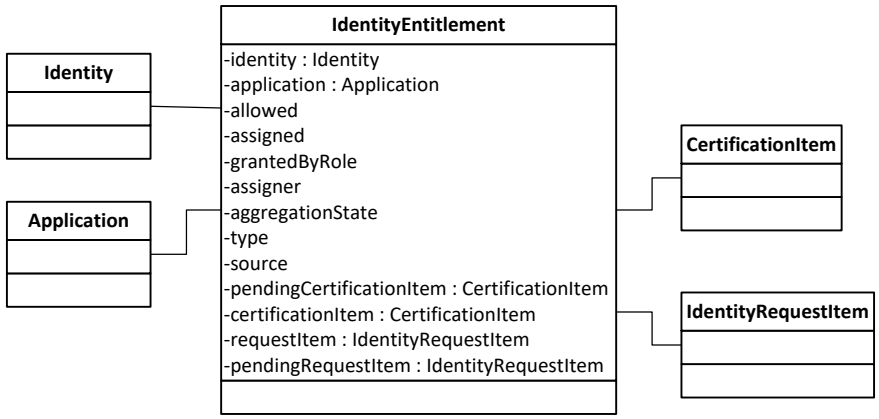


Figure 3: IdentityEntitlement Object and its Attribute Relationships

Attributes on the **IdentityEntitlement** object are described in this table:

Attribute	Description
identity	Identity object to which the attribute is associated
application	Application object from which the entitlement originated (null for assigned and detected roles
allowed	Flag indicating that a detected role is "allowed" as part of the "permits" or "requires" list on one of the Identity's assigned roles; only applies to detected roles
assigned	Flag indicating that the attribute was directly assigned (through LCM or other means); not applicable to Roles (
grantedByRole	Flag indicating that the entitlement was granted indirectly by one of the roles assigned to the user
assigner	displayName of user who assigned the entitlement, if it was assigned; can be null or "system" as well

aggregationState	Enumeration: Connected (found on application on last aggregation) or Disconnected (not found during last application scan – typically means the entitlement was assigned through LCM or other means and not yet reaggregated) Always null for assigned and detected roles
type	Permission or Entitlement; if this represents a Role, type is null
source	Source of the entitlement, typical values are Task or LCM
pendingCertificationItem	Pending CertificationItem object that references this entitlement. The CertificationEntity and parent Certification can be accessed through the CertificationItem
certificationItem	Current CertificationItem object that references this entitlement
requestItem	IdentityRequestItem object referencing this entitlement; from there, other request information is accessible
pendingRequestItem	IdentityRequestItem that is pending

Capability

The Capability Object represents collections of system permissions available to users within the IdentityIQ application itself.

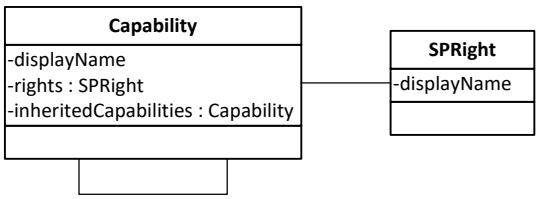


Figure 4: Capability Object and its Attribute Relationships

The table below lists attributes of the **Capability** object and their descriptions.

Attribute	Description
displayName	Name assigned to the Capability
rights	List of SPRight objects (contain low-level, task-based rights within IdentityIQ)
inheritedCapabilities	List of Capability objects that the Capability inherits; rights are also inherited from these objects

The SPRight objects represent specific, highly-granular rights which are tied to various UI page permissions and can be grouped together in Capabilities to grant access permissions to users. Custom SPRights can be created to manage access to specific objects (e.g. custom tasks, reports, etc.) and both core-product SPRights and custom SPRights can be grouped together in custom Capabilities. Additionally, the provided Capabilities can be edited to include different SPRights than their default set of rights.

SPRight objects only have a displayName attribute. When they are specified in the RequiredRights list of any SailPointObject in the system, they automatically govern access to that object, requiring the accessing user to

have that right. The JavaScript in various UI pages also uses SPRights to control which users can access the pages and perform certain functions.

Attribute	Description
displayName	Name assigned to the right

Application

The Application object represents an application defined to IdentityIQ.

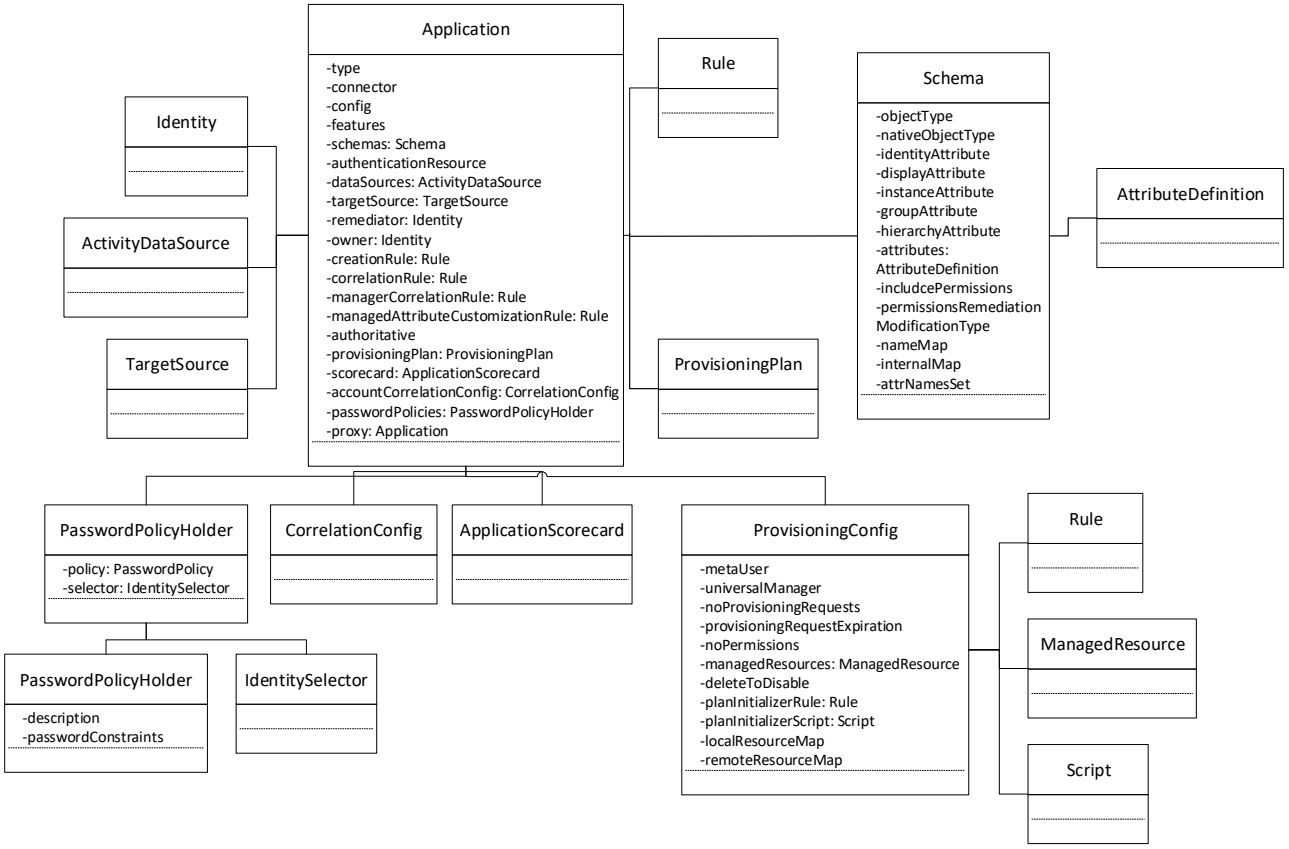


Figure 5: Application Object and its Attribute Relationships

Attributes of the **Application** object and their descriptions are listed in the table below.

Attribute	Description
objectConfig	Searchable and extended attributes for the application (cached for frequent use by persistence layer, so accessed through cacheReference)
type	Application type/ connector type (e.g. LDAP, Active Directory, Delimited File, JDBC, etc.)
connector	Fully qualified name of connector class used by this application
config	Map of attributes used to configure the connection to the application (name/value pairs)

features	List of features supported by the application type (enumerated list); recorded as “featuresString” in the application XML representation
schemas	List of Schema objects (account and group schemas)
authenticationResource	Flag indicating whether application can be used for pass-through authentication
dataSources	List of ActivityDataSource objects (source of application activity data when activity monitoring is enabled)
targetSource	TargetSource object for targeted collection (for applications which support unstructured targets)
remediators	List of Identity objects set as remediators for the application
creationRule correlationRule customizationRule managerCorrelationRule managedAttributeCustomizationRule	Rule objects run during application creation, correlation, etc.
authoritative	Flag indicating whether the application is an authoritative data source
templates	Template objects that represent the provisioning policies for the application (pre-7.0 only)
provisioningForms	List of provisioning policy Form objects (7.0+)
scorecard	Application scorecard object (contains calculated risk score info)
accountCorrelationConfig	CorrelationConfig object – represents the configuration for account correlation – contains application account attribute to Identity attribute assignments (alternate to rule-based correlation)
provisioningConfig	ProvisioningConfig object; holds connector information related to provisioning (for most applications using read/write connectors with the PROVISIONING feature enabled/supported)
passwordPolicies	List of password policies created for the application (stored as a list of PasswordPolicyHolders – see below)
proxy	Secondary application whose connector is used to manage provisioning for this application’s accounts; when set, this application typically does not have a connector and is not required to have a schema – it inherits the proxy applicaton’s schema if it does not have one of its own; this is used for applications managed through a PIM application or through the Cloud Identity Bridge

The **Schema** object describes the account and group definitions supported by the application, including the native object type on the application, the attribute that will be used as the identity, the account’s display attribute, and all the data elements stored in IdentityIQ for accounts on the application. Only two schema types are used: account and group; all applications will have a defined account schema. Attributes of the Schema object and their descriptions are listed in the table below.

Attribute	Description
objectType	Type of object the schema is defining (IdentityIQ internal type -- account or group)
nativeObjectType	Native application object type (i.e. inetOrgPerson)

identityAttribute	Name of the attribute that will be used to uniquely identify an account on the application
displayAttribute	Name of attribute that will be used as display attribute when displaying this account
instanceAttribute	Name of attribute to be used as instance identifier for applications
groupAttribute	Name of attribute in the Account schema whose values will be the ids of the objects in the Group schema
hierarchyAttribute	Name of the attribute in the Group schema whose values will be the ids of other group objects in a hierarchy (usually an inheritance relationship)
attributes	List of AttributeDefinition objects that define the attributes on this schema
includePermissions	Flag indicating whether permissions are part of this schema or not
permissionsRemediationModificationType	Enumerated value specifying how remediation of permissions on this schema can occur (if null, permissions cannot be modified through a remediation request)
nameMap	Map of attributes (name/value pairs) for faster lookup
internalMap	Map of attributes (name/value pairs) as used by connector
attrNames	Hash map of attribute names

Attributes of the **ProvisioningConfig** object and their descriptions are listed in the table below. This is an optional object that describes attributes of the connector used to perform the provisioning task for the application. It is only used for read/write connectors (though the LDAP read/write connector does not use one).

Attribute*	Description
metaUser	Flag indicating if underlying system maintains a “meta” user that requests must be performed against rather than being a simple pass-through
universalManager	Flag indicating whether this connector application can handle provisioning for all applications
noProvisioningRequests	Flag indicating that ProvisioningRequest objects should not be saved when a plan is sent through this connector Prior to 6.0, the default was not to save ProvisioningRequests and the ProvisioningConfig attribute to turn on saving was called saveProvisioningRequests; beginning in 6.0, the default is to save them.
provisioningRequestExpiration	Number of hours a ProvisioningRequest can live before it is considered expired
noPermissions	Flag indicating that the connector does not handle PermissionRequests; only AttributeRequests will be sent to this connector; PermissionRequests will be routed to the unmanaged plan and be displayed in workItems
managedResources	List of ManagedResource objects (applications that will be provisioned using this config)
deleteToDisable	Flag indicating delete requests should be translated into disable requests for all applications managed by this config

planInitializerRule	Rule object that loads the needed data into the ProvisioningPlan (omitting any additional data that is not needed (e.g. name instead of whole Identity object)
planInitializerScript	Script object used to initialize the ProvisioningPlan (alternate to planInitializerRule)
localResourceMap remoteResourceMap	Runtime caches of managed resource name/value pairs

***** ProvisioningConfig does not contain the attributes discussed in the *Objects' Shared Attributes* section.

PasswordPolicyHolder objects match a PasswordPolicy object with an IdentitySelector (defines the identities to whom the passwordPolicy applies). The table below describes the PasswordPolicy object's attributes and their usages.

Attribute	Description
policy	PasswordPolicy object defining the password requirements
selector	IdentitySelector object which identifies the identities for whom this password policy should be used when setting their account passwords

PasswordPolicy objects hold the password rules for setting or changing passwords on managed applications. The table below describes the PasswordPolicy object's attributes and their usages. They are connected to applications through PasswordPolicyHolder objects, which also define the specific subset of identities to whom the policy relates; it is possible to define different password policies for different system users (e.g. admin or privileged accounts can have stronger password requirements).

Attribute	Description
description	Text describing the password policy rules
passwordConstraints	Map of constraints which define the password policy (e.g. must contain 1 number, must contain 2 uppercase letters, must be at least 10 characters)

Link

The Link Object represents an account on an Application correlated to an Identity.

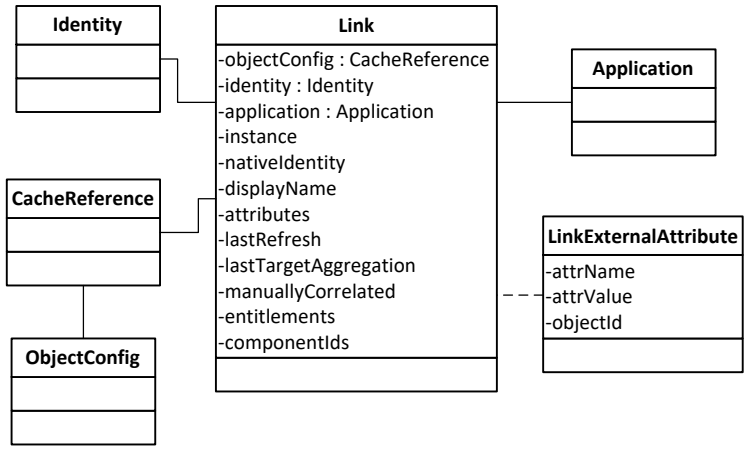


Figure 6: Link Object and Its Attribute Relationships

These are the key attributes on the **Link** object. It connects an Identity object and an Application object.

Attribute	Description
objectConfig	Searchable and extended attributes for the link (cached for frequent use by persistence layer, so accessed through cacheReference)
identity	Identity object to which the account belongs (owning Identity)
application	Application object on which the account exists
instance	the specific instance where the account exists when a single application represents multiple instances
nativeIdentity	“Raw” account identity on the application (e.g.: dn on directory applications)
displayName	Alternate “friendly” name for the account on the application
attributes	Application attribute map (name/value pairs)
lastRefresh	Date the account data was last aggregated into IdentityIQ
lastTargetAggregation	Date the account was last aggregated through target aggregation (specific aggregation of a single record, generally following a provisioning action)
manuallyCorrelated	Flag indicating whether the link was manually correlated to the Identity through the UI; when True, the Link’s Identity correlation will not be altered by future aggregations
entitlements	Flag indicating whether this link contains entitlement attributes
componentIds	String field containing list of other Link IDs (populated only for composite accounts with component accounts)

This table describes key attribute on the **LinkExternalAttribute** object. This object models the multi-valued extended attributes for a Link. Its connection to the Link is through the objectId field, which contains the associated Link’s unique identifier.

Attribute*	Description

attrName	Name of the multi-valued extended attribute
attrValue	Attribute value
objectId	ID value for Link to which the attribute belongs

***** LinkExternalAttribute does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Bundle

The Bundle Object represents IdentityIQ's Roles. Roles can be used to model a variety of concepts, ranging from an organizational structure to a complex collection of application entitlements. In the default IdentityIQ Role configuration, the Profiles attribute on the Bundle is only used with entitlement-level (IT) Roles, while the Permits and Requires attributes apply only to Business Roles. The IdentitySelector object represents assignment rules, which are set on Business Roles for automated assignment (and thus also only applies to Business Roles).

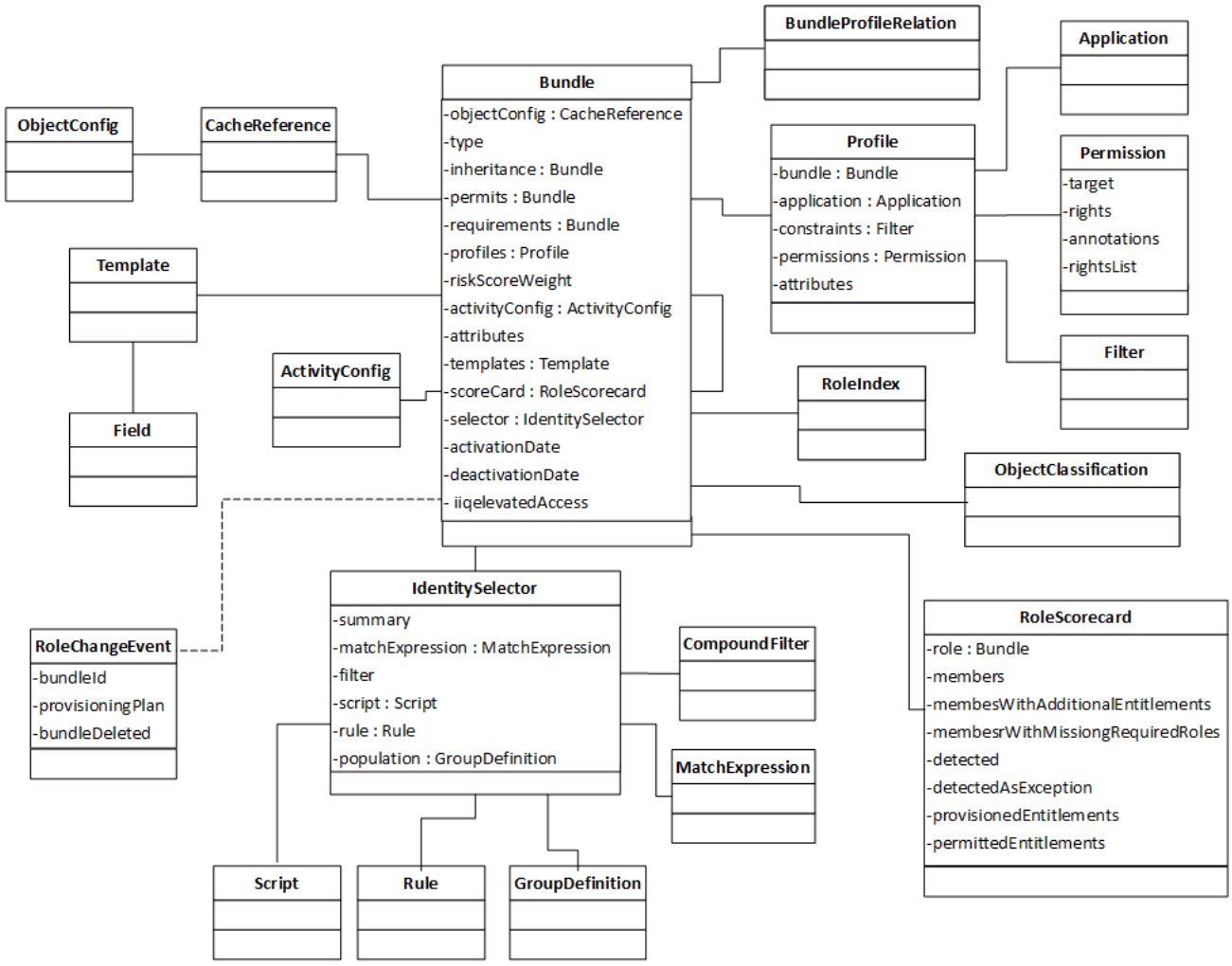


Figure 7: Bundle Object and its Attribute Relationships

Attributes of the **Bundle** object and their descriptions are listed in the table below. Other Bundles specified in the various lists (inheritance, permits, requires) determine the interconnection between Bundles in the Role structure.

Attribute	Description
objectConfig	Searchable and extended attributes for the bundle (cached for frequent use by persistence layer, so accessed through cacheReference)
type	Role type (e.g. Business, IT, etc.)
inheritance	List of Bundle objects from which this Bundle inherits directly
permits	List of Bundle objects that are permitted for an Identity to which this Bundle is assigned
requirements	List of Bundle objects that are required for an Identity to which this Bundle is assigned
profiles	List of Profile objects connected to this Bundle (used for IT roles); Profile objects record sets of access rights on an application; define role detection rules
riskScoreWeight	Weight assigned to this Bundle for Identity Risk Scoring
activityConfig	activityConfig object, describing activity monitoring configuration
attributes	Bundle extended attribute map (name/value pairs)
templates	List of Template objects, representing the provisioning policies specified for this Bundle (currently only one Template per role) (pre-7.0 only)
provisioningForms	List of Form objects representing the provisioning policies specified for this Bundle (currently only 1 per role) (changed from Template in 7.0+)
scorecard	roleScorecard object holding statistics calculated for this Bundle
selector	An IdentitySelector object, specifying the assignment rule for a Business Role (implements automated assignment of this Bundle based on an Identity's attributes)
activationDate	Date this role turns from inactive to active (role sunrise)
deactivationDate	Date this role turns from active to inactive (role sunset)

Attributes of the **Profile** object and their descriptions are listed in the table below. Profile objects record sets of access rights on an application; they are used within the Bundle class to define role detection rules.

Attribute	Description
bundle	Owning Bundle object
application	Target application for the permissions
constraints	List of account attribute values (Filter objects) required by the account; typically either constraints or permissions are used but not both
permissions	List of permission objects, containing the Target and list of system rights
attributes	Profile extended attribute map (name/value pairs)

Attributes of the **IdentitySelector** object and their descriptions are listed in the table below. IdentitySelector represents different ways sets of Identities can be selected for assignment to the role (applies only to roles which can be assigned – i.e. only Business roles in the default role model). Though more than one selection

method can be used at once, usually only one is used. Each identity is evaluated against the IdentitySelector's criteria and if they match it, the role is assigned.

Attribute*	Description
summary	Brief summary of what the selector is doing
matchExpression	Simple list of attribute value or permission matches
filter	Compound filter that can accommodate Boolean operators; may reference both identity and link attributes in any combination
script	Inline script; returns true if the role should be assigned to the user
rule	External rule reference; returns true if the role should be assigned to the user
population	Population reference; filter only applies to identity cube attributes

***** IdentitySelector does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **RoleIndex** object holds statistics about roles. Its values are populated through a series of tasks which first track meta-data about role associations to identities and then calculate statistics for the roles themselves (specifically an Identity Refresh task with the *Refresh role metadata* option selected, followed by a Refresh Role Scorecard task).

Attribute*	Description
bundle	The Bundle object (role) to which these statistics apply
assignedCount	Number of identities which have this role assigned
detectedCount	Number of identities which have this role detected
associatedToRole	Only for detected roles; Flag indicating whether this is a detected role which is currently associated with an assignable role through a permits or requires list
lastCertifiedMembership	Date this role was last included in a role membership certification
lastCertifiedComposition	Date this role was last included in a role composition certification
lastAssigned	Date this role was last assigned to a user
entitlementCount	Number of entitlements directly attached to this role (in its entitlement profile)
entitlementCountInheritance	Number of entitlements attached to this role through its inheritance hierarchy (count of entitlements an identity would get by being associated to this role)

***** RoleIndex does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **RoleChangeEvent** object was introduced in IdentityIQ version 6.4 to support propagation of role changes to identities who already held the role when it was modified. This object tracks changes made to role definitions (the required IT roles for a Business role and the entitlements contained within an IT role), whether additive or subtractive, and stores provisioning plans through which the entitlements on any identity who has the role can be updated. This object does not have a direct connection to a Bundle in that it does not store a reference to the bundle object, but it does contain a bundleId attribute which points to the Bundle object for the changed role. RoleChangeEvents only exist from the time the role is edited until the time the role propagation task processes the event. Attributes of the **RoleChangeEvent** object and their descriptions are listed in the table below.

Attribute	Description

bundleId	The ID value of the role which was changed and for which the propogation activity needs to be performed
provisioningPlan	ProvisioningPlan object describing how to update the identities to reflect the role definition change; this provisioningPlan does not contain a target identity – it is a generic provisioning plan which indicates the action to be performed (e.g. add or remove an entitlement) but the identity details will be calculated as part of the propagation action when the record is processed by the role propagation task
bundleDeleted	A flag which is set to true if the role was deleted (i.e. the "change" was its deletion from the system)

Role Mining

Roles can be defined in IdentityIQ through mining activities which look at known information about identities (typically attributes and entitlements they share in common) and assist in creating role definitions to represent those commonalities.

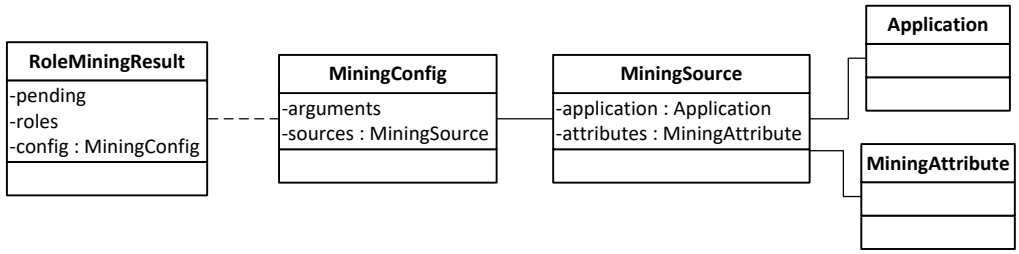


Figure 8: Role Mining objects and their relationships

The **RoleMiningResult** objects hold the results of IT role mining sessions. These mining activities do not automatically create roles; instead, they create collections of data which can be used to create roles as the administrator's discretion. Attributes of the RoleMiningResult object and their descriptions are listed in the table below.

Attribute	Description
pending	Flag indicating the mining task is still running, generating data
roles	List of candidate roles which can be selected and created
config	Internal copy of the MiningConfig which defined the parameters for the roles were identified (may not match the current top-level MiningConfig of the same name if that configuration gets changed)

The **MiningConfig** objects hold the parameters for IT role mining sessions. Attributes of the MiningConfig object and their descriptions are listed in the table below.

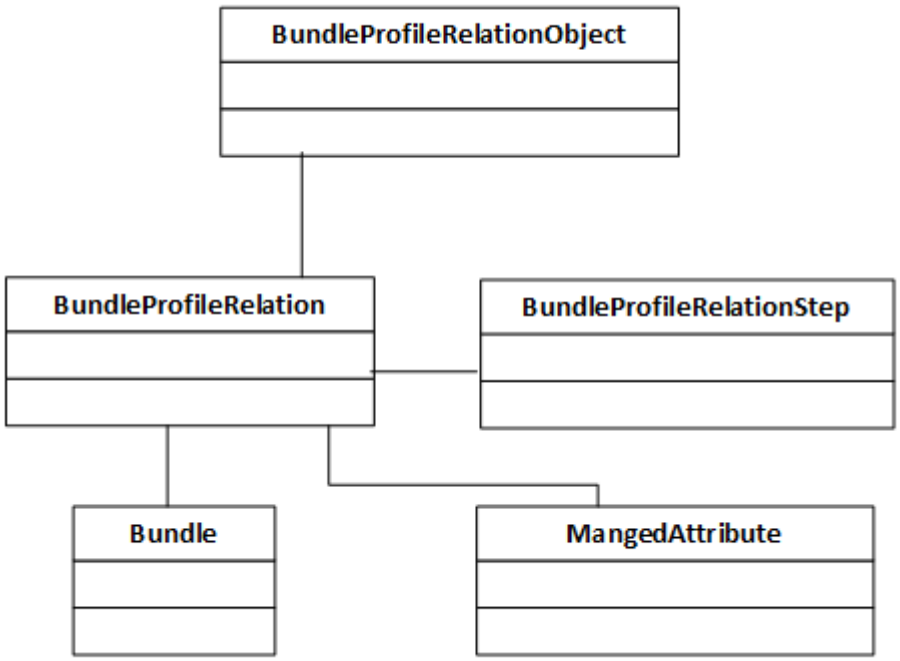
Attribute	Description
arguments	An attributes map of arguments, corresponding to constants defined in the class (see the Javadocs for those constant values)
sources	List of MiningSource objects which define which applications, attributes, and permissions to include in the mining activity

The **MiningSource** objects define which applications, attributes, and permission to include in the mining activity.

Attribute	Description
application	The application to examine in the role mining activity
sources	List of MiningAttribute objects; A MiningAttribute object contains an attribute name from the application, a list of values, a flag indicating whether the attribute is a permission, and a flag indicating whether to negate the condition (e.g. look for: attribute does not contain any of the values in the list).

Role-Entitlement Associations

New in version 8.3. These objects specify the associations between the role (also referred to as a Bundle in the object model) and any granted entitlements. These objects were introduced as part of role enhancement that tracks the relationship of an organization's roles (as defined in IdentityIQ) to the entitlements and permissions in the Entitlement Catalog, by providing a new database table to store the associations between roles and entitlements. This allows roles to be filtered based on details of the entitlements included directly in their profiles, or in the profiles of other roles with a required/permitted/inherited relationship.



- BundleProfileRelation : represent the effective entitlements and permissions that a Bundle has, as determined by analyzing the Bundle's profiles and the profiles of its inheritance and permitted/required graphs.
- BundleProfileRelationObject : events that indicate that a Bundle's entitlements need to be re-evaluated

- BundleProfileRelationStep : describes the chain of relations between roles

BundleProfileRelation attributes:

Attribute	Description
bundleId	The id of the bundle that is declared to be associated (either directly or indirectly) with an entitlement or permission. The entitlement or permission is associated with this bundle because a) this bundle itself has the profile, or b) this bundle is associated to the profile via a chain of inheritance and/or permitted/required relationships.
sourceBundleId	The id of the Bundle that actually has the profile. If bundleId != sourceBundleId, then bundleId is associated with the profile through some relationship (inheritance or permitted/required) with sourceBundleId. If bundleId = = sourceBundleId, then this object is describing an entitlement or permission that bundleId is directly associated with via one of its profiles.
sourceProfileId	The id of the Profile for which the entitlement or permission is declared to be directly associated with.
sourceApplication	The application that the Profile is associated with, and thus also the entitlement or permission is associated with.
type	The type of managed attribute. This can be: a) Entitlement, b) Permission or c) any schema objectType on the given application that maps to an 'account' schema entitlement attribute.
attribute	If this object represents an entitlement attribute, this is the value of an attribute. If this represents a permission, this will either be null or the value of a permission right.
hash	A unique hash created by combining _sourceApplication, _type, _attribute, and _value. It is a simple SHA1 has of those concatenated values. This is used to effectively join to a row in ManagedAttribute with the same hash.
displayValue	If this object represents an entitlement attribute, this is the displayable value of an attribute. If this represents a permission, this will be null.
required	Boolean: true if this profile was found by following any permitted branches. Otherwise, false.

Certification and Certification Group

Certification data is split across several interrelated objects.

- **CertificationGroup**: represents the certification specification created to generate a set of Access Reviews and the high-level tracking group for the Access Reviews; called a “Certification” in the UI.
- **Certification**: represents an individual Access Review; called an “Access Review” in the UI.

- **CertificationEntity**: represents the state of one entity within the certification (an Identity, AccountGroup, etc.).
- **CertificationItem**: represents the state of one certifiable item (e.g., a role, an entitlement) within a CertificationEntity
- **CertificationDefinition**: stores the specification that dictates how the certification should be created and what behavior it should exhibit ((e.g. phase durations, notifications/reminder configuration, type of certification, etc.)
- **CertificationAction**: represents the decision and actions driven by that decision; part of AbstractionCertificationItem class from which Certifiable items (CertificationEntities and CertificationItems) inherit
- **CertificationDelegation**: holds the delegation state of the Certification item; part of AbstractionCertificationItem class from which Certifiable items (CertificationEntities and CertificationItems) inherit
- **CertificationChallenge**: holds challenge information when Identity being certified challenges a revocation decision

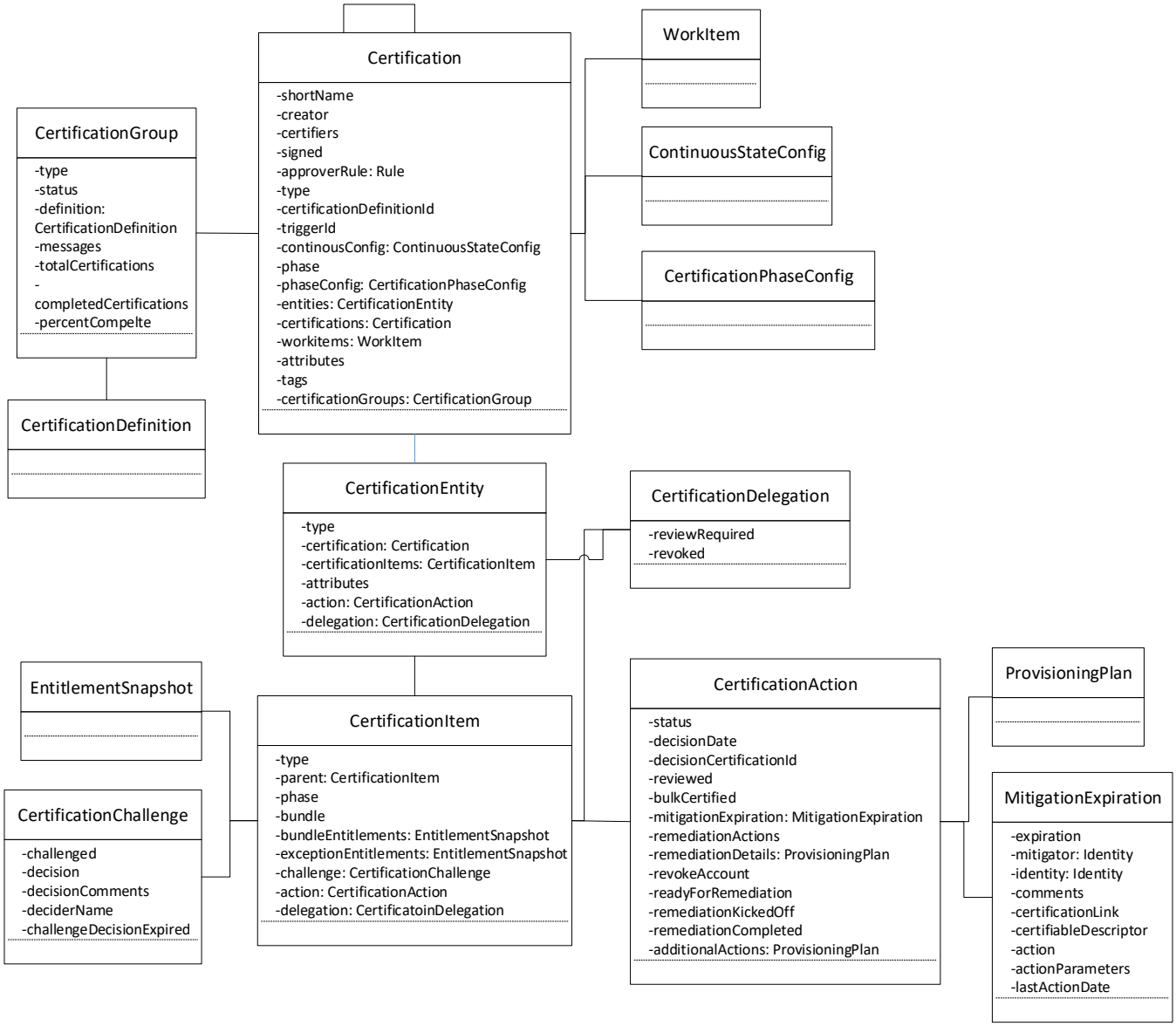


Figure 9: Certification-Related Objects and Relationships

The table below lists and describes attributes of the **CertificationGroup** object.

Attribute	Description
type	Certification type (one of enumerated list)
status	Status of the certification (enumerated list)
definition	CertificationDefinition Object
messages	List of Message objects – warnings or errors attached to CertificationGroup when certification was generated
totalCertifications	Total count of access reviews created for the group
completedCertifications	Total count of the completed access reviews in the group
percentComplete	Percentage of access reviews in the group that are complete

This table describes attributes of the **Certification** object.

Attribute	Description
shortName	Short name of the access review
creator	Name of the Identity who created the access review
certifiers	List of certifier names
signed	Date review was signed off
approverRule	Reference to the rule that runs when the cert is signed to determine if anyone else needs to approve (second-level signoff)
type	Certification type (one of enumerated list)
certificationDefinitionID	Id of the CertificationDefinition that created the review
triggerID	ID of the IdentityTrigger that created this certification if it was generated in response to a certification event
continuousConfig	ContinuousStateConfig object that identifies the states applicable for the certification as specified
phase	Review Phase this review is currently in (enumerated list)
phaseConfig	CertificationPhaseConfig object that lists all phases configured for the certification
certificationEntities	List of CertificationEntity objects in this certification (contains the certification state of each)
certifications	child Certification Objects (if any) created by delegation actions
workItems	List of WorkItem objects created for the access review
attributes	Certification extended attribute map (name/value pairs)
tags	Tags specified for the certification
certificationGroups	List of CertificationGroup objects to which this certification belongs (currently only use single certificationGroup – list is for possible future expansion)

This table describes attributes of the **CertificationEntity** object.

Attribute	Description
type	Type of this certificationEntity (Identity, AccountGroup, DataOwner, BusinessRole)
certification	Parent Certification object for the CertificationEntity
certificationItems	List of CertificationItem objects belonging to the CertificationEntity
attributes	CertificationEntity Attribute map (name/value pairs)

This table describes attributes of the **CertificationItem** object.

Attribute	Description
objectConfig	Searchable and extended attributes (cached for frequent use by persistence layer, so accessed through cacheReference)
type	Type of this item (account group membership, bundle, exception, capability, policy violation, etc.); enumerated list
parent	Parent CertificationEntity object to this CertificationItem

phase	For continuous certifications, certification phase for the item; null for periodic certifications since this is tracked at the Certification level for these
bundle	Name of Bundle for bundle type CertificationItems
bundleEntitlements	List of EntitlementSnapshot objects containing entitlements that exist within Roles for the Identity; for bundle type CertificationItems only
exceptionEntitlements	List of EntitlementSnapshot objects containing entitlements that exist outside of Roles for the Identity; for Exception type CertificationItems only

The table below lists and describes attributes of the **CertificationAction** object.

Attribute	Description
status	Status of item set by certifier (approved, mitigated, remediated, etc.) (enumerated list)
decisionDate	Date on which decision was made
decisionCertificationId	ID of the certification in which the decision was made
reviewed	Flag indicating a delegated action has been reviewed
bulkCertified	Flag indicating whether item was bulk certified
mitigationExpiration	Date at which conditional approval (mitigation) expires
remediationAction	How remediation should be handled (workItem, ticket, provisioning request) (enumerated list)
remediationDetails	Provisioning plan object to be used to complete the remediation (modify or remove accounts)
revokeAccount	For remediations, flag indicating if whole account should be removed rather than individual entitlements
readyForRemediation	Flag used to temporarily mark remediations before process that launches them all runs
remediationKickedOff	Flag indicating whether remediation has been requested
remediationCompleted	Flag indicating data has been reaggregated and remediation has been determined to be completed
additionalActions	Provisioning plan object containing other object requests to be executed with primary plan; includes provisioning of missing required roles and/or revoking of permitted/required roles of a revoked role

The table below lists and describes attributes of the **CertificationDelegation** object.

Attribute	Description
reviewRequired	Flag indicating post-delegation review is required by original certifier (configuration may require post-delegation review regardless of this flag)
revoked	Flag indicating delegation has been revoked and IdentityIQ must cancel the workItem associated with the delegation

The table below lists and describes attributes of the **CertificationChallenge** object.

Attribute	Description

challenged	Flag that indicates the revocation decision is being challenged
decision	Decision made by certifier in response to challenge (enumerated list)
decisionComments	Comments made by certifier to explain decision
deciderName	Name of Identity that made decision on the challenge
challengeDecisionExpired	True if challenge was not addressed by certifier by end of challenge period

An **EntitlementSnapshot** object is very similar to an **EntitlementGroup** object (discussed in the *Identity* section) except that it contains string identifiers for other objects instead of direct references to them so it can be used in archival objects where the direct connections could be broken.

Attribute	Description
Id	Id used for hibernate when entitlementSnapshot is used within a certificationItem
application	Application name (at time of snapshot)
instance	Application instance name
nativeIdentity	Account native identity (unique identifier)
displayName	Account display name (used when native identity is not user friendly
accountOnly	Boolean attribute indicating we are only certifying the user's account (not any entitlements/permissions within it)
permissions	List of permissions from account
attributes	Attributes map (entitlements) from account

When an item is allowed as an exception in a certification process or on a policy violation work item, it is allowed for a fixed period of time; the end date of that time period is the expiration of the mitigation, stored in a MitigationExpiration object. The **MitigationExpiration** object contains these attributes:

Attribute	Description
expiration	Date the allowed exception expires
mitigator	Identity of the user who allowed the exception
identity	Identity on which the exception was allowed
comments	Comments entered at the time the exception was allowed
certificationLink	CertificationLink object which points to the certification and certificationEntity on which the exception was allowed
certifiableDescriptor	CertifiableDescriptor object which represents the item which was allowed as an exception, which will be a policy violation, a role, or an entitlement
action	The action to take when this mitigation expires; an enumeration which is currently either NOTHING (take no action) or NOTIFY_CERTIFIER (send an email to the mitigator using the mitigationExpirationEmailTemplate configured in the system configuration)
actionParameters	Attributes map of parameters to pass to the action
lastActionDate	Date the action was taken (marks the MitigationExpiration object as processed)

Workflow and WorkflowCase

The **Workflow** object represents both the workflow definition and the state of a running workflow. In its "workflow definition" usage, it contains a list of all the steps involved, and its Steps in turn contain a list of Transitions to other steps. Steps can also optionally reference subProcesses which are other **Workflow** objects.

When a Workflow is executed, a copy of its definition is placed within a **WorkflowCase** object, which represents a specific instance of the Workflow; the **Workflow** object's (and **Step** objects') state-tracking components are used within the **WorkflowCase**.

Workflow Approval is tracked across the **Approval**, **ApprovalSet** and **ApprovalItem** Objects.

Many of the objects in this section of the model (as noted below) are XML objects and do not contain the attributes discussed in the Objects' Shared Attributes section of this document.

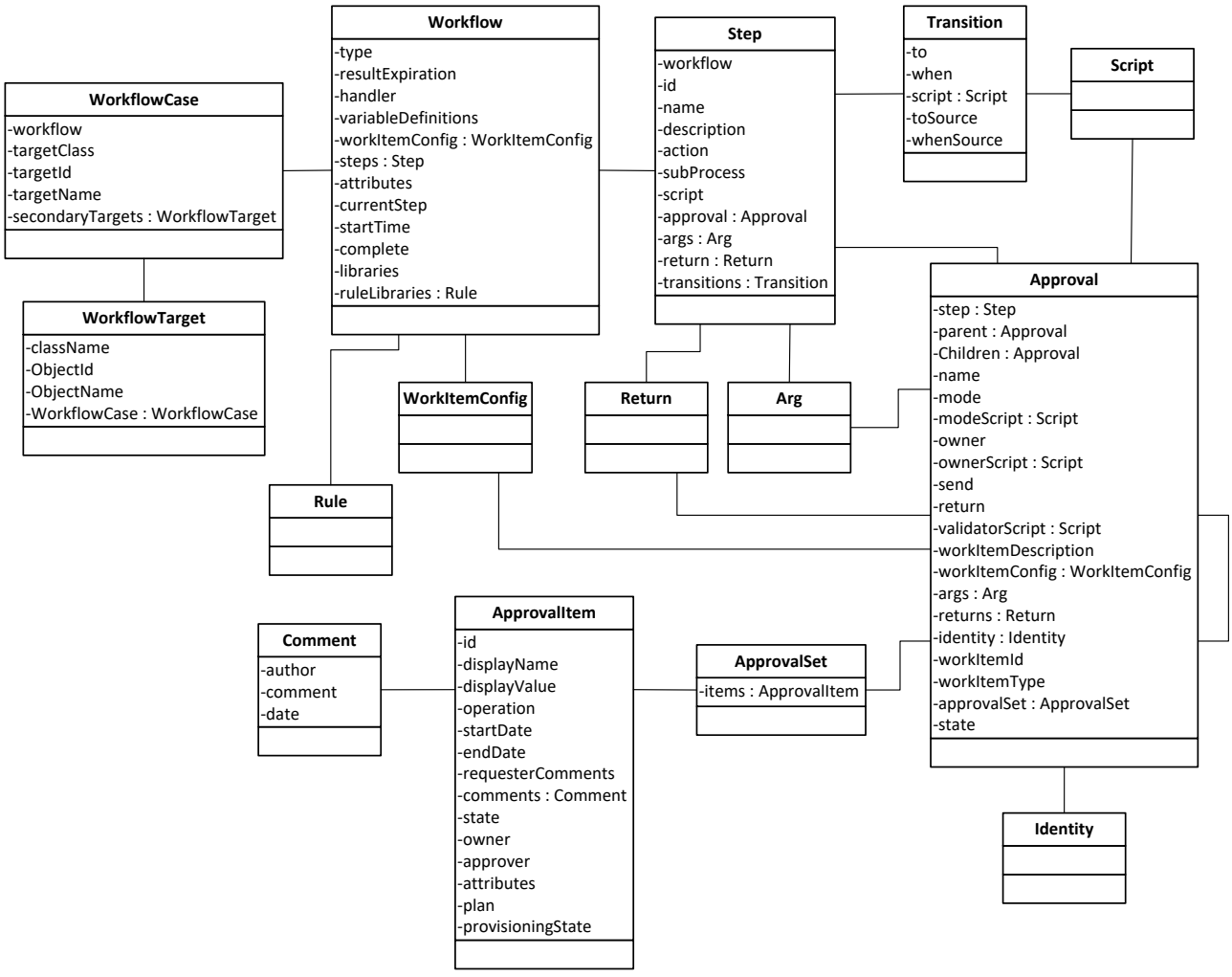


Figure 10: Workflow, WorkflowCase, and Approval Objects and their Attribute Relationships

The **Workflow** object has the following attributes defined:

Attribute	Description
type	String value describing workflow type
resultExpiration	Length of time to keep task results for the process
libraries	CSV of library class names for the library classes containing common methods used by workflows
ruleLibraries	List of Rule libraries to include in all steps that use script and rule calls
handler	Name of handler class to receive notification of progress
variableDefinitions	List of Variables used to declaration of process variables; can be used to supply initial values to those variables
steps	Ordered List of step objects in workflow
attributes	Workflow attribute map (name/value pairs)
currentStep	ID of current step
startTime	Time when workflow started
complete	Flag indicating workflow has finished

The **Step** object defines each step in a Workflow, including its transitions to other steps and the actions within the step. Within a WorkflowCase, the Step also tracks its own current status and the state of the subProcess it executes (if any).

Attribute*	Description
workflow	Parent Workflow object for the step
id	System-generated ID value for step
name	Step name
description	Long description for the step
action	Scriptlet
script	Script object (complex script definition)
subProcess	Subprocess Workflow object (if set, causes Action and Script attributes to be ignored)
approval	Approval object – approval definition for an approval step
args	List of Arg objects (arguments for step handler)
returns	List of Return objects (optional)
transitions	List of Transition objects (transitions to another step)
startTime	Time when the step started
complete	Flag marking the step complete
subcase	WorkflowCase object containing the runtime state of the subprocess

***** Step does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **Transition** object determines progression from its owning step to the next step in the workflow using these attributes:

Attribute*	Description
to	Scriptlet evaluating to name of next step to process
when	Scriptlet condition for simple expressions
script	Script element for more complex expressions

toSource	Source object that parses scriptlet to determine next step
whenSource	Source object that parses scriptlet to evaluate expression

***** Transition does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **WorkflowCase** object tracks a single running instance of a workflow through these attributes:

Attribute	Description
workflow	Workflow object containing workflow definition
targetClass	Database object class of the associated object
targetId	Unique database ID of the associated object
targetName	Optional display name of the associated object
secondaryTargets	List of workflowTarget objects; only needed if more than one target; used instead of class, ID, Name

A **WorkflowTarget** object describes the target object for a workflow when more than one target applies. When only one target applies, this information is stored directly in the WorkflowCase object.

Attribute	Description
className	Database object class of the associated object
objectID	Unique database ID of the associated object
objectName	Optional display name of the associated object
workflowCase	Parent WorkflowCase

The **Approval** object controls actions taken during approval steps, including management of parallel or serial approval processes. These are its attributes:

Attribute*	Description
step	Step that owns the root approval
parent	Parent approval (if multiple approval levels are involved)
children	List of approval objects that are "children" of this one (i.e. included in a joint approval sequence, often falling under the configuration rules of the parent; for example, the parent could specify the approval mode through which all of the children should be presented and interpreted)
name	Optional approval name
mode	Approval evaluation mode (default = serial)
modeScript	Complex script to calculate the mode
owner	Scriptlet to define the owner
ownerScript	Complex script to calculate the owner
send	List of variables to include in the workItem (CSV)
return	List of variables to return from workItem (CSV)
validatorScript	Complex script to do validation on workItem before assimilated
description	Scriptlet defining the workItem Description
workItemConfig	Optional WorkItemConfig to override the one in Workflow
form	Form object for presenting the approval/data request to the user (through the workItem)

args	Optional list of args that define additional information to be passed into workItem or the owner source
returns	Optional list of returns that define how things should be assimilated from workItem back into Workflow case (more powerful but more complex alternative to return csv list)
identity	Identity that will own the workItem generated for this approval
workItemId	Database ID of the workItem generated to handle user interaction
workItemType	Type of workItem generated (enumerated list)
approvalSet	ApprovalSet object that contains the list of ApprovalItems for this approval action
state	Completion state if the approval was acted upon by the owner

***** Approval does not contain the attributes discussed in the *Objects' Shared Attributes* section.

This table describes the attributes on the **ApprovalItem** object. The **ApprovalSet** object is simply a list of ApprovalItem objects.

Attribute*	Description
id	Unique ID used in matching ProvisioningPlan fragments to an Item
displayName	Display name of the attribute that was changed
displayValue	Display value of the attribute that was changed
operation	string representation of ProvisioningPlan.AttributeRequest.Operation or ProvisioningPlan.Operation
startDate	Optional time at which a requested item will be given (sunrise date)
endDate	Optional time at which a requested item will be taken away (sunset date)
requesterComments	Comments made by the requester
comments	List of Comments (in comment objects) added to the item as it goes through the approval process
state	Approval state copied from the workItem
owner	Name of the owner of the approval item
approver	Name of the approver of the approval item (in case owner is a workgroup, this tells which member of workgroup)
attributes	Attribute map (name/value pairs) storing any other interesting information about the approval item
plan	provisioningPlan object to be executed if item is approved
provisioningState	Provisioning status of the approval item (identifies when the item has been successfully applied to the Identity

***** Neither ApprovalItem nor ApprovalSet contains the attributes discussed in the *Objects' Shared Attributes* section.

The table below describes the **WorkItemConfig** object and its attributes. WorkItemConfig is the object that holds various options for the generation and management of work items. In addition to Workflow and Approval, TaskDefinition, Policy, and any other object with an associated background task that needs to create a work item may reference the WorkItemConfig object.

Attribute	Description

escalationStyle	keep track of the desired style of notification while the config is being edited in the UI; helps the UI show only those fields that are necessary and allows it to show clearer labels for the confusing hoursTillEscalation and maxReminders properties
parent	Parent WorkItemConfig, if this one inherits from another
owners	Default list of identities to be assigned the work item
ownerRule	Rule for generating the owner list
notificationEmail	Email template object used for the initial notification of the work item assignment
noWorkItem	Flag to disable generation of work items (used rarely, when a notification is desired but a work item is not needed)
descriptionTemplate	Name of message template for rendering the work item description
hoursTillEscalation	Number of hours work item exists before reminder and escalation process begins (if not >0, reminders and escalations are not sent)
hoursBetweenReminders	Number of hours between reminders
maxReminders	Maximum number of reminders to send before beginning escalation process
reminderEmail	EmailTemplate object used for reminder emails
escalationEmail	EmailTemplate object used for escalation emails
escalationRule	Rule that calculates new owner after max reminders is reached (escalation owner) (if null or returns null, continues to send reminders)

WorkflowRegistry

The WorkflowRegistry object was added in version 6.2 to provide more user-friendly information to users creating workflows through the Business Process Editor. It contains entries which provide help text around each of the available workflow types and the callable workflow library methods so when a user is selecting one of those values in a drop-down listbox in the UI, they can see additional data about the selection. This helps them verify that they have made the selection they intended.

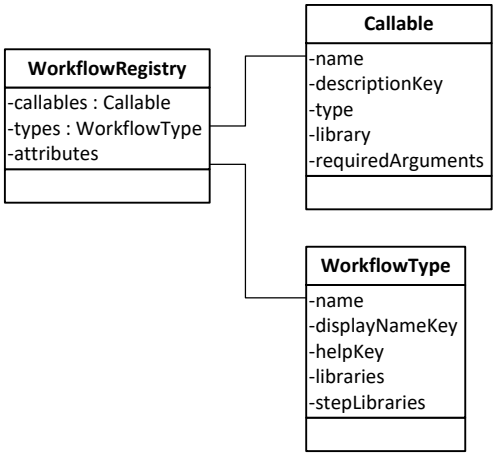


Figure 11: WorkflowRegistry Object and its Attribute Relationships

The attributes of the **WorkflowRegistry** object and their usages are shown in this table:

Attribute	Description

callables	List of Callable objects which represent and describe workflow methods available in the workflow libraries; these can be called from workflow steps
types	List of WorkflowTypes which user can select from in identifying the purpose of this workflow; used in some other UI dropdowns to filter the list of workflows which can be associated to certain system events; can also affect the workflow libraries which are available to the workflow
attributes	Attributes map for storing other registry elements such as the list of possible approval modes and the names of the icons which can be associated to each step to visually represent its type

The **Callable** object contains these attributes:

Attribute*	Description
name	The workflow library method name
descriptionKey	MessageKey containing description text for the method
type	either Step or Initialization, depending on whether the method is used to initialize a variable or to execute functionality in a workflow step
library	The workflow library in which the method can be found
requiredArguments	List of arguments which must be provided to the method; shown in help text

***** Callable does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **WorkflowType** object contains these attributes:

Attribute*	Description
name	Workflow type name
displayNameKey	MessageKey containing user-friendly display name for the type
helpKey	MessageKey containing help text about the workflow type
libraries	List of Workflow Libraries made available to rules of this type; not currently used
stepLibraries	CSV of step libraries, containing new step types to display in the the Add a Step pane, to include when this workflow type is selected. Step Libraries are created as workflows of type="StepLibrary" and template="true". They are not shown in the workflow list in the business process editor but are only used to augment the set of predefined step types which can be created in other workflows. IdentityIQ supports the creation of a step library named "Custom Step Library" in any installation and will automatically make the set of steps defined there available for every workflow type, without requiring that Custom Step Library be added to the stepLibraries list. Customers can alternatively create step libraries with any name they choose and add them to each relevant workflowType's stepLibraries list to make the step set available only to workflows of the specified types. Use "Generic Step Library" as a model for creating custom ones. See the Administration Guide for more information on Step Libraries.

***** WorkflowType does not contain the attributes discussed in the Objects' Shared Attributes section.

Workflow Monitoring

The **ProcessLog** object is used to record workflow metrics when business process monitoring is enabled. It is purposely connected to workflows, workflowcases, and identities only by string name references so there is no

dependency established (i.e. the workflowcase can be deleted or the workflow itself can even be changed, renamed, or deleted without affecting the statistical records collected).

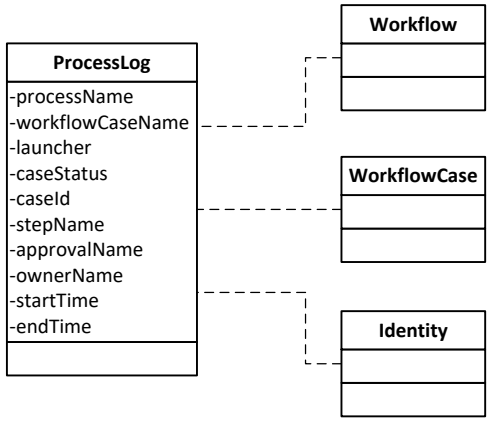


Figure 12: ProcessLog Object and its Attribute Relationships

The **ProcessLog** object contains these attributes:

Attribute	Description
processName	Workflow name
workflowCaseName	Name of the workflow case being monitored (individual running instance of the workflow)
launcher	Name of the user who launched the workflowcase
caseStatus	Current status of the workflowcase
caseId	Unique ID identifying the workflowcase
stepName	Specific step within that workflowcase to which the start/end times apply
approvalName	Name of the approval, if the statistics in the record apply to an approval step
ownerName	Name of the user who is assigned as the approval owner
startTime	Beginning timestamp for step execution; used with endTime to calculate duration
endTime	Ending timestamp for step execution; used with startTime to calculate duration

EmailTemplate

Email Templates are used to send notifications to users. They are used in several sections of the IdentityIQ application, including Workflows and Certifications. The email message is built using the parameters provided in the **EmailTemplate**.

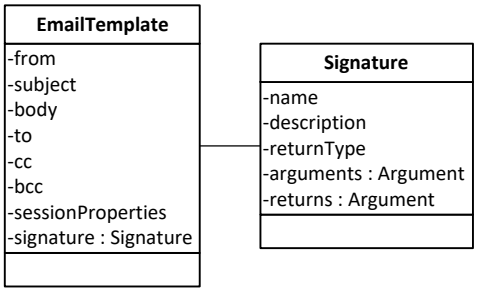


Figure 13: EmailTemplate Object and its Attribute Relationships

These are the attributes of the **EmailTemplate** object and their usages.

Attribute	Description
from	Sender's email address
subject	Subject line for email message
body	Email body content
to	recipient email address
cc	Carbon copy email addresses
bcc	Blind carbon copy email addresses
sessionProperties	Properties for the javax.mail.Session (e.g. mail.smtp.host, etc.)
signature	Formal definition of input variables that can be referenced by the template; also used by Rule objects, in which it also defines returns and the expected return type

MessageTemplate

Conceptually similar to EmailTemplate is the **MessageTemplate** object, which allows formatting with variable substitution of text fields (e.g. descriptions and details fields in work items); it requires and contains no email delivery overhead details. This is an old construct which pre-dates workflows in IdentityIQ, and these are seldom (if ever) used today.

Attribute	Description
text	The message text, which can include Velocity variables for substitution on rendering

Template and Form

As noted in the Bundle and Application sections, prior to version 7.0, provisioning policies for those two object types were represented by the **Template** object. Templates contain **Field** objects that represent the fields required to complete the provisioning activity. If the Field definitions do not provide a value, these fields will be presented to a user on a provisioning policy form, and that user will be required to provide the necessary value for the field.

In 7.0+, provisioning policies are now represented through **Form** objects. Form objects are also used to build custom forms for gathering data from a user through an approval work item (see *Workflow and WorkflowCase*

for more information on the Approval object), as well as for defining identity provisioning policies and for defining report filtering/input forms. Like Templates, Forms are implemented using **Field** objects, but fields on Forms are presented to the user regardless of whether a value is set for the Field.

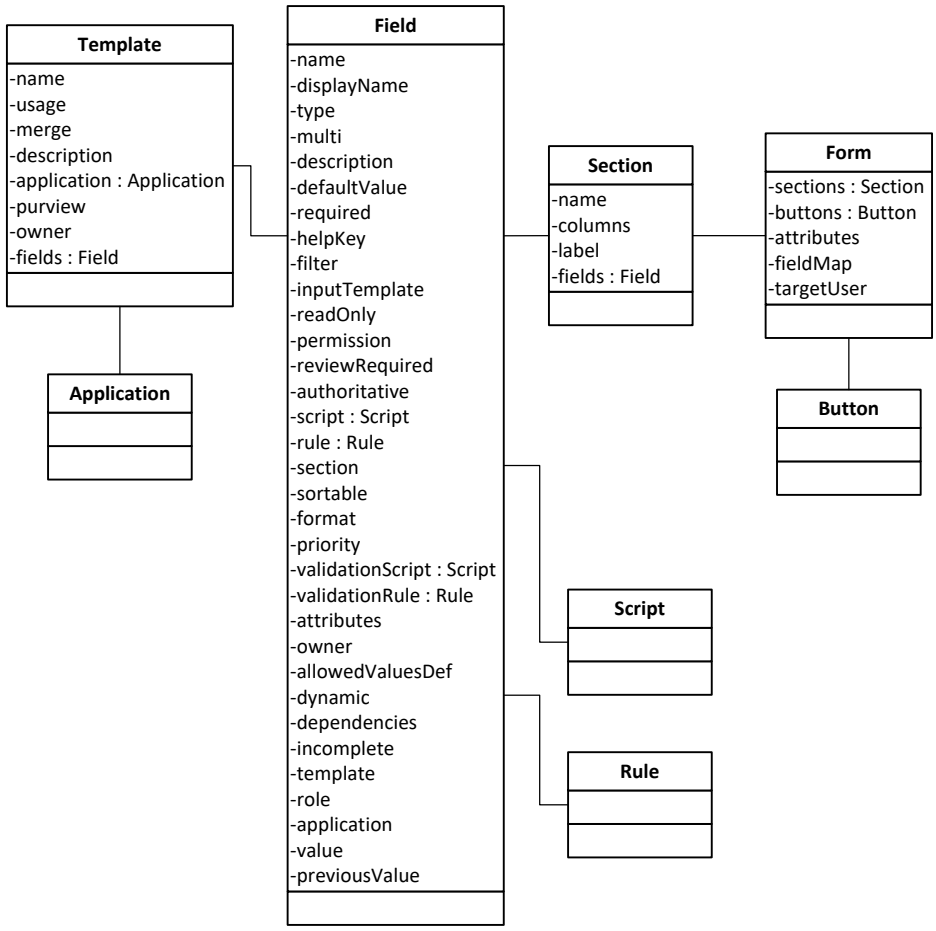


Figure 14: Template, Form and Field Objects and their Attribute Relationships

The **Form** object's attributes and their descriptions are listed in the table below.

Attribute	Description
sections	List of sections defined for the form (subdivides form fields)
buttons	List of action/submission buttons on the form
attributes	Map of attributes (name/value pairs) used in form rendering
fieldMap	Map of fields (name/Field-object pairs) for field object lookup during assimilation
targetUser	Name of user to be shown this form

Attributes of the **Field** object and their descriptions are listed in the table below. **NOTE**: On Forms (but not on Templates), Fields must be defined within Sections.

Attribute*	Description
name	Canonical name for the field
displayName	Display name to use in UI (can also be message key)
type	The variable type for the field (influences rendering)
multi	Flag indicating field can be multi-valued
description	Optional long description for the field
defaultValue	Default value for the field; if set, the field will not be presented on a provisioning policy form to a user unless reviewRequired is set to True; presented on approval form regardless of whether this value is set or not
required	Flag indicating value for this field is required
helpKey	Tooltip help text for the field
filter	Applies to fields with "type" values that are SailPoint Objects (Identity, Permission, etc.) to restrict the list of selectable objects
inputTemplate	Path to an xhtml template used to generate the UI input when a form is created from a signature
readOnly	Flag indicating the field is a read-only field (used on Forms, not on Templates)
permission	Flag indicating if the field represents a permission rather than an attribute (name is permission target and value is the right)
reviewRequired	Flag indicating field should be shown to user even if non-null; used only on Template fields
authoritative	For multi-valued attributes – flag indicating that the value should replace other existing values instead of merge with them
script	Script to calculate field value (priority over defaultValue if both are set)
rule	Rule to calculate field value (priority over script if both are set)
section	Name of form section; used to group fields during form compilation
sortable	For multi-valued attributes – flag indicating the values of the field are sortable
format	Additional info about how field values are stored (e.g. for multi-valued – store as list or CSV)
priority	Number that influences field presentation order (Template fields only)
validationScript	Validation script for field
validationRule	Validation rule for field (alternative to validationScript to make logic reusable for multiple fields)
attributes	Map of extended attributes (name/value pairs) that can influence the renderer
owner	Dynamic value to define field owner (Template felds only)
allowedValuesdef	Dynamic value to specify the allowed values for the field
dynamic	Indicates the field is dynamic (re-runs the allowed values script any time a dependency field changes)

dependencies	List of fields referenced by the scripts in this field (so those other fields will be evaluated before this one)
incomplete	Flag indicating field does not have dependencies met and should not yet be rendered (on workflow forms only)
template	Name of template for field
role	Name of role to which the field's template applies
application	Name of application to which the field's template applies
value	Current value of the field at any point in the update process
previousValue	Previous value of the field (if applicable) during the update process

***** Field does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **Section** object's attributes and their descriptions are listed in the table below.

Attribute*	Description
name	Canonical name for the section
columns	Number of columns into which the form fields should be divided within the section; fields will be placed in columns from left to right and then top to bottom, so a 2-column section with 4 fields will appear like this: Field 1 Field 2 Field 3 Field 4
label	Text or message key to display at the top of the section as a header (optional)
fields	List of field objects within the section

***** Section does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Attributes of the **Template** object and their descriptions are listed in the table below. Templates define role and application provisioning policies – specifically, the fields that are necessary to successfully complete the provisioning request – in pre-7.0 versions of IdentityIQ.

Attribute*	Description
name	name assigned to the specific provisioning policy
usage	Only used on application templates; identifies whether it is applies to the application's create, update, or delete provisioning policy (assumed "create" if null for backward compatability with versions that only included create provisioning policies) Currently, role templates are only for role assignment (update/delete policies don't exist), so this attribute is not used for those templates
merge	Flag that indicates template fields are to be merged with profile rather than replacing it; for role templates only
description	Text that describes what the template does
application	Reference to the associated Application object
owner	Provisioning policy form owner (also applies to fields unless field has a specific owner listed)
fields	List of Field objects for provisioning policy

***** Template does not contain the attributes discussed in the *Objects' Shared Attributes* section.

WorkItem

WorkItems represent manual tasks assigned to an Identity within IdentityIQ. They can be created by a Certification process, a Workflow, or an action taken through Lifecycle Manager, among others.

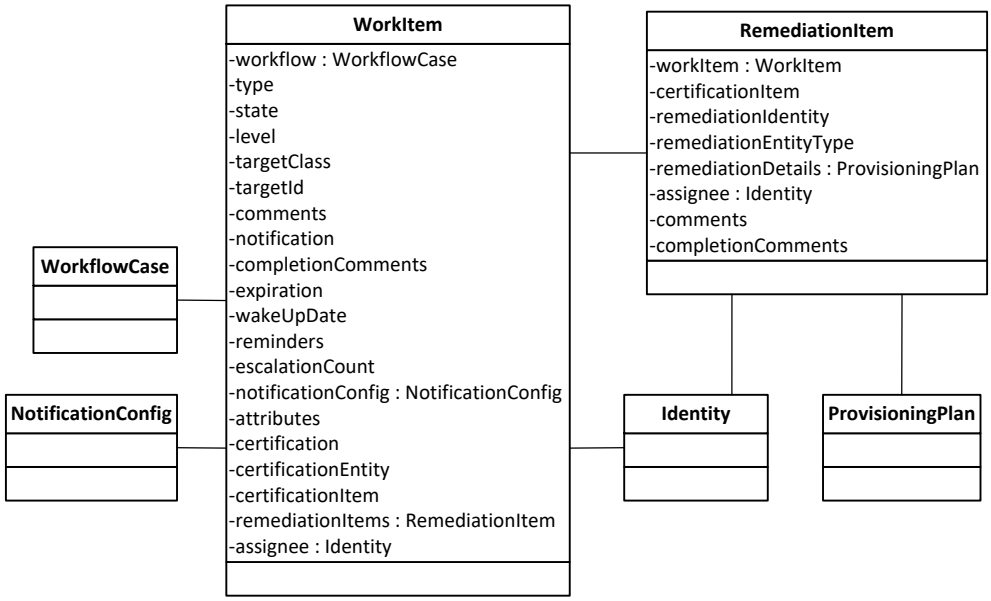


Figure 15: WorkItem Object and its Attribute Relationships

This table lists the attributes (and their purposes) of the **WorkItem** object.

Attribute	Description
workflow	WorkflowCase object that generated the workItem (only for workItems created by Workflows)
type	Type of workItem (Enumerated list; eg: Generic, Certification, Remediation, Challenge, PolicyViolation, etc.)
state	Completion state (null until completed; then shows status information such as finished, rejected, etc.)
level	Severity level of the item
targetClass	Database object class of the associated object, if any; used for Approval, Signoff, and PolicyViolation items
targetId	Unique database ID of the associated object
comments	List of verbose comments that describe what the owner of the workItem is supposed to do
notification	Date the last notification was sent for this workItem
completionComments	Comments the owner may leave after completing the workItem
expiration	Date the workItem expires (reminders get sent after this date; value reset after reminder sent)

wakeUpDate	Date next reminder should be sent (for workItems that have reminders such as certification and remediation requests if configured for certification)
reminders	Number of reminder messages that have been sent
escalationCount	Number of times this workItem has been escalated
notificationConfig	NotificationConfig object specifying configuration of reminders/escalations for workItem
attributes	WorkItem extended attribute map (name/value pairs)
certification	Associated certification object ID (if workItem created for Certification)
certificationEntity	Associated certificationEntity object ID (if workItem created for Certification)
certificationItem	Associated certificationItem object ID (if workItem created for Certification)
remediationItems	List of RemediationItem objects (for remediations only)
assignee	Identity to whom this WorkItem is assigned (only set when Owner is a workgroup; otherwise workItem is assigned to Owner)

WorkItems for remediations have additional data associated with them through the RemediationItems objects they contain. The **RemediationItem** object's attributes and their usages are listed in the table below.

Attribute	Description
workItem	workItem object to which the RemediationItem belongs
certificationItem	ID of the certificationItem being remediated
remediationIdentity	Name of Identity for which the entitlement needs to be remediated
remediationEntityType	Type for the remediation entity (enumerated list: Identity, Account Group, Profile, Business Role)
remediationDetails	Provisioning Plan for the remediation
comments	List of comments on this item
completionComments	Comments entered at completion time for this remediation item
assignee	Identity object to whom the RemediationItem is assigned

Policy and PolicyViolation

The Policy object encapsulates the definition of a company policy in areas such as separation of duties, system activities, and risk. IdentityIQ checks whether Identities are in violation of these policies and reports violations on the Identity Cube (through the PolicyViolation object).

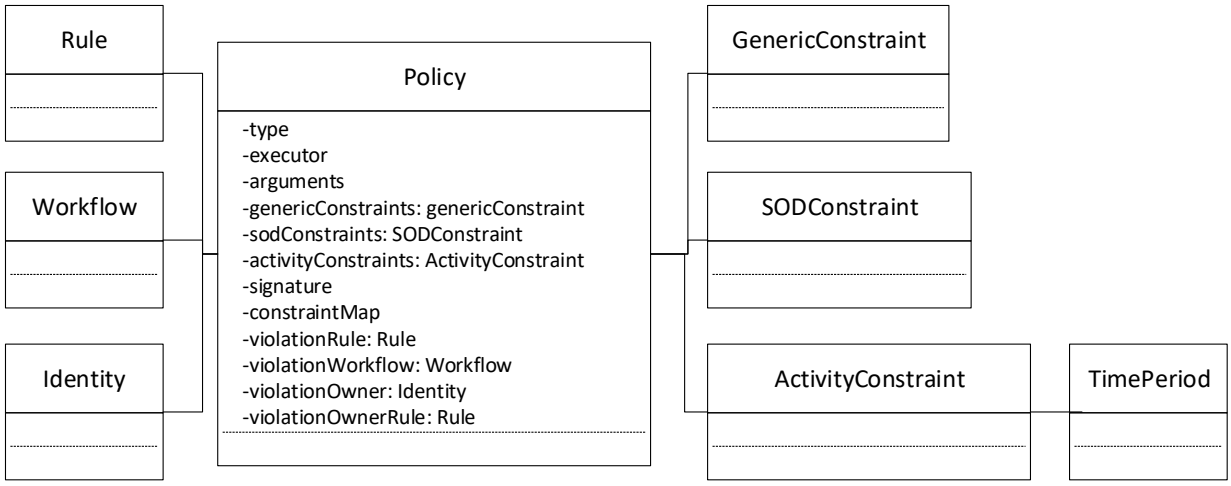


Figure 9: Policy Object and its Attribute Relationships

The table below describes the key attributes on the **Policy** object.

Attribute	Description
type	String value describing policy type
executor	Fully qualified class name of PolicyExecutor implementation
arguments	Policy attribute map (name/value pairs)
genericConstraints, sodContstraints, activityConstraints	Lists of genericConstraint, sodConstraint, or activityConstraint objects representing the set of constraints on the policy (only one of these will be populated based on the policy type)
signature	Defines configuration form for the policy when a simple policy is defined; rendered automatically by policy pages with results left in attributes map
constraintMap	Runtime map of name/value pairs for looking up constraints by ID
violationRule	Rule object for violation formatting
violationWorkflow	Workflow object that handles violation
violationOwner	Identity object for the Identity who owns the violations
violationOwnerRule	Rule object that determines violation owner (alternative to violationOwner)

TimePeriod objects record date ranges which define certain periods of time to IdentityIQ (quarters, workdays vs weekends, office hours vs. not). These are used in defining activity policies and are referenced by the ActivityConstraints objects; activity policies can be constrained to apply to only activities which occur within a selected time period. Time Periods can be configured in the user interface through the System Setup pages.

Attribute	Description
classifer	Classifier type (enumeration: DateRange, DateSet, DaysOfWeek, TimeOfDayRange)
initParameters	Attributes map of initialization parameters for the time period (specifies the details of the time period – the relevant date values or time values)

The PolicyViolation object records Identities' violations of configured policies.

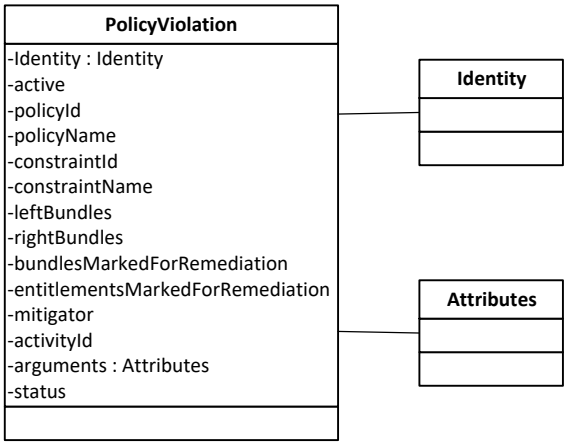


Figure 16: PolicyViolation Object and its Attribute Relationships

The table below describes the key attributes on the **PolicyViolation** object.

Attribute	Description
identity	Identity object found in violation of the policy
active	Flag marking violations as active or inactive
policyId	ID value of the Policy object violated
policyName	Name of the Policy object violated
constraintId	ID value of the Base Constraint within the policy
constraintName	Name of the BaseConstraint
leftBundles	For SOD violations – name of the "left" business role that was in conflict
rightBundles	For SOD violations – name of the "right" business role that was in conflict
bundlesMarkedForRemediation	For SOD violations – names of bundles marked to be remediated (can be left or right; stored as csv list)
entitlementsMarkedForRemediation	For Entitlement SOD violations – value of entitlements to be remediated (CSV list: application, name, value)
mitigator	Name of Identity that performed a mitigation on this violation
activityId	Unique ID of the associated application activity object
arguments	Optional arguments to the violation (name/value pairs)
status	Current status of policy violation

Group and Population

Groups are created based on a single attribute shared across a set of Identities. Multiple Groups are usually created at once, since IdentityIQ creates one Group per value of the attribute. (This is distinct from the account groups defined on an application to manage application permissions; these are now tracked as ManagedAttributes, discussed in the next section.) Populations are created based on an Advanced Analytics query that can contain one or more criteria; a single population is created at a time. The criteria for generating

a Group or Population are stored in a GroupDefinition object. The GroupFactory object relates only to Groups and identifies the attribute used to create the groups. The GroupIndex object stores the statistics for each Group (number of members, policy violations, composite risk score, etc.).

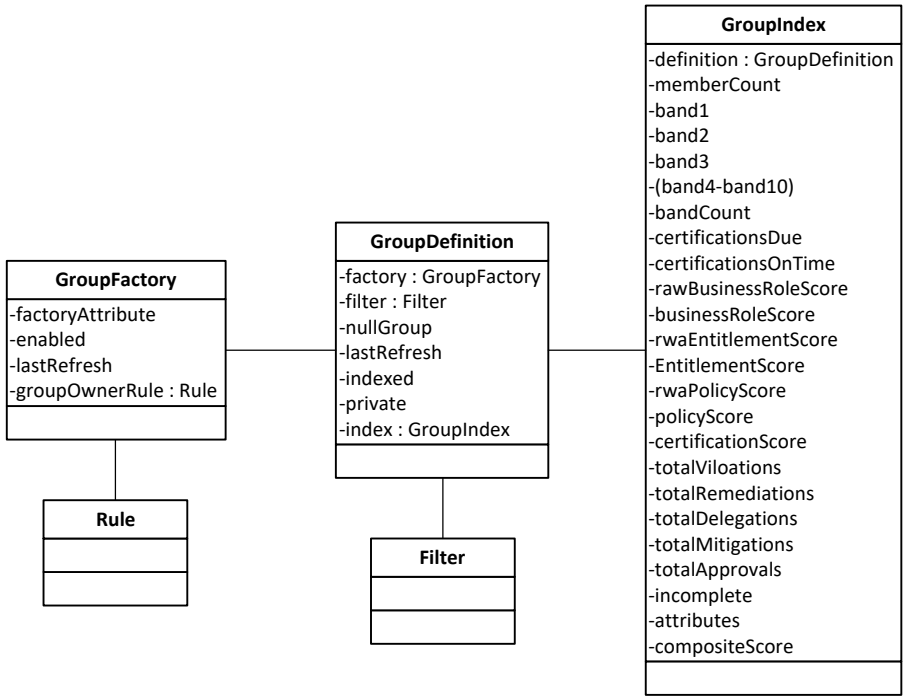


Figure 17: Group-related Objects and their Attribute Relationships

This table describes the attributes on the **GroupDefinition** object.

Attribute	Description
factory	The GroupFactory object used to create the Group Definition (null for Populations)
filter	The filter object for this definition,which specifies the selection criteria for the identities in the group
nullGroup	Flag for Groups that represent a null value of the Group Factory Attribute
lastRefresh	Date of last time Group Refresh task ran to refresh this group
indexed	Flag indicating whether to generate GroupIndex for the group
private	Flag indicating whether the owner of the Group is the only user who can use/edit/delete the Group
index	GroupIndex object for most recent set of indexes created for the group (more than one GroupIndex may exist for a Group to track history)

This table describes the attributes on the **GroupFactory** object.

Attribute	Description
factoryAttribute	Name of Identity Attribute used to create the groups
enabled	Flag indicating whether the GroupFactory is enabled or disabled

lastRefresh	Date of last time Group Refresh task ran to refresh the groupFactory
groupOwnerRule	Rule object for assigning owners to each Group

This table describes the attributes on the **GroupIndex** object. Although total counts and average scores in all categories are kept for the group, currently only the policy violation count and composite score average for the group are displayed in the user interface.

Attribute	Description
definition	GroupDefinition object for the group
memberCount	Number of members in the group at the time the index object was generated
band1 – band10	Numeric values representing the scoring divisions into groups (default is to use 3: low, medium, high; 5 and 10 are also potentially common); presently configured with max of 10
bandCount	Number of bands with valid values (number in use)
certificationsDue	Number of certifications due for the group
certificationsOnTime	Number of certifications owned by the group that were completed on time
rawBusinessRoleScore	Score based on the roles assigned to or detected for the identities, not compensated by certification
businessRoleScore	Score based on the roles assigned to or detected for the identities, compensated by certification
rawEntitlementScore	Score based on the extra entitlements discovered for the identities, not compensated by certification
entitlementScore	Score based on the extra entitlements discovered for the identities, compensated by certification
rawPolicyScore	Score based on policy violations for the identities, without compensation
policyScore	Score based on policy violations for the identities, with compensation
certificationScore	Score based on certification history (time since last certification, number of mitigations/remediations during certification)
totalViolations	Count of policy violations for group
totalRemediations	Count of remediations for group
totalDelegations	Count of certification delegations for group
totalMitigations	Count of violation mitigations for group
totalApprovals	Count of certification approvals for group
incomplete	Flag indicating full score/index could not be calculated for some reason
attributes	Attribute map (name/value pairs) containing scores and statistics
compositeScore	Composite score for the group

Rule

A Rule contains a script that can be executed to perform custom logic. Each rule declares the language in which it is written, the source code for the rule, and context about how the rule should be run.

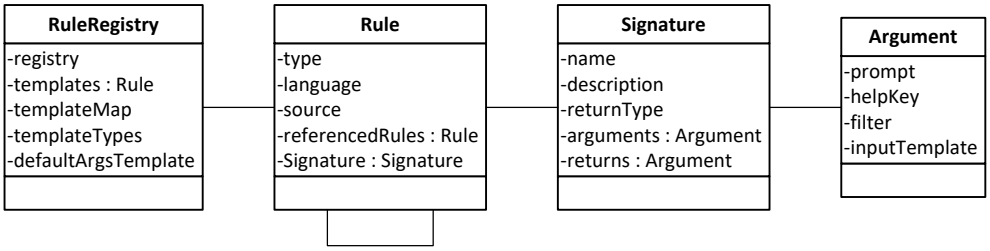


Figure 18: Rule Object and its Attribute Relationships

This table describes the **Rule** object's attributes and their usages.

Attribute	Description
type	Enumerated list of rule types defined for IdentityIQ
language	Name of language used to write the rule; the default and only currently supported language is BeanShell
source	String source code for the rule
referencedRules	List of Rule objects referenced in the source code of this rule
signature	Signature object containing meta data describing the rule's inputs and outputs

The **RuleRegistry** object records all of the different types of rules which can be created in IdentityIQ along with signatures and descriptions for each. This is used by the user interface to populate the Rule Editor's description, arguments, type, and return data, which help rule developers understand the rule type and how to write the rule. This information is automatically copied from the RuleRegistry entry to the Rule object when a rule is created using the Rule Editor in the user interface.

In a very small number of cases, the Rule Registry may contain the actual rule logic; these are called RuleCallouts. A RuleCallout is always used for an auto-create user rule (creates an Identity when a user signs in through pass-through authentication but does not yet have an identity in IdentityIQ). A RuleCallout can also be used to specify a default Identity Correlation rule or a default Identity Refresh rule for the environment.

This table describes the **RuleRegistry** object's attributes and their usages.

Attribute	Description
registry	List of ruleCallouts configured for the environment (may not exist if no rule callouts have been defined).
templates	List of Rules (no source included), specifying type, signature, and description
templateMap	A map of rule types and corresponding rule templates, held in memory and used to retrieve the required template when needed; not persisted
templateTypes	List of the rule types for which templates exist; held in memory; not persisted

defaultArgsTemplate	A Rule template stored in the RuleRegistry identified by the name “Default Arguments Template” which specifies the arguments provided to every rule (currently *context* and *log*); these arguments are added to the signature for any rule viewed in the rule editor, and to the rule itself when the rule is saved, so they do not need to be listed in the signature of each template in the RuleRegistry list (though they are often explicitly listed in each rule signature anyway)

Managed Attribute

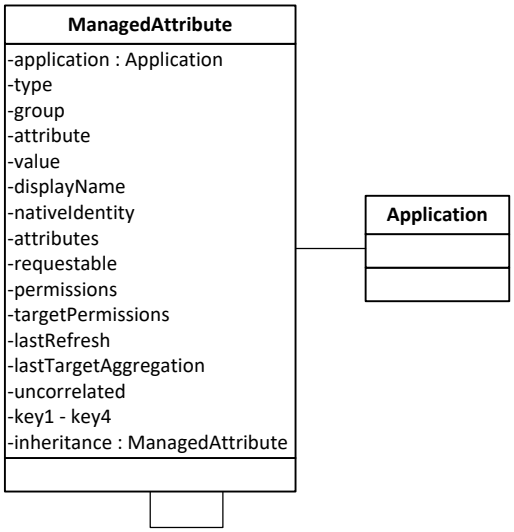


Figure 19: ManagedAttribute object and its attribute relationships

ManagedAttribute represents attribute or permission values on applications that are managed by IdentityIQ. This serves as a registry to look up managed attributes. In addition to other managed attributes (entitlements and permissions), account groups are now tracked here (as managedAttributes of type: Entitlement with the group flag = true). The AccountGroup object that existed in IdentityIQ releases prior to 6.0 has gone away. ManagedAttributes are now displayed in the UI as the Entitlement Catalog.

Attribute	Description
application	The application object from which the managed attribute was discovered
type	Type of managed attribute (enum: Entitlement or Permission)
group	Boolean indicating that this object represents the value of a group attribute (corresponds to AccountGroup in previous IdentityIQ releases)
attribute	For entitlements, name of the attribute in the account schema For permissions, name of permission target
value	For entitlements: value of attribute For permissions, value of the permission right (or can be null)
displayName	Alternate value to display for real attribute value (typically used when value is a DN)

nativeIdentity	The “raw” group identity (e.g. for directories, the DN)
attributes	Attribute map (name/value pairs) of extended attributes – stores extended Hibernate attributes promoted to columns for searching and some internal attributes such as the list of localized descriptions
requestable	Boolean indicating the attribute can be requested/provisioned
permissions	List of entitlements implied by group membership (only set for managedAttributes representing groups); normally set to the value of the “directPermissions” attribute in the ResourceObject returned by the connector
targetPermissions	Permissions added to the group by target aggregations (not overridden by group aggregation); only set for managedAttributes that represent groups
lastRefresh	Date account attributes were last reset
lastTargetAggregation	Date of last target aggregation
uncorrelated	Flag set when referenceAttribute value is found but cannot be correlated to a ResourceObject with that name
key1-key4	Correlation keys; key values pulled out of the attributes map during aggregation and maintained for searching; changed only during aggregation – not kept in sync with changes to the attributes map otherwise
inheritance	List of other managedAttributes from which this managedAttribute inherits (if any); only relevant for group attributes and is used to represent hierarchical groups (object inherits the two permission lists – permissions and targetPermissions – of the parent(s))

Localized Attribute

The **LocalizedAttribute** object records localized (region/language-specific) descriptions for applications, roles, managed attributes, and policies. This construct was introduced in version 6.0.

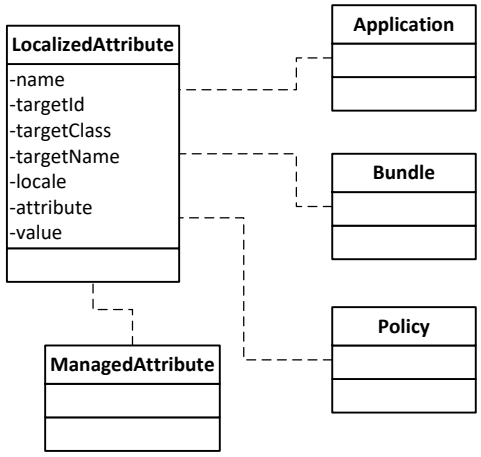


Figure 20: LocalizedAttribute Object and its Attribute Relationships

LocalizedAttributes are only connected to other objects by the string ID and Name (targetId, targetName) of the associated object, not by direct references between the objects.

LocalizedAttribute contains the attributes listed in the table below.

Attribute	Description
name	Name; used for lookup in console or debug pages
targetId	ObjectID of the object to which the localized attribute belongs (i.e. the ID of the application, role, managed attribute, or policy)
targetName	Name of the object to which the localized attribute belongs
targetClass	Class type of the object to which the localized attribute belongs (Application, Bundle, ManagedAttribute, Policy)
locale	Name of the Locale (java.util.Locale) to which the value applies (e.g. en_US)
attribute	Attribute being localized (currently always description)
value	Localized value for attribute

Quick Links and Categories

In version 7.0+, QuickLinks are presented in a side panel that provides instant access to the QuickLinks' functions from any page in the IdentityIQ UI. QuickLinks can also be represented with QuickLink Cards on the home page for any user. Out of the box, the QuickLinks are subdivided into one to three categories: *My Tasks* for the compliance QuickLinks and, if Lifecycle Manager is enabled, *Manage Access*, and *Manage Identity*. Additional custom categories can be added through system configuration entries and additional links can be added to any of the categories.

User access to all QuickLinks is managed through associations to DynamicScope objects which specify populations of users through inclusion lists, exclusion lists, and IdentitySelector definitions. In version 7.0, a new object called QuickLinkOptions was introduced in the QuickLink - DynamicScope relationship; it manages key details about each quicklink and its relationship to the DynamicScope. In version 7.1, the QuickLinkOptions object was merged into the QuickLink object so it is no longer a top-level object of its own, though the relationships between the objects remain the same.

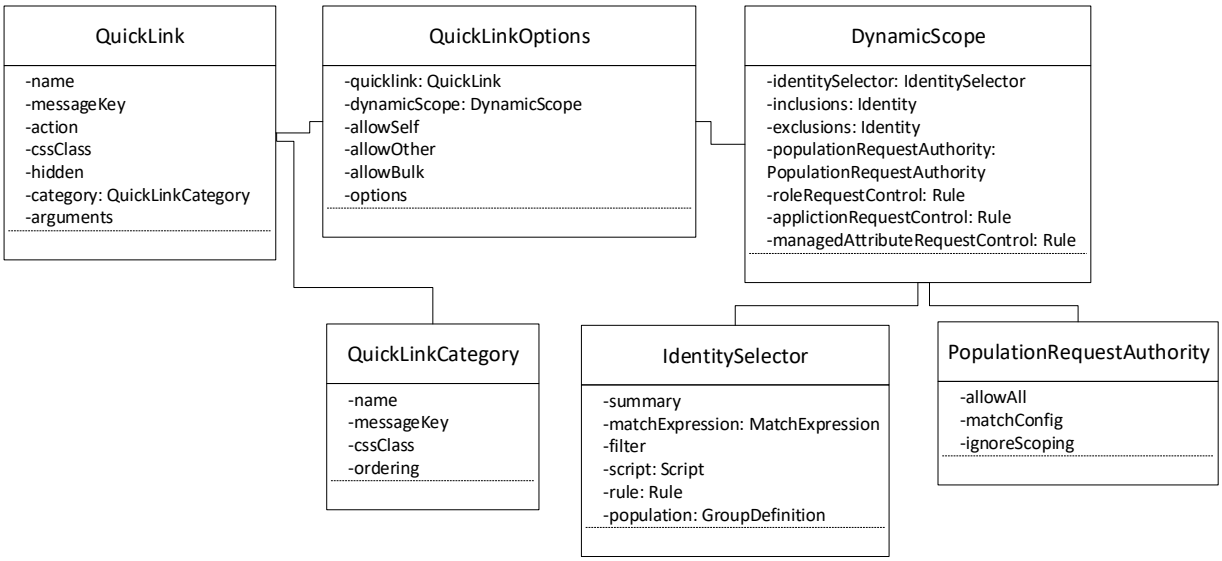


Figure 21: QuickLink Objects and their Attribute Relationships

The **QuickLink** object represents an individual link on the dashboard and contains these attributes:

Attribute	Description
name	Canonical name for the link
messageKey	Text or message key to display for link
action	What to do when the quicklink is clicked; choices include workflow (run the named workflow), external (load an external URL), or the name of action referenced in faces-config.xml that directs the controller to another page in IdentityIQ
cssClass	CSS class to use for rendering links in the Quicklink sidebar menu
hidden	Flag indicating whether the link should be hidden or displayed on the quicklink menu panel
category	Name of the QuickLinkCategory to which the link belongs; this determines the group in which the quicklink is included in the quicklink menu panel
arguments	Arguments used in rendering or executing the link, including: • displayCount (whether or not to display the count of related objects) • countScript (script that calculated the count of related objects) • workflowName (name of workflow to run when link is clicked) • workflowSuccess (message to display when workflow is successfully launched) • displayText (whether or not to display text from textScript) • textScript (script that determines text to display) • url (url of external web page opened by quickLink) For workflow quickLinks, arguments to the workflow are passed through this arguments map
ordering	Order in which the link should appear within the category

The **QuickLinkOptions** object connects a QuickLink to a DynamicScope and sets the parameters that apply to the Quicklink when accessed by that population of users.

Attribute	Description
quickLink	Reference to the QuickLink object to which this QuickLinkOption object applies; each QuickLinkOption contains only one QuickLink reference
dynamicScope	Reference to the DynamicScope objects which control the set of users who can see the quickLink; each QuickLinkOption object contains a single dynamicScope reference
allowSelf	Flag identifying that the dynamicScope population can use this quickLink for self-service requests or operations
allowOther	Flag identifying that the dynamicScope population can use this quickLink to take the action on behalf of another user
allowBulk	Flag identifying that the dynamicScope population can use this quickLink on behalf of multiple other users in a single request

options	An attributes map of other options which apply to the dynamicScope population's ability to use the quickLink (only relevant for the OOTB Lifecycle Manager QuickLinks)

The **QuickLinkCategory** object represents a category, or set, of links on the dashboard and contains these attributes:

Attribute*	Description
name	Canonical name of category
messageKey	Text or messageKey to display as category header
cssClass	CSS class to use for rendering links in the Quicklink menu
ordering	Specifies numerical ordering of quickLinkCategory columns

***** QuickLinkCategory does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **DynamicScope** object represents the set of users who can access its associated QuickLink(s). Four DynamicScopes are predefined in the product: "Everyone", "Manager", "Help Desk", "Self Service". Additionally, beginning in version 7.0, the DynamicScope contains details about the valid target user populations for request-for-others operations initiated by this set of users, as well as the rules that scope objects for role, application, and entitlement requests. **NOTE**: The UI supports DynamicScope creation and management, referring to these objects as Quicklink Populations. DynamicScope objects contain these attributes:

Attribute	Description
identitySelector	An IdentitySelector object which specifies membership criteria – can be a match expression, script, rule, compound filter, or defined population
inclusions	List of references to Identity objects who are explicitly included in the dynamicScope (granted access to the QuickLinks)
exclusions	List of references to Identity objects who are explicitly excluded from the dynamicScope (not granted access to the QuickLinks)
allowAll	Flag indicating that all users in the installation should be included in this dynamicScope (and therefore granted access to the associated QuickLinks)
populationRequestAuthority	Configuration object which calculates the list of users which the dynamicScope user population can make requests on behalf of
roleRequestControl	Rule which controls which roles (Bundle objects) should be presented to this dynamicScope's user population when they are making access requests through LCM
applicationRequestControl	Rule which controls which applications should be presented to this dynamicScope's user population when they are making account-related access requests through LCM; also impacts which entitlements they will be able to request in entitlement requests – only entitlement from authorized applications will be shown
managedAttributeRequestControl	Rule which controls which entitlements (managedAttribute objects) should be presented to this dynamicScope's user population when they are making access requests through LCM

The **PopulationRequestAuthority** object is persisted as a part of the DynamicScope object and can only be accessed in the object model through the DynamicScope, but it has its own set of attributes which define the target population for requests.

Attribute	Description
allowAll	Flag specifying whether the DynamicScope users should be able to make requests for all users in the installation
matchConfig	Configuration which specifies what subset of users in the installation the dynamicScope population can request for; this configuration can further specify options such as matchAll, match by shared attribute (i.e. same department, same job title), or subordinate control (i.e. users can make requests on behalf of those who report to them, directly or indirectly)
ignoreScoping	Flag which can be used to turn off the Scope configuration for quickLink driven requests for installations which have implemented scoping in IdentityIQ

Attributes of the **IdentitySelector** object and their descriptions are described in the *Bundle* section above. In this use case, instead of determining role assignments, they determine membership in the dynamicScope and therefore access to the associated QuickLinks.

Provisioning

Provisioning is a complex process that can involve one or more of the sets of objects described in this section.

- **ProvisioningPlan**: represent the provisioning request and its details
- **ProvisioningProject**: represents a compiled provisioningPlan, including subdivisions of the master plan into smaller provisioningPlan objects that can be processed by a single connector
- **IntegrationConfig**: represents the connector configuration for provisioning tasks conducted by an integration executor (not internal workItem and not read/write connector)
- **ProvisioningRequest**: represents saved, pending requests; used in comparing to new requests to filter duplicates while a request is still pending
- **IdentityRequest**: represent LCM provisioning requests; holds current status and history of requests; exists in release 5.5+ to persist this information for viewing through the UI after the taskResults expiration/deletion
- **ProvisioningTransaction**: represents a submitted provisioning transaction and supports forcing immediate retry of pending events or manual action override for failed requests (introduced in 7.1)

Many of the objects in this section of the model (as noted below) are XML objects and do not contain the attributes discussed in the Objects' Shared Attributes section of this document.

ProvisioningPlan

The ProvisioningPlan object represents a provisioning request involving one or more applications. A provisioning plan may be implemented in a variety of ways: trouble tickets, work items, and automatic provisioning. The ProvisioningResults object tracks the status of the provisioning effort, including the

application used to process the request, the status, and any warnings and errors incurred during the provisioning activity.

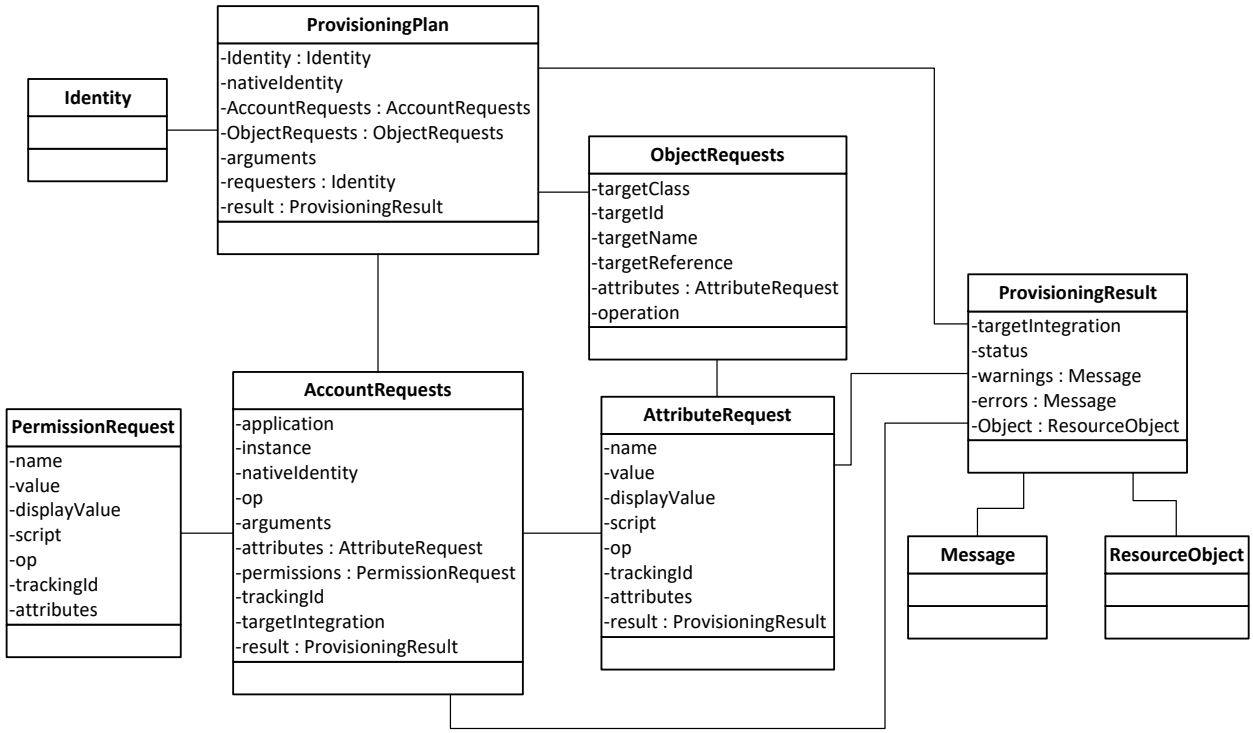


Figure 22: ProvisioningPlan Object and its Attribute Relationships

The table below describes the key attributes on the **ProvisioningPlan** object.

Attribute*	Description
identity	Identity object for identity being acted upon
nativeIdentity	Name of the account in the provisioning system that corresponds to the identity
accountRequests	List of account operations to process
objectRequests	List of object (non-account) operations to process
arguments	Map of extensible attributes (name/value pairs) associated with the plan
requesters	List of Identity object(s) who made the requests (typically only one in list)
targetIntegration	Name of integrationConfig the plan will be sent to
result	ProvisioningResult object (plan evaluation results)

***** ProvisioningPlan does not contain the attributes discussed in the *Objects' Shared Attributes* section.

AccountRequest and **ObjectRequest** objects both contain **AttributeRequest** objects where the details of the attributes for the provisioning action are stored. AccountRequest objects also contain **PermissionRequest** objects to house permission-related provisioning details. The Op attribute on both AccountRequest and ObjectRequest determines what operation will be performed (e.g. Create, Delete, Modify, Enable, Lock, etc.). None of these objects contain the attributes discussed in the Objects' Shared Attributes section.

The **ProvisioningResult** object maintains results from provisioning requests sent to an application's connector or a provisioning system.

Provisioning Project

During the provisioning process, the Provisioning Plan is evaluated and compiled into a ProvisioningProject, with the plan itself often partitioned into multiple smaller ProvisioningPlans that each involves a single target. The ProvisioningProject contains its master ProvisioningPlan (the original master plan that is compiled into multiple partitioned plans) and a list of one or more partitioned plans.

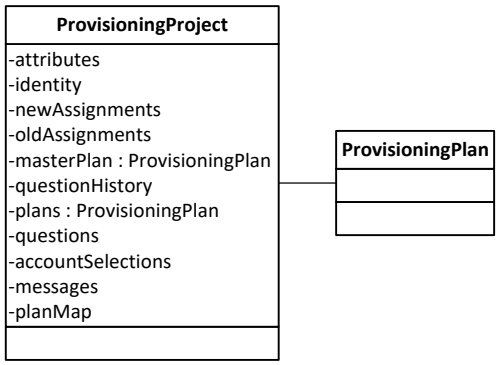


Figure 23: ProvisioningProject Object and its Attribute Relationships

The table below describes the key attributes on the **ProvisioningProject** object.

Attribute*	Description
attributes	Provisioning options in name/value pairs; originally come from the task arguments of the task that is using the provisioner
identity	Name of the associated Identity
newAssignments	New role assignments being applied to the identity
oldAssignments	Old role assignments held by the Identity
masterPlan	Original master plan for the provisioning action (before compiled into multiple smaller plans)
questionHistory	Questions previously submitted and answered (see questions attribute below)
plans	List of ProvisioningPlan objects – one or more partitioned plans generated during compilation
questions	List of missing account attributes that must be specified before the provisioning can proceed
accountSelections	List of ambiguous accounts that must be selected before provisioning can proceed
messages	List of warning and error messages generated during compilation and evaluation
planMap	Map of integration plans keyed by IntegrationConfig name

***** ProvisioningProject does not contain the attributes discussed in the *Objects' Shared Attributes* section.

IntegrationConfig

IntegrationConfig objects define Provisioning Integration Modules (PIMS) which allow IdentityIQ to provision to other Identity Management systems (OIM, SIM, Tivoli, Novell, BMC ESS). They can also be used to configure provisioning through Provisioning Table Integration, Help Desk systems such as Remedy, and JMS/MQ Series Enterprise Message Busses. Applications that use read/write connectors and Connector Gateway use a ProvisioningConfig (defined within the *Application* object itself) rather than an IntegrationConfig.

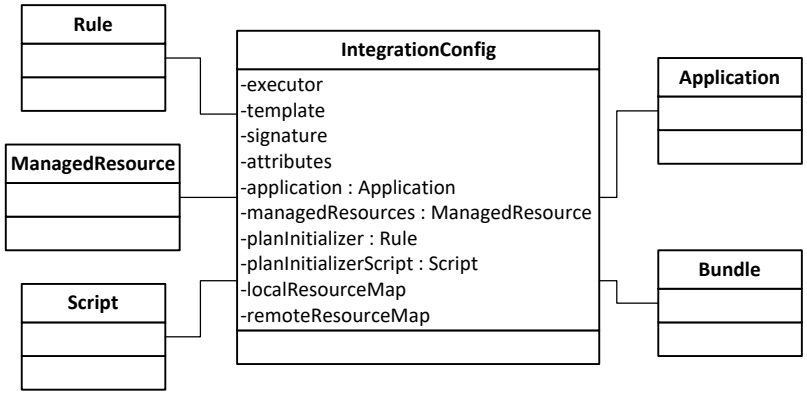


Figure 24: IntegrationConfig Object and its Attribute Relationships

The table below describes the key attributes on the **IntegrationConfig** object.

Attribute	Description
executor	The name of a class implementing the IntegrationExecutor interface
template	Flag indicating whether this is a template configuration
signature	Signature describing configuration attributes (currently not in use; for future development)
attributes	Configuration attribute map (name/value pairs)
application	Optional IdentityIQ application that represents the IDM system; used when IDM system is aggregated into IdentityIQ and needs to represent a Link to the IDM meta-account
managedResources	List of ManagedResource objects (target applications managed by the config)
planInitializer	Rule object that loads the needed data into the ProvisioningPlan (omits any additional data that is not needed – e.g. could load Identity name instead of whole Identity object)
planInitializerScript	Script object used to initialize the ProvisioningPlan (alternate to planInitializer
localResourceMap remoteResourceMap	Runtime caches of managed resource name/value pairs

ProvisioningRequest

During plan evaluation, if the IntegrationConfig specifies that pending requests should be saved (Attribute “saveProvisioningRequests” = “true”), a ProvisioningRequest object is created for the pending request. These can be evaluated during processing of subsequent ProvisioningPlans so that duplicate requests can be removed

from the plans. This is particularly useful for provisioning through ticketing systems, where re-aggregation to reflect provisioned changes may occur days later.

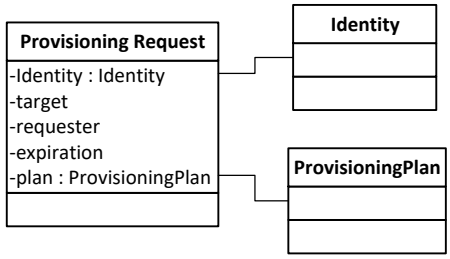


Figure 25: ProvisioningRequest Object and its Attribute Relationships

The table below describes the key attributes on the **ProvisioningRequest** object.

Attribute	Description
identity	Identity for whom the request was made
target	Name of the application or integrationConfig this request was sent through
requester	Name of the Identity or system entity that made this request
expiration	Date the request expires (date the system will stop waiting for the request to be processed by the target system and delete the request, allowing it to be generated again)
provisioningPlan	The provisioning plan that was sent to the connector

IdentityRequest

IdentityRequest objects contain status information about Lifecycle Manager requests once they have been submitted. The current status and historical LCM request data can be viewed through the My Access Reviews page.

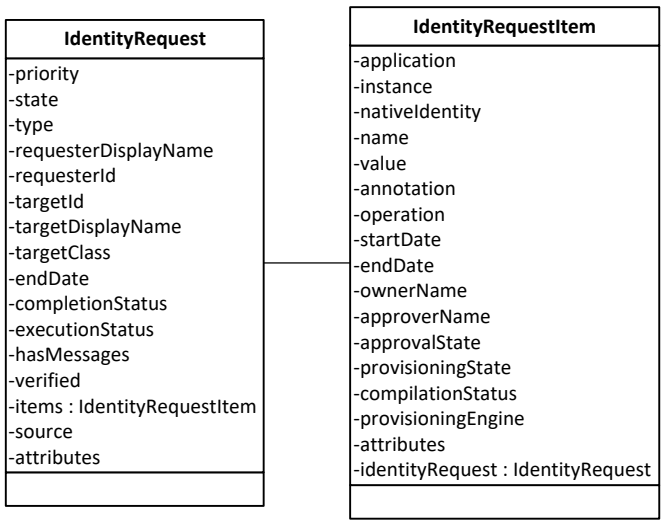


Figure 26: IdentityRequest Object and its Attribute Relationships

The table below describes the key attributes on the **IdentityRequest** object.

Attribute	Description
priority	Priority of the request (set from the Workflow inputs)
state	Current state of the request (e.g. Init, Approve, Provision, Notify, etc.)
type	Type of request (based on workflow that originated the request)
requesterDisplayName	displayName of Identity who made the request
requesterId	ID of the Identity who made the request
targetId	Target of request (typically name and ID of Identity to be created/updated
targetDisplayName	Friendly display name for target
targetClass	Object class of the target (currently always Identity)
endDate	Completion date of the request
completionStatus	Completion status of the request (set after completion) – Success, Failure, Incomplete
executionStatus	Execution status of request (executing, verifying, terminated, complete)
hasMessages	Flag indicating one or more errors/messages in the request's attributes
verified	Date the entire request has been verified by the provisioning scanner
items	List of items for the request (includes things requested and added through expansion during compilation)
source	String version of source object where the request originated
attributes	Xml-based (non-query-able) attribute map (name/value pairs); typically includes approvalSummaries, finalProject, errors, warnings

The table below describes the key attributes on the **IdentityRequestItem** object. Each individual action involved in processing the request is recorded through this object. These are a simplified representation of the ProvisioningPlan objects (e.g. AccountRequest, AttributeRequest).

Attribute	Description
application	Application name for request (null for Identity-level attributes)
instance	Instance identifier for template applications
nativeIdentity	Native identity of the application account link
name	Name of attribute or target of permission
value	Value of attribute or right of permission
annotation	Annotation of permission (undefined for attributes)
operation	String representation of operation to be performed
startDate	Optional time at which requested item will be given (sunrise date)
endDate	Optional time at which requested item will be taken away (sunset date)
ownerName	Name of owner of the request item
approverName	Name of approver of the request item
approvalState	Approval state copied from approvalItems produced by the workflow
provisioningState	Provisioning status of the item (whether it has been successfully applied to the Identity
compilationStatus	Field indicating whether item was part of original request, added during compilation, or filtered during compilation as a duplicate request (other pending requests exist for same action)

provisioningEngine	Name of provisioning engine that handled the request
attributes	Attribute map storing other interesting info about the approval item (name/value pairs)
identityRequest	Back reference to IdentityRequest object

ProvisioningTransaction

ProvisioningTransaction objects contain status information about provisioning operations once they have been submitted. They are created for all types of provisioning processes – requests processed through Lifecycle Manager workflows or custom workflows, automatic role assignment from an identity refresh, certification or policy violation remediation, etc. They track the full set of details about what was requested and its provisioning result status. IdentityIQ supports options to request an immediate retry of any pending ProvisioningTransactions or to force an override of a provisioning failure by converting the request to a manual provisioning request (i.e. a manual work item for provisioning). The system can be configured to record provisioningTransaction objects only for failures, for pending and failed transactions, and for all transactions (including successfully provisioned). This object was introduced in version 7.1.

ProvisioningTransaction objects can contain name/ID references to several other objects in the system, depending on the specific transaction details.

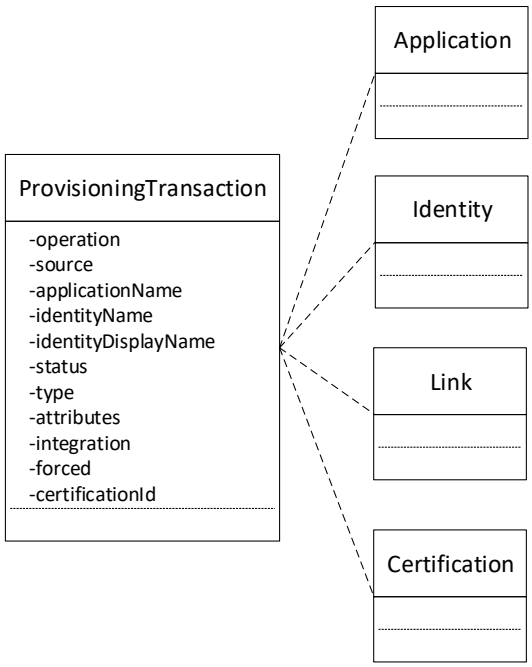


Figure 27: ProvisioningTransaction object and its object relationships

This table describes the key attributes on the **ProvisioningTransaction** object.

Attribute	Description
operation	Provisioning operation, either from the AccountRequest or the ProvisioningPlan, depending on what is reflected in the rest of the record

source	Source enumeration value showing the origin of the transaction (null for custom code)
applicationName	Name of the target application for the provisioning operation
identityName	Name of the target identity
identityDisplayName	Display name for the target identity
nativeIdentity	Account identifier for the application account to which the provisioning operation was directed
accountDisplayName	Display name for the application account
status	Status of the original provisioning request (Success, Pending, Failed)
type	Whether the provisioning transaction was a manual work item transaction (originally) or an automatic provisioning operation to be provisioned through an integration or connector (Manual, Auto)
attributes	Attributes map of attributes which provide the details for the transaction
integration	Name of the target integration (can be a connector) that the transaction was provisioned through
forced	Flag indicating that the transaction was pushed to a manual work item by an administrative user following failure
certificationId	ID of the certification which created the provisioning request, if source = Certification

Risk Scoring

The risk scoring functionality is implemented through a ScoreConfig configuration object, its component objects, and independent scoring objects. The scoring objects are passed the configuration object and the necessary

ScoreDefinition object, along with the object being scored; they store the results in a Scorecard object (also passed in to the scoring object).

Risk Scoring Configuration Objects

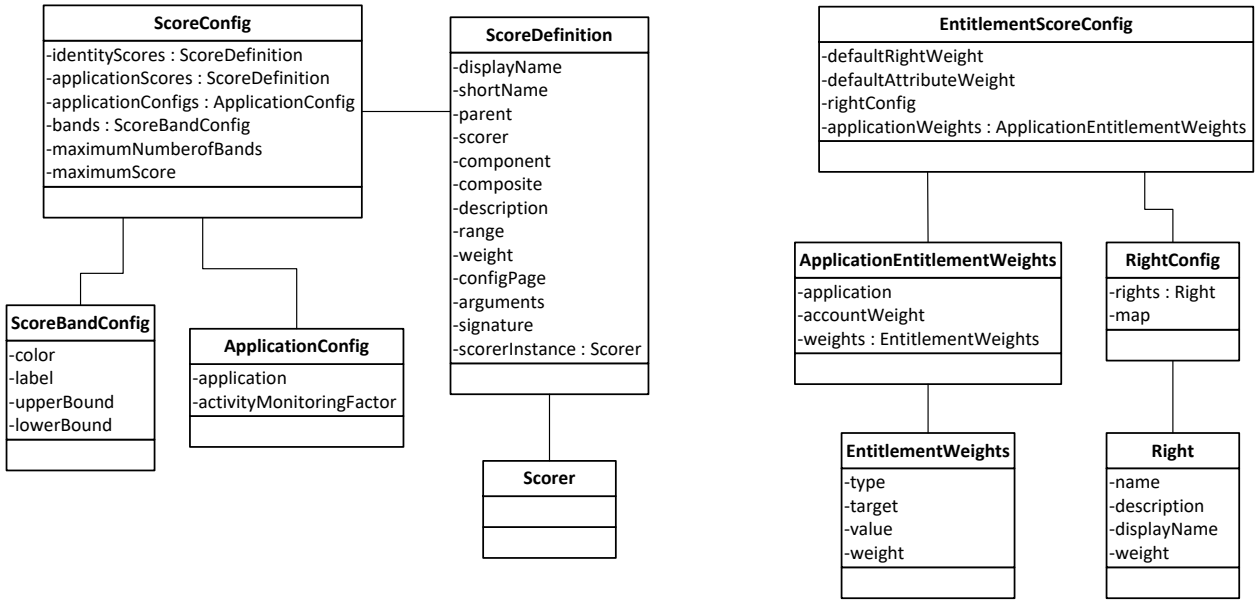


Figure 28: Scoring Configuration Objects and their Attribute Relationships

This table describes the attributes on the **ScoreConfig** object. ScoreConfig represents the global configurations for risk scoring.

Attribute	Description
identityScores	List of ScoreDefinition objects for Identity Scoring
applicationScores	List of ScoreDefinition objects for Application Scoring
applicationConfigs	List of ApplicationConfig objects that provide activity monitoring factors (used in calculating role and entitlement scores for Identity scoring)
bands	List of ScoreBandConfig objects (contain band display configuration and score ranges)
maximumNumberofBands	Total number of bands configured
maximumScore	Maximum score allowable

This table describes the attributes on the **ScoreDefinition** object. It represents the definition of one scoring algorithm.

Attribute	Description
displayName	Name displayed in the UI score configuration pages
shortName	Short name displayed when showing tables of scoreItems (see scoring objects)
parent	Name of parent scoreDefinition (used for baseline and compensated scoreDefinitions where compensated is implemented with same configuration as baseline)
scorer	Fully qualified path name to scorer implementation

component	Flag indicating if this calculates a score that will be used in the composite score; if false and composite is also false, this is a “raw” or “baseline” score
composite	Flag indicating if this calculates a composite, rather than a component or baseline score
description	Verbose description of score definition
range	Upper bound of the score produced by this scorer
weight	Weight this score has relative to peers – percentage of the composite score this score contributes
configPage	URL of the page containing the configuration panel for this score
arguments	Optional arguments to the scorer
signature	Used for custom scores; defines configurable arguments so configuration form can be automatically generated
scorerInstance	Cache resolved scorer instance

This table describes the attributes on the **EntitlementScoreConfig** object. EntitlementScoreConfig can be passed a ScoreConfig and ScoreDefinition in its constructor method. This is an object used during risk scoring as a handle for managing the related scoring components; it is not persisted and accessible through the UI (including Debug pages).

Attribute	Description
defaultRightWeight	Default weight for rights in a permission
defaultAttributeWeight	Default weight for values of an attribute
rightConfig	RightConfig object extracted from the ScoreConfig
applicationWeights	List of ApplicationEntitlementWeight objects (Application-specific weights extracted from the ScoreDefinition’s argument list)

***** EntitlementScoreConfig does not contain the attributes discussed in the *Objects’ Shared Attributes* section.

This table describes the attributes of the **RightConfig** object. This object specifies the default scoring weights to apply to each right for permissions scoring. There is a single top-level RightConfig object for the entire installation, but this can be overridden for any application in its scoreConfig as needed.

Attribute	Description
rights	List of Right objects, which is a combination of the right name and scoring weight
map	Name lookup cache: map of name-value pairs into which the rights are stored internally so the class can serve up a requested right by name

Identity Scoring Objects

The Scorer objects are used to calculate the various score components and the composite score. When calculating a score, they record the score on the Identity’s Scorecard object. EntitlementScorer, PolicyScorer, and IdentityAttributeScorer calculate component score values. CompositeScorer uses data from the Scorecard (written to it by the other Scorers) to calculate the compositeScore value.

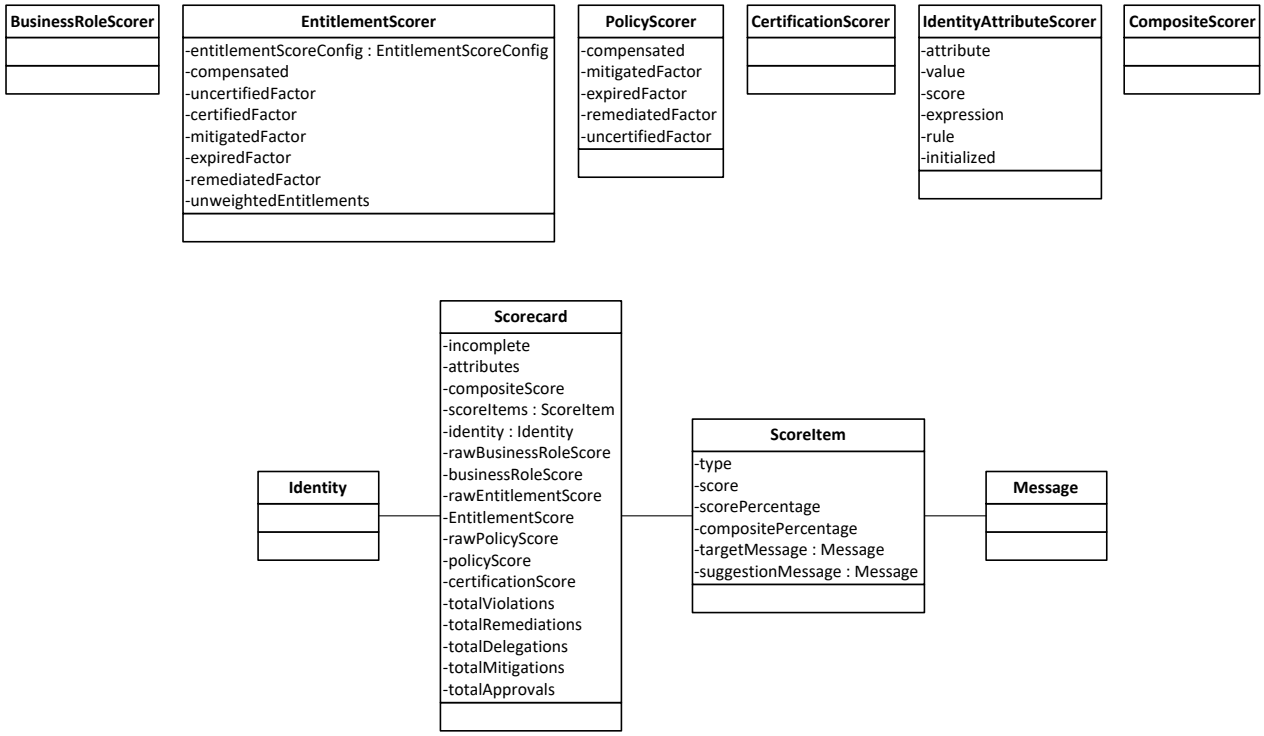


Figure 29: Identity Scoring Objects and their Attribute Relationships

This table describes the attributes on the **Scorecard** object. The last 5 fields (the "total" fields) are used for Identity statistics, not for the Risk Scoring model.

Attribute	Description
incomplete	Flag indicating full score/index could not be calculated for some reason
attributes	Attribute map (name/value pairs) containing scores and statistics
compositeScore	Composite score calculated from others for the Identity
scoreItems	Optional list of items describing the most significant factors that contributed to the composite score
identity	Identity to which these scores and statistics apply
rawBusinessRoleScore	Score calculated based on the roles assigned to or detected for the Identity, not compensated by certification
businessRoleScore	Score calculated based on the roles assigned to or detected for the Identity, compensated by certification
rawEntitlementScore	Score calculated based on the extra entitlements discovered for the Identity, not compensated by certification
entitlementScore	Score calculated based on the extra entitlements discovered for the Identity, compensated by certification
rawPolicyScore	Score calculated based on policy violations for the Identity without compensation
policyScore	Score calculated based on policy violations for the Identity with compensation
certificationScore	Score calculated based on certification history (time since last certification, number of mitigations/remediations during certification)
totalViolations	Count of policy violations for Identity
totalRemediations	Count of remediations for Identity

totalDelegations	Count of certification delegations for Identity
totalMitigations	Count of violation mitigations for Identity
totalApprovals	Count of certification approvals for Identity

This table describes the attributes on the **ScoreItem** object. It lists the most significant factors contributing to the composite score.

Attribute	Description
type	Score type (shortName from ScoreDefinition)
score	Actual value contributed to the score for this type by this item
scorePercentage	Percentage of the score for this type contributed by this item
compositePercentage	Percentage of the composite score contributed by this item (tells how important this item was to the composite score)
targetMessage	Brief description of thing to which this item is most closely associated (e.g. for Roles, name of the Bundle; for Entitlements, attribute/value or right/target pair)
suggestionMessage	Optional suggestion on what user could do to reduce this score

Application Scoring Objects

As with Identity Scoring, the application Scorer objects are used to calculate the various score components and the composite score, saving them to the ApplicationScorecard object. LinkAttributeScorer, ViolatorAccountScorer, RiskyAccountScorer, and DormantAccountScorer calculate component scores. CompositeScorer uses data from the ApplicationScorecard to calculate the compositeScore value.

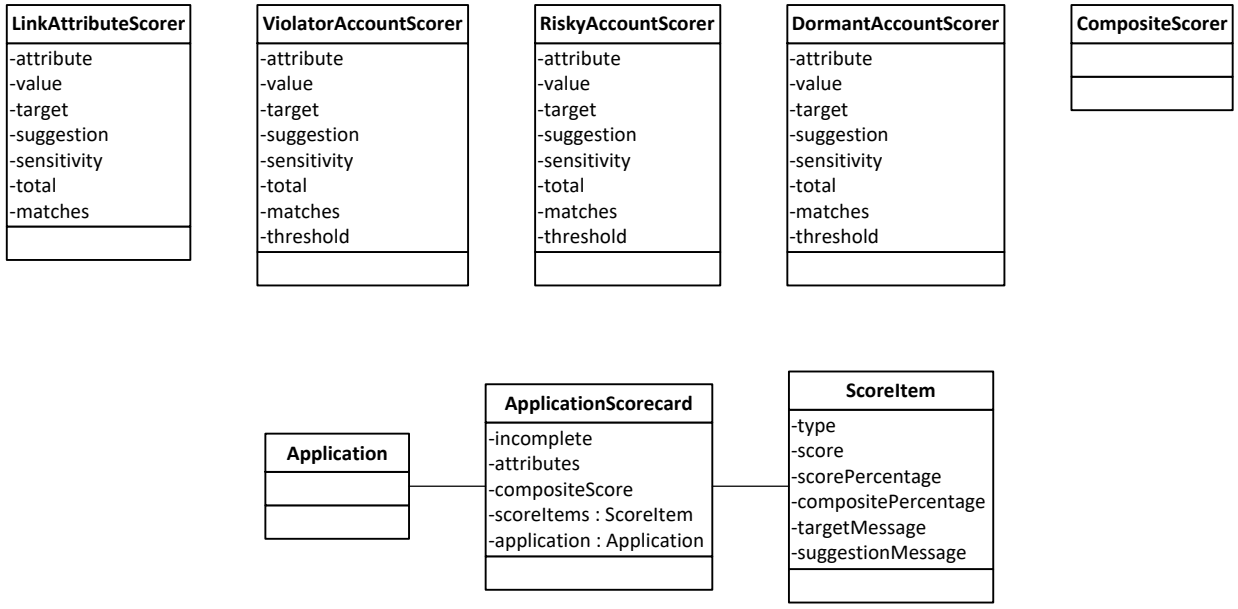


Figure 30: Application Scoring Objects and their Attribute Relationships

This table describes the attributes on the **ApplicationScorecard** object.

Attribute	Description
incomplete	Flag indicating full score/index could not be calculated for some reason

attributes	Attribute map (name/value pairs) containing scores and statistics (this is where the component scorers write their score results)
compositeScore	Composite score calculated from others for the application
scoreItems	Optional list of items (ScoreItem objects) describing the most significant factors that contributed to the composite score; this is the same ScoreItem object type described above under the *Identity Scoring Objects*
application	The Application object to which this Scorecard relates

Unstructured Target Data Collection

IdentityIQ supports gathering of data on unstructured targets – representations of access rights which are not as neatly organized and easily collected as entitlements or permissions on accounts. Target data can be attached to a managedAttribute (e.g. an account group) or to a specific identity.

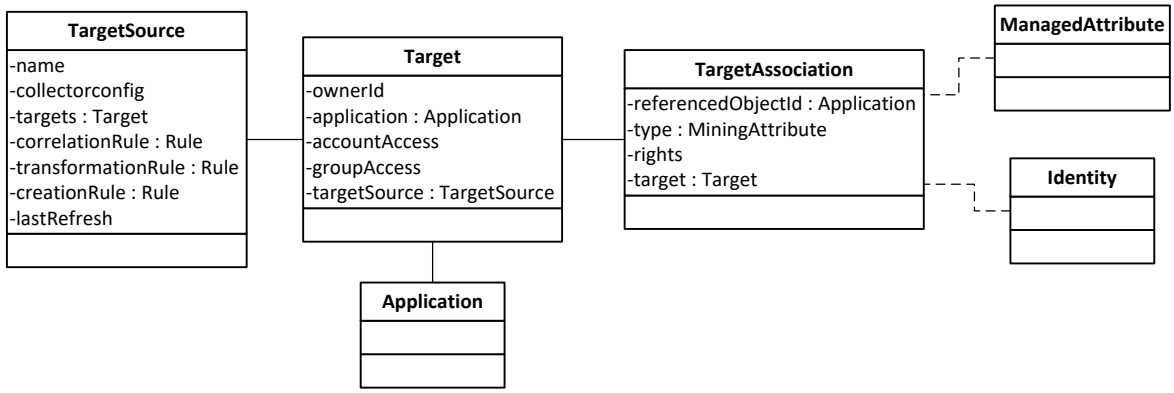


Figure 31: Unstructured Target Data Collection Objects and their Attribute Relationships

The **TargetSource** object represents the configuration for the target collection process. It specifies the collector class and configuration options to use, the rules to use for creating the target object, and the rule to use for matching the Target to an identity or managedAttribute.

Attribute*	Description
name	Name of this TargetSource object
collector	class name that implements fetching data from this datasource
config	Attributes map of the configuration that is supported for this datasource
targets	List of the names of the associated unstructured targets (Target objects)
correlationRule	Rule object that can be invoked to
transformationRule	Rule object that can be used to normalize the collected data
creationRule	Rule object that can used to set anything on the target before it gets persisted (final rule hook before saving)
lastRefresh	Timestamp for the last time activity data was refreshed from the underlying datasource

***** TargetSource does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **Target** object represents the unstructured data access held by the owner (the identity or group) on a given application.

Attribute	Description
ownerId	Native owner ID that will be correlated to an Identity during the target aggregation process
application	Application to which this target applies
accountAccess	Transient list of AccessMappings which includes mapping of rights to native user IDs
groupAccess	Transient list of AccessMappings which includes mapping of rights to native group IDs
targetSource	TargetSource object which aggregated this target

The **TargetAssociation** object represents the connection between the Target object and the owner (the Identity or ManagedAttribute object).

Attribute	Description
referencedObjectId	GUID of the managedAttribute or the Identity referenced by this association
type	Type of object referenced: either ManagedAttribute or Account
rights	CSV list of Rights assigned to the associated object
target	Reference to the Target object

Activity Monitoring

Activity monitoring can be configured in IdentityIQ per Identity or per Role; for each entity, activity monitoring can be specific to individual applications or can be turned on for all applications. The objects in this section define how activity information is gathered from an application. Activity monitoring data is commonly referred to as the "event log" and contains a record of what the account did on the application. It reflects activity that occurs in the native applications (e.g. read, start, stop, authenticate, login, delete, create, etc.)

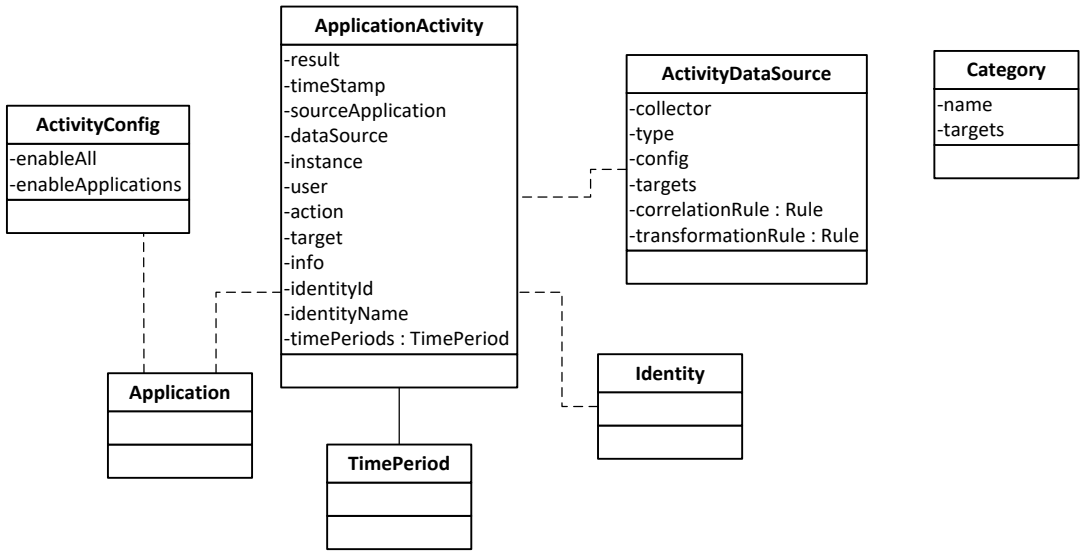


Figure 32: Activity Monitoring Objects

The **ActivityConfig** object represents the configuration for activity monitoring. ActivityConfigs are specific to an Identity or a Role (and are held within the associated Identity or Role).

Attribute*	Description
enableAll	Flag indicating all applications should be monitored
enabledApplications	Set of application IDs that should be tracking activity data

***** ActivityConfig does not contain the attributes discussed in the *Objects' Shared Attributes* section.

This table describes the attributes on the **ApplicationActivity** object, which represents the activities in the event log.

Attribute	Description
result	Stores status of native event (success or failure)
timeStamp	Date-time of event (comes from native event)
sourceApplication	Application name where event occurred
dataSource	String name of the data source that produced the activity (especially important when an application has more than one data source)
instance	Instance identifier when the sourceApplication represents multiple instances
user	Native Identity engaging in the activity
action	Action that took place (enumerated list; e.g. read, delete, accept, start, login, etc.)
target	Actual target of the action (e.g. EmployeeDataTable)
info	Any additional info coming from the native event
identityID	Hibernate ID of the Identity object
identityName	Name of the Identity object
timePeriods	List of time periods in which this activity occurred

This table describes the attributes on the **ActivityDataSource** object, which defines the source for gathering activity data (the collector) and the logic for transforming the collected data into a normalized format, as well as logic for matching the activity to a target identity in the system.

Attribute	Description
collector	Class name for the class which retrieves data from the data source
type	
config	Attributes map of configuration data for the datasource
targets	List of activity targets (string values)
correlationRule	Rule object which can be used to identify the Identity performing the recorded activity
transformationRule	Rule object which normalized the collected data so it can be recorded appropriately in IdentityIQ

The **Category** objects are used to categorize activity data for an application. This categorization can be used in activity policy definitions to group types of activity into a policy together. Category objects are defined through the IdentityIQ user interface on the Define -> Activity Target Categories page and are composed of these attributes:

Attribute	Description
name	String name of the category
targets	List of String names of the targets (defined in the Application's Activity Data Source definition) which have been linked to the category

Alerts

Alerts, introduced in version 7.1, are similar to activities in some ways but have been specifically developed to respond to alerts provided by an unstructured data governance solution like SecurityIQ. They allow IdentityIQ operations, such as workflows or email notifications, to be triggered by those alerts. There are two groups of objects involved in the alert process, with three central objects: the AlertDefinition, Alert, and AlertAction objects.

Alerts are the objects which are aggregated from the data access governance tool (e.g. SecurityIQ). AlertDefinitions are the configuration objects for what IdentityIQ should do in response to a given type of Alert. The Alert Processing Task (a new task in IdentityIQ 7.1) tries to match the Alerts to a configured AlertDefinition. When it does, the action configured in the AlertDefinition is executed and creates an AlertAction; that AlertAction gets stored on the Alert to show what has occurred.

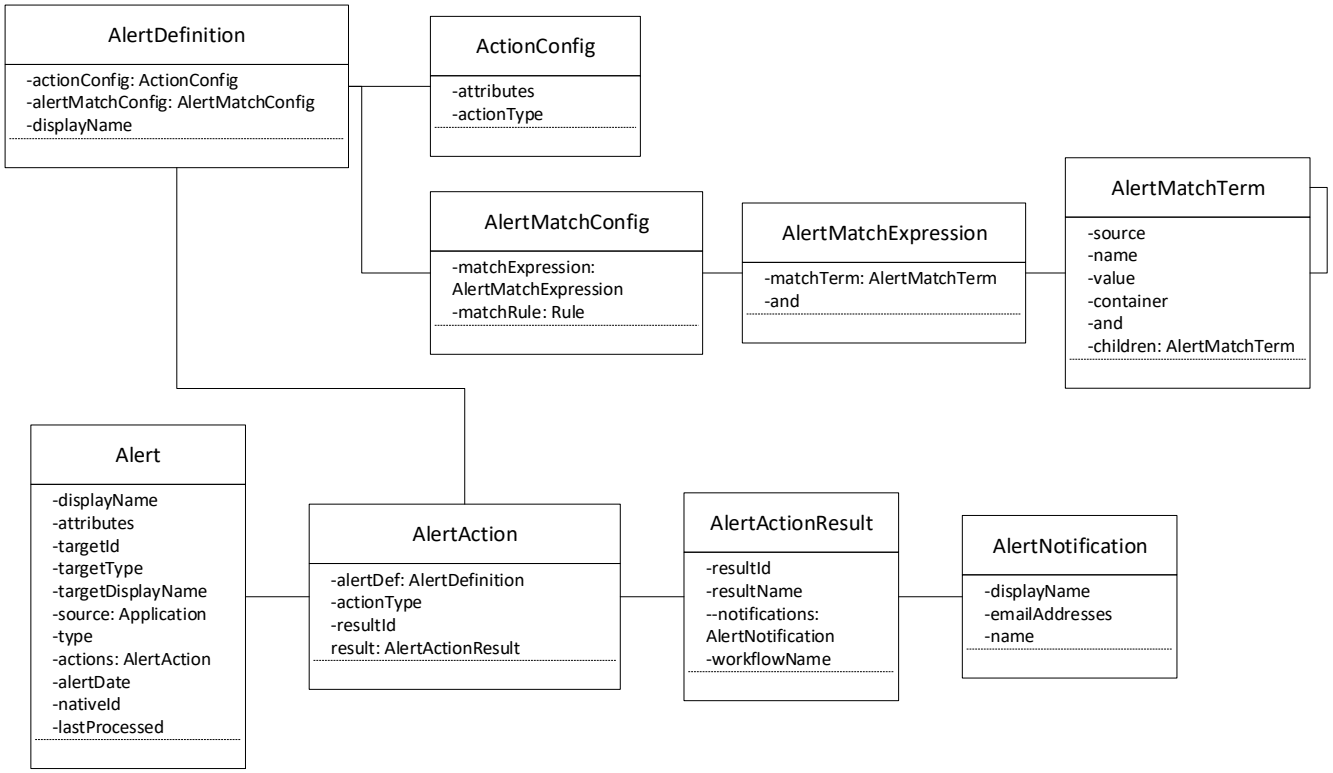


Figure 33: Alert and Alert Definition objects and their attribute relationships

The **AlertDefinition** object represents what IdentityIQ should do in response to a specific type of alert being discovered, as well as the criteria for triggering the action.

actionConfig	ActionConfig object which defines what the system should do when the alert type is discovered
alertMatchConfig	AlertMatchConfig object which defines the criteria for taking the configured action
displayName	Name of the AlertDefinition to show in the UI

The **ActionConfig** object represents the details of the action to take: both the type, which is one of 3 options, and the details to use in processing that action type.

Attribute	Description
attributes	Attributes map to store information for the action (e.g. workflow name and arguments, email template and recipient, certification event to fire)
actionType	One of the enumeration of alert definition types: “Workflow, Certification, Notification”; identifies whether the alert will run a workflow, generate a certification, or send an email

The **AlertMatchConfig** object defines the details of the matching criteria, which is either configured through a set of match terms or through a rule.

Attribute	Description
matchExpression	AlertMatchExpression object which defines the condition for triggering the alert
matchRule	Rule object which defines the condition for triggering the alert (alert triggered if rule returns true); alternative to matchExpression

The **AlertMatchExpression** object is a collection of match terms which are used to define the triggering criteria.

Attribute	Description
matchTerms	List of AlertMatchTerm objects which should trigger the action
and	Boolean indicating whether matchTerms should be related through logical AND or OR

The **AlertMatchTerm** object is a single match criterion (or possibly a nested set of match criteria within the larger MatchExpression) which helps define the match expression.

Attribute	Description
source	Source application of the alert (optional)
name	Name of the Alert attribute to examine
value	Alert attribute value that should trigger the action
container	Boolean indicating this is a container of terms (i.e. children list is populated)
and	Boolean indicating whether children should be related through logical AND or OR
children	List of AlertMatchTerm objects when grouping a set of terms together

The **Alert** object is the alert aggregated from SecurityIQ or a similar system which can trigger the configured action when it meets the alert definition conditions.

Attribute	Description
displayName	
attributes	Attributes map populated for the alert type
targetId	ID of the related SailPointObject
targetType	Type of the SailPointObject (simple class name)
targetDisplayName	DisplayName of the related SailPointObject
source	Source application where the alert originated
type	Type of the alert
actions	List of AlertActions objects
alertDate	Date the alert was created in the source system
nativeId	ID for the alert in the source system
lastProcessed	Date the alert was aggregated into IdentityIQ

The **AlertAction** object represents the action taken when the Alert meets the AlertDefinition requirements. It ties the Alert to the AlertDefinition and the action result.

Attribute	Description
alertDef	AlertDefinition
actionType	One of the enumeration of alert definition types: “Workflow, Certification, Notification”
resultId	ID of the result object generated
Result	AlertActionResult object

The **AlertActionResult** object reports the results of the alert action (the workflow run, the emails sent, etc.

Attribute	Description
resultId	taskResultID or certificationID (depending on what the action was)
resultName	Name of the result object corresponding to the resultId
Notifications	List of AlertNotification objects
workflowName	Name of workflow executed

The **AlertNotification** object contains the name of the email message sent and the recipients of the message when the alert action includes (or is) a notification.

Attribute	Description
displayName	Display name of identity/group notified
emailAddresses	List of email addresses to whom the email message was sent
name	Name of identity/group notified

Batch Requests

Version 6.0 introduced the batch request functionality, through which batch updates can be made to Identities by loading data from a file. Batch requests can do any of these:

- Create Identity
- Modify Identity
- Create Account
- Delete Account
- Enable / Disable Account
- Unlock Account
- Add Role
- Remove Role
- Add Entitlement
- Remove Entitlement
- Change Password

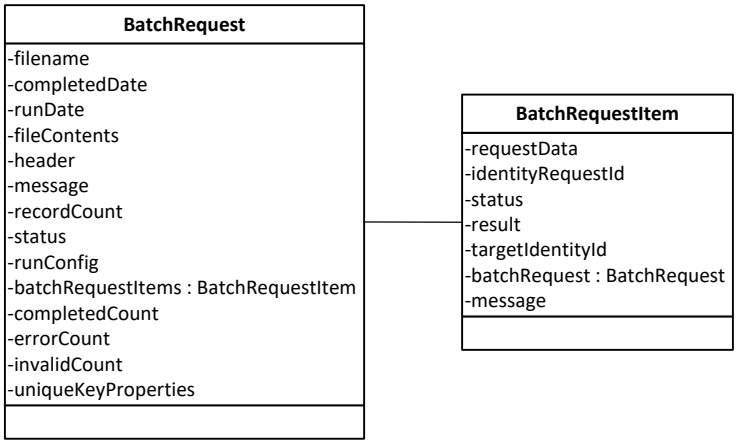


Figure 34: BatchRequest Objects and their Attribute Relationships

The **BatchRequest** object represents the batch request file and contains these attributes:

Attribute	Description
fileName	Full path name of the file containing the batch requests
completedDate	Date on which the batch request was completed
runDate	Date on which the batch request was executed (started)
fileContents	A string representation of the full contents of the file
header	A string representation of the header record in the file (first row)
message	String containing any error messages about the file or its processing
recordCount	Total count of non-header records in the file
status	Status of processing (e.g. executed, invalid, running, scheduled, terminated, approval)
runConfig	An attribute map of the batch request's settings, as specified on the batch request UI page
batchRequestItems	List of batchRequestItem objects in the file
completedCount errorCount invalidCount	Counts of records that were successfully processed or errors/invalid records encountered
uniqueKeyProperties	A map of request properties that uniquely identifies this request; if not otherwise specified, the filename and created date are used as the key. This is a system-used field and can be left alone to use the defaults.

The **BatchRequestItem** object represents an individual request item within the request file and contains these attributes:

Attribute	Description
requestData	String representation of the request line item from the batch file
identityRequestId	System-assigned ID for the IdentityRequest created in response to a batch request item, when the IdentityRequest generation option is selected

status	Running status of the request item (e.g. Running, Finished, Terminated, Invalid)
result	End result of the item request (e.g. Success, Failed, Approval, Skipped, ManualWorkItem, PolicyViolation, ProvisioningForm, AccountSelection)
targetIdentityId	Id of the Identity on whom the action was taken (not set for create requests)
batchRequest	Reference back to the batchRequest to which this item belongs
message	String message containing the reason the item was not processed successfully (if applicable)

Lifecycle Events and Certification Events

Lifecycle events and certification events are defined using an **IdentityTrigger** object. This object specifies what identity change should cause the process to run, as well as what action should be taken when the trigger condition is met.

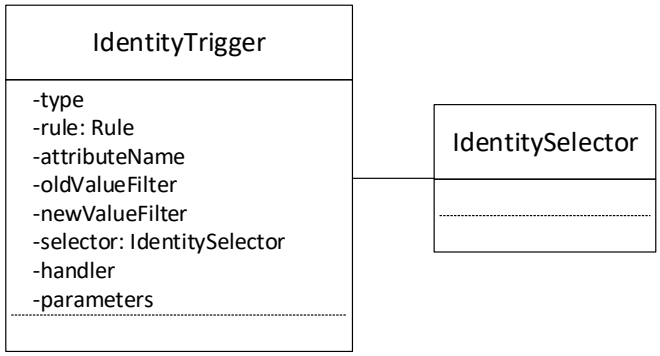


Figure 35: BatchRequest Objects and their Attribute Relationships

The **IdentityTrigger** object contains these attributes:

Attribute	Description
type	Type of event (enumeration: Create, Delete, AttributeChange, Rule, ManagerTransfer, NativeChange)
rule	Rule to run if the trigger type is Rule
attributeName	Name of the attribute to watch for a change in value if the trigger type is AttributeChange
oldValueFilter	For AttributeChange triggers; only trigger the event if the specified value was the previous value for the attribute
newValueFilter	For AttributeChange triggers; only trigger the event if the specified value is the new value for the attribute
selector	IdentitySelector which specifies constraints on the set of identities to which this trigger applies (e.g. can run for only identities in the Accounting department or only identities which have an account on application X, etc.)
handler	Fully-qualified name of the IdentityTriggerHandler class – determines whether this is a certification event trigger or a lifecycle event workflow trigger
parameters	Attributes map of parameters to pass to the IdentityTriggerHandler; includes the name of the workflow to run or the certification definition to execute

Auditing and Reporting

Auditing

The AuditConfig object and its related objects define the configuration for what goes in the audit log, while the AuditEvent object represents the log entry itself.

The AuditEvent object's action attribute usually specifies the name of a configured AuditAction, connecting the configuration details to the audited event.

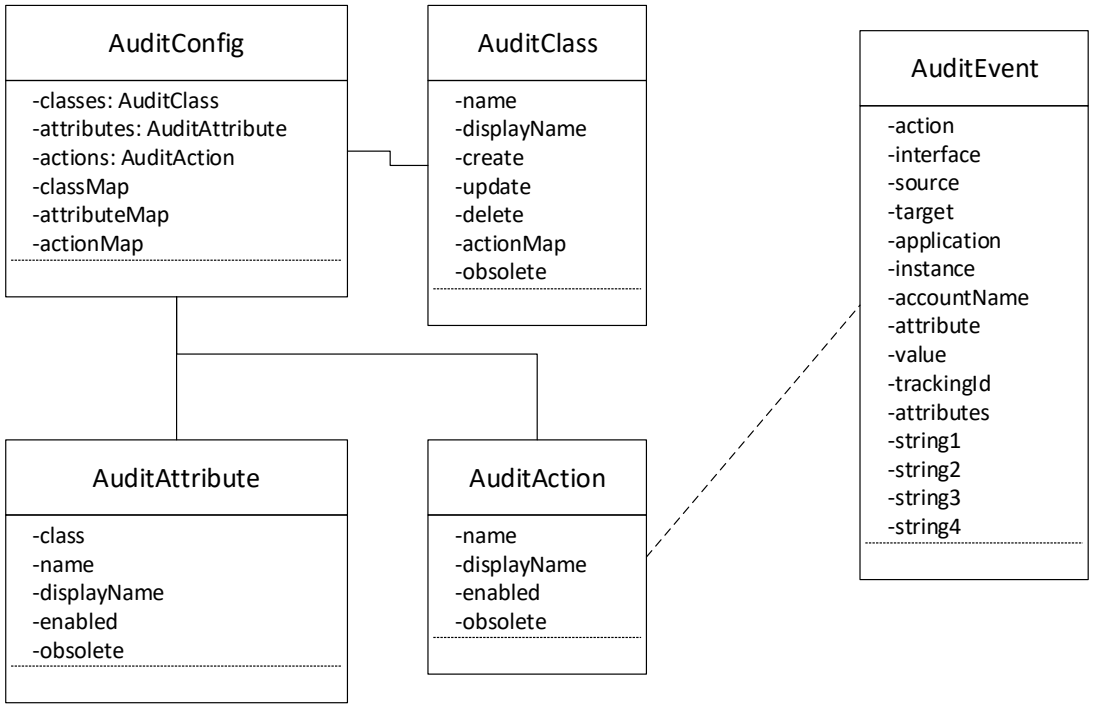


Figure 36: Auditing-related Objects and their Attribute Relationships

The table below describes the key attributes on the **AuditConfig** object.

Attribute	Description
classes	List of AuditClass objects (class configurations)
attributes	List of AuditAttribute objects (attribute configurations)
actions	List of AuditAction objects (general actions)
classMap	Runtime map (class name/value pairs) created from Classes to speed lookup
attributeMap	Runtime map (attribute name/value pairs) created from Attributes to speed lookup
actionMap	Runtime map (action name/value pairs) create from Actions to speed lookup

This table lists attributes of the **AuditAction** object and their descriptions.

Attribute*	Description

name	Name of action for which audit records should be written
displayName	Display name of attribute for which audit records should be written
enabled	Flag saying this audit requirement is enabled

***** AuditAction does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The table below describes the key attributes on the **AuditClass** object.

Attribute*	Description
name	Name of class for which audit records should be written
displayName	Display name of class for which audit records should be written (e.g. Role is the display name for the class Bundle)
create	Flag indicating whether to audit creation events
update	Flag indicating whether to audit update events
delete	Flag indicating whether to audit delete events
actionMap	Runtime map of action flags (create/update/delete, True)

***** AuditClass does not contain the attributes discussed in the *Objects' Shared Attributes* section.

This table lists attributes of the **AuditAttribute** object and their descriptions. This represents the actual audit log entries.

Attribute*	Description
class	Class of object to which the auditable attribute belongs (currently only identity attributes)
name	Name of attribute for which audit records should be written
displayName	Display name of attribute for which audit records should be written
enabled	Flag indicating this audit requirement is enabled

***** AuditAttribute does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The table below lists attributes of the **AuditEvent** object and their descriptions.

Attribute	Description
action	String indicating what happened; usually matches an AuditAction from the AuditConfig
interface	Source object (sailpoint.object.source object) from which the event was initiated
source	The Identity name taking the action (can also be "system" for system events or can be blank for anonymous events)
target	Name of item on which the action was performed (usually a SailPoint object – Identity, Link, Bundle, etc.)
application	Application where action occurred (IdentityIQ or an application name)
instance	Application instance where action occurred (IdentityIQ or an application name)
accountName	Native account ID where the change was made
attribute	Name of the attribute that was changed
value	Value that was changed
trackingID	Tracking ID used to correlate events in groups (WorkflowCase ID is stored here to track all actions from a given workflow)
attributes	Map of event attributes (name/value pairs)

string1 – string 4	String values saved for the audit event (e.g. with attribute events, string1 = attribute name, string2 = operation (set, add, remove), and string3=CSV of the values); varies by event type

Reporting

IdentityIQ 6.0 introduced a reporting infrastructure which makes custom report development much easier to do. Reports are specified and managed as TaskDefinition objects and the entire structure of the report is represented within a LiveReport object inside that TaskDefinition.

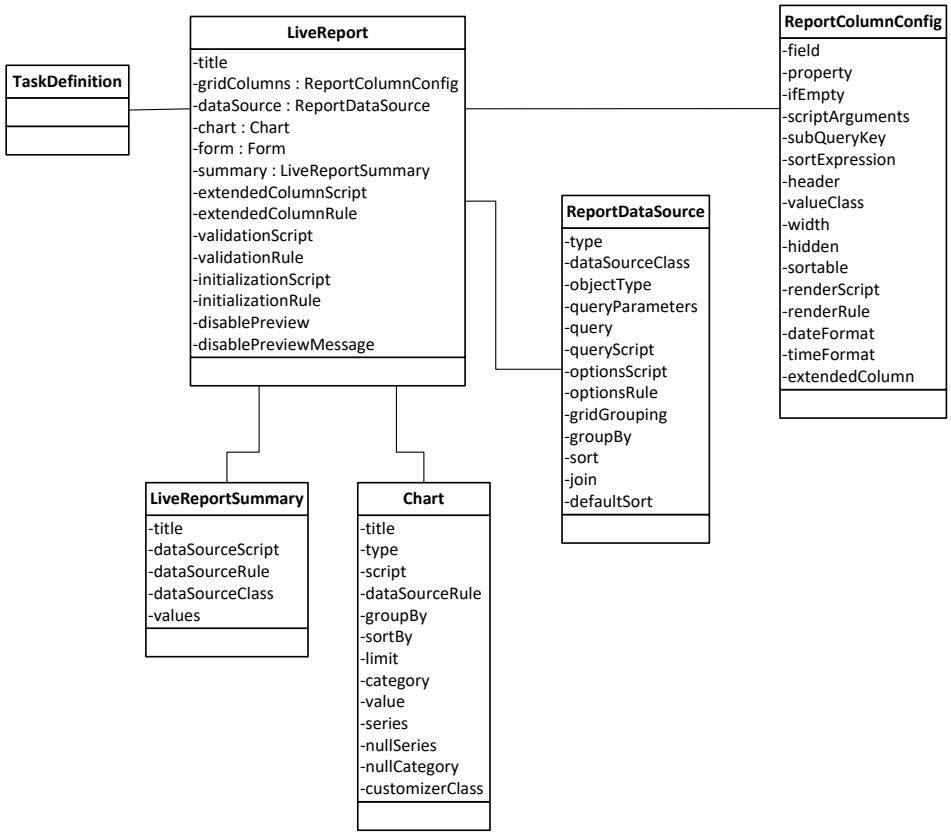


Figure 37: Reporting Objects and their Attribute Relationships

The **LiveReport** object is the master object for the 6.0+ reporting architecture. Its attributes and their usages are shown in this table.

Attribute*	Description
title	Report title (prints at top of report)
gridColumns	List of ReportColumnConfig objects, representing columns on the report
dataSource	ReportDataSource object that defines the source of the report's data
chart	Chart object that defines the line graph, bar graph, or pie chart displayed in the report's summary section
form	Form object used for gathering input parameters for the report from a user in the reporting user interface

summary	LiveReportSummary object defining the summary table of information in the report's summary section
extendedColumnScript	Script used to dynamically add to the report output columns that are not part of the base report definition (in the reportColumnConfigs)
extendedColumnRule	Rule encapsulated version of an extendedColumnScript (holds a rule reference)
validationScript	Script that validates fields on the form when the user saves the report for execution
validationRule	Rule encapsulated version of a validationScript (holds a rule reference)
initializationScript	Script that runs when the report is first loaded; can be used to build form pages dynamically (e.g. populating the form with all standard and extended identity attributes as filter options)
initializationRule	Rule encapsulated version of an initializationScript (holds a rule reference)
disablePreview	Flag that indicates that preview mode is disabled for this report
disablePreviewMessage	Message to user explaining why preview mode is disabled for the report

***** LiveReport does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **ReportDataSource** object, which defines the source of the data to display in the main body of the report, contains the following attributes:

Attribute*	Description
type	Type of datasource for the report body: Filter, HQL or Java
dataSourceClass	Name of the class that serves as the report data source when the datasource type is Java
objectType	Primary object against which the query will be executed when the datasource type is Filter
queryParameters	This is the most frequently used query specifier for a Filter datasource; used to specify filter criteria for the datasource (one criterion per queryParameter, with multiple queryParameters permitted per datasource)
query	Query criteria for Filter datasource specified as a hard-coded (no variable substitution) filter string
queryScript	A beanshell script that creates the filter string for a Filter datasource
optionsScript	A beanshell script to modify an existing filter string for a Filter datasource
optionsRule	A reference to a rule that modifies an existing filter string for a Filter datasource (alternative to optionsScript)
gridGrouping	Specifies the list of columns by which the report results grid rows should be grouped
groupBy	Specifies the list of columns to be used in the SQL group by clause
sort	Specifies the sort order for records returned from the Filter datasource
join	Specifies the join parameters when a second object needs to be joined to the primary object to execute the query for the datasource
defaultSort	Default sort order to apply to report records

***** ReportDataSource does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **ReportColumnConfig** object specifies how a report column will be formatted in the grid display. It is similar to a ColumnConfig for that controls the display of columns in UI grids.

Attribute*	Description
field	Name to assign to this column for programmatic use (e.g. sorting, grouping)
property	Object property to display as the value in this column

ifEmpty	Alternative property value to display if the column's property is empty/null
scriptArguments	CSV list of other column properties to send to the renderScript/renderRule on this column
subQueryKey	When joining to a secondary object that has a one-to-many relationship with the primary object, this value indicates the column on that secondary object to use to connect the two objects (e.g. Account Group Membership Access Review Live Report retrieves "tag" names through this ReportColumnConfig, specifying a subquery on the tags table through its id field: <ReportColumnConfig field="tags" header="rept_cert_col_tags" property="parent.certification.tags.name" subQueryKey="id" width="110"/>)
sortExpression	CSV list of properties on which to base the sort when this columns is used as a sort column (required when the column is a calculated field – i.e. using a renderScript -- that has been marked as sortable)
header	Title to display as column header
valueClass	Class/datatype of the column property; assumed to be string if not specified
width	Forced width for the column in the display (columns will be given equal space across the grid if no widths are specified)
hidden	Flag indicating whether the column should be hidden from the display when the report contents are rendered
sortable	Flag indicating the report data can be sorted by this column
renderScript	Allows processing (including subqueries) based on the value of the column property to display different information in the grid column
renderRule	Encapsulation of a renderScript in a re-usable rule (reference to the rule)
dateFormat	Date format string (SHORT, MEDIUM, LONG, per java.text.DateFormat)
timeFormat	Time format string (SHORT, MEDIUM, LONG, per java.text.DateFormat)
extendedColumn	True if the column was not part of the base report but was added by the report's extended column definition

***** ReportColumnConfig does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **LiveReportSummary** object represents the summary table displayed in the report's summary section and contains these attributes:

Attribute*	Description
title	Text to display above the summary table as its title
dataSourceScript	Beanshell script containing the logic for retrieving the data for the summary section
dataSourceRule	Reference to a Rule which contains the beanshell logic for retrieving the data for the summary section (alternative to dataSourceScript)
dataSourceClass	Name of the class that serves as the summary data source (alternative to dataSourceScript and dataSourceRule)
values	A list of dataSourceValue objects (made up of name, label, and value) to display in the summary

***** LiveReportSummary does not contain the attributes discussed in the *Objects' Shared Attributes* section.

The **Chart** object represents the graph (bar graph, line graph, or pie chart) that can be displayed in the report's summary section. Its attributes are listed in this table:

Attribute*	Description

title	Text to display above the chart as its title
type	Type of chart (bar, line, pie)
script	Used to define a datasource for the chart other than the report detail data; script contains a <Source> element with beanshell content
dataSourceRule	Used to define a datasource for the chart other than the report details data; contains a reference to a rule where the beanshell has been encapsulated
groupBy	String value of column name (or CSV list of column names) to group data by for graphing counts
sortBy	List of sort columns for data; seldom used since groupBy can specify multiple fields in a CSV list
limit	Limits the number of records examined for the graph; seldom used, unless a similar limit was imposed on the report detail data, because the graph should generally represent all of the data in the report detail
category	Defines the X axis in line or column graphs; defines the separate sections of a pie graph
value	Defines the Y axis in line or column graphs; defines the portion (fraction) of the pie that belongs to each section in a pie graph; often a count
series	Defines the separate columns or lines on line or column graphs; ignored for pie charts
nullSeries	Label to display if the series value is null (e.g. a null certification action status means "Open" so that should be the label for the group of certs whose action.status is null)
nullCategory	Label to display for data group when the category value is null
customizerClass	Only applies when rendering jasper charts; left null for most charts

***** Chart does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Legacy Reporting Infrastructure

Prior to version 6.0, reports were defined through individual java classes which supported each report taskDefinition and generated their own JasperReports output. A few of the existing reports have not been converted to the new report style, and the older infrastructure still exists and functions for installations which wrote their own custom reports using that structure. The two objects JasperTemplate and JasperResult are part of that older reporting infrastructure. Most modern installations will only use these objects if they have legacy reports they have migrated forward from an older version of IdentityIQ.

JasperTemplate objects are the definition of a JasperTemplate – the XML format, pre-compilation representation of the report.

Attribute	Description
reportXml	XML string representation of the design of the report; stored in the repository and possibly edited through the web as needed
report	JasperReport object; compiled version of the report which is not stored and only created when requested
design	JasperDesign object representing the report design; not stored in the repository but loaded based on the stored reportXml when needed

Report Output

JasperResult objects are the results of a report execution, whether the report is defined from a JasperTemplate or from a LiveReport task definition. In the old infrastructure, this is a wrapper around the JasperPrint object,

which is the result of filling the report with data. In the LiveReport infrastructure, this includes an attributes map which specifies the report configuration and the row count on the report, plus a Files list which specifies the file names and IDs of the persisted pdf and csv files (stored in the database until the user requests to download them).

Attribute	Description
JasperPrint	Object containing the report data
printXml	XML representation of the JasperPrint object for storage in the repository
handlerId	ID to help find the associated pages
pageCount	Total number of papers generated as part of this report (both internally and externally stored)
externalPageCount	Number of pages externally stored in JasperPageBucket objects
pagesPerBucket	Number of pages per bucket for the report; can be configured per report instance
pageHandler	Copy of the handler cached on the object so the state can be reused when serving up pages
files	List of PersistedFile objects for this report
attributes	Attributes map of data related to this result object

Syslog Event

Errors written to the system logs are now (versions 6.0+) also captured in the database for viewing through the IdentityIQ UI on the Advanced Analytics Syslog Search page.

These are standalone objects. No diagram is provided because there are no connections to illustrate.

The attributes of the **SyslogEvent** object and their usages are shown in this table:

Attribute	Description
quickKey	Unique ID for the event shown in the UI when the error occurs; used to look up the error details
eventLevel	Severity level of the event
className	Name of the class that created the event
lineNumber	Line number at which the event occurred
message	Error message for the event
username	Name of the user for whom the event occurred (“system” if unavailable)
server	Server on which the event occurred
thread	Name of the thread in which the event occurred
stacktrace	Full stacktrace of the event

Process Scheduling Objects

Task Scheduling

Tasks are Quartz-scheduled activities that run in the background in IdentityIQ. They are managed through the **TaskSchedule** and **TaskDefinition** objects, their results are tracked in the **TaskResult** object, and they are executed through the implementation of a **TaskExecutor** object (or a custom object that extends the **TaskExecutor**). (In practice, task executors extend the **AbstractTaskExecutor**, which is itself an extension of **TaskExecutor** that includes a Monitor object that helps the task executors write task progress.)

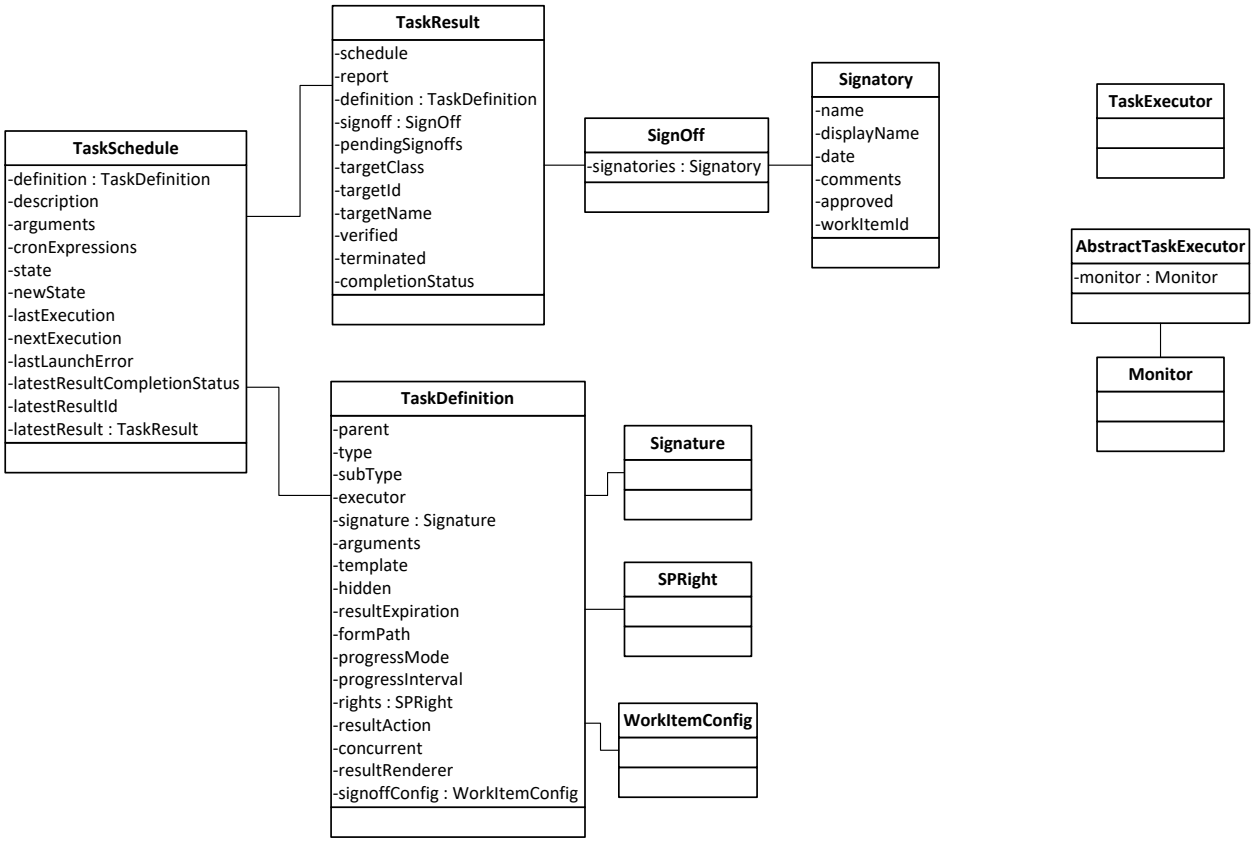


Figure 38: Task-related Objects and their Attribute Relationships

The table below describes the key attributes on the **TaskSchedule** object.

Attribute	Description
definition	TaskDefinition object for the scheduled task
description	Optional lengthy description for the task
arguments	arguments to be passed to the JobExecutor; used for anything that can be dynamically configured and displayed in the scheduling UI (map of name/value pairs)
cronExpressions	List of Scheduling cron expressions
state	Current state of the job (read-only)

newState	Field set to request that the job enter a new state
lastExecution	Date the job was last executed (if known)
nextExecution	Next date the job will be executed
lastLaunchError	Error text from last attempted launch
latestResultCompletionStatus	Completion status of latest run
latestResultId	ID value for latest run taskResult
latestResult	taskResult object for latest result

The table below describes the key attributes on the **TaskDefinition** object.

Attribute	Description
parent	taskItemDefinition object of parent if this is a specialization of another TaskDefinition
type	Task type (for searches and filtering in UI)
subType	Task sub-type (for searches and filtering in UI)
executor	Name of the task's executor class
signature	Metadata describing inputs and outputs of the task
arguments	Map of arguments to the executor (name/value pairs)
template	Flag indicating this is a template definition (not run directly but cloned with launch arguments fleshed out in the clone)
hidden	Flag indicating that the definition should not be displayed in the task launch UI (used when custom pages exist to manage tasks)
resultExpiration	Length of time results are kept before purging 0 = no expiration (keep indefinitely); >=1 = time in days; -1 = expire immediately; <=-2 = time in seconds
formPath	Optional location to find task form that is used to render the task arguments
progressMode	Publishes how executor will show progress, if at all
progressInterval	Interval at which the task will update progress
rights	Optional list of rights for restricting access to this task
resultAction	resultAction object that specifies how previous results of this task are to be processed; if not specified, Delete is assumed
concurrent	Flag indicating that the task can be launched even if there is already a task with this definition running
resultRenderer	Optional name of a custom result renderer page
signoffConfig	workItemConfig object that, when set, signifies that the task results must be signed by the workItem owner

The table below describes the key attributes on the **TaskResult** object.

Attribute	Description
schedule	name of the TaskSchedule object that ran the executor that produced this result
report	Object holding the report, if this was a reporting task
definition	taskDefinition object for the task; set by TaskExecutor -- may be null for some tasks (if result is intelligible on its own)

signoff	Signoff comments if signoff was enabled in the TaskDefinition
pendingSignoffs	Number of active signoffs remaining for the task
targetClass	Database object class of the associated object
targetId	Database id of the associated object
targetName	Display name of associated object (optional)
verified	Optional date field that indicates the result has been verified
terminated	Flag indicating task was terminated
completionStatus	Status of task when completed: success, failure, warning, or terminated

NOTE: These objects also relate to reporting, since reports are recorded in IdentityIQ as TaskDefinition objects. See *Reporting* for other reporting-related objects.

Request Scheduling

Requests are scheduled background activities that do not need the level of infrastructure built around Quartz tasks. The primary uses of the request objects are (1) email notification retries, (2) role sunrise/sunset actions (deferred activation/assignment or schedule deactivation/deassignment), and (3) launching workflows from within other workflows. Unlike tasks, their "results" are not visible in the UI, with the exception of role assignment/deassignment, which can be seen on the Identity's Events tab.

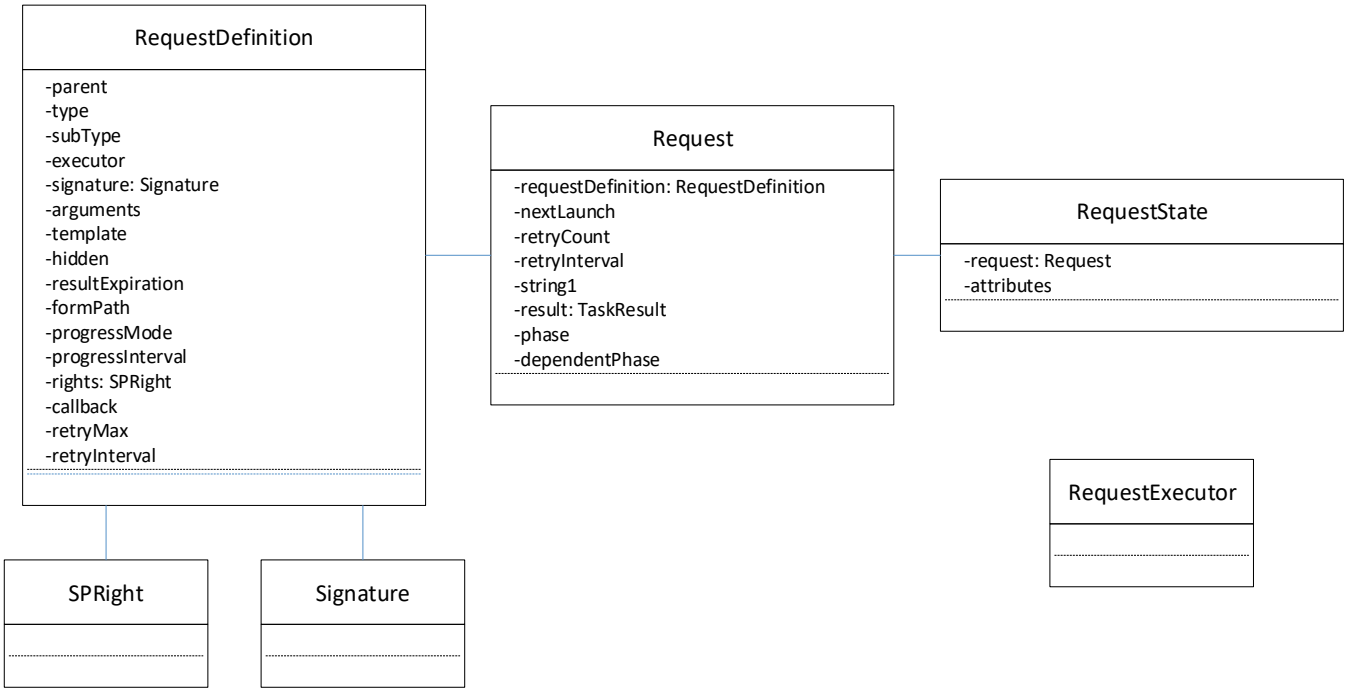


Figure 39: Request, RequestDefinition Objects and their Attribute Relationships

Version 6.2 implemented partitioned tasks through the Request infrastructure. When a partitioned aggregation or identity refresh task runs (for example), the task generates Request objects for each partition, and the processing of each partition is managed on a request host. The results from these processes are incorporated into the TaskResult object for the master task.

Version 7.3 introduced the concept of task resiliency, through which we can restart processes which were abnormally terminated due to a server failure without having to re-process the whole data set. The RequestState object stores information about a restart point for those tasks. This feature only applies to partitioned tasks.

The table below lists attributes of the **Request** object and their descriptions.

Attribute	Description
definition	requestDefinition object associated with the request (specifies executor, retry modes, arguments, etc.
nextLaunch	Desired date of execution for this request; email notification requests leave this null initially, to execute request immediately, and then set this value based on retryInterval if needed; event requests set this based on sunrise/sunset date specified; workflows can set this to determine when the independent workflow they are launching should kick off
retryCount	Number of retries that have been executed
retryInterval	Gap between consecutive retries in seconds
string1	General purpose sortable/searchable request property (can be null)
result	TaskResult object shared by a set of Requests created for a partitioned task
phase	Phase number for partitioned requests; valid only for requests with a shared TaskResult
dependentPhase	Phase number on which this phase is dependent; zero means dependent on all phases with lower number -1 means no dependencies -2 means won't run until process changes dependentPhase value – used when need to commit of large number of Requests over time and don't want RequestProcessor to start processing them until all are created

The table below describes the key attributes on the **RequestState** object.

Attribute	Description
request	Request object for which this RequestState object is holding the rollback-state information
Attributes	An Attributes map of values which define the information useful during a restart of the request; each request type can populate this map as needed for its specific processing

The table below describes the key attributes on the **RequestDefinition** object.

Attribute	Description

parent	taskItemDefinition object of parent if this is a specialization of another Request Definition
executor	Name of the request's executor class
signature	Metadata describing inputs and outputs of the request
arguments	Map of arguments to the executor (name/value pairs)
template	Flag indicating this is a template definition (not run directly but cloned with launch arguments fleshed out in the clone)
resultExpiration	Length of time results are kept before purging 0 = no expiration (keep indefinitely); >=1 = time in days; -1 = expire immediately; <=-2 = time in seconds
formPath	Optional location to find task form that is used to render the request arguments
progressMode	Indicates how executor will show progress, if at all
progressInterval	Interval at which the request will update progress
rights	Optional list of rights for restricting access to this request
callback	Name of the request callback object, which reports whether the request ended in success, permanent failure, or temporary (retry-able) failure
retryMax	Maximum number of retries
retryInterval	Time interval between retries

Archive Objects

Several object types in IdentityIQ can be archived. In most cases (except for BundleArchive), archived versions of objects cannot be viewed from the IdentityIQ user interface, but they are available for use in reporting or exporting from the environment.

Archive objects are intentionally stand-alone objects without connection to other objects in the system. Therefore, no interconnections exist, so no diagram is provided here to illustrate them.

The **BundleArchive** object is an archive of a Bundle object – a role. This archive is different from other archive objects in that it is created to support the role rollback option if a change to a role needs to be undone. The previous state (stored in the bundleArchive) of the role can be seen through the Role Editor when role archiving and rollback are enabled.

The **IdentityArchive** object is an archive of an Identity. It is similar to an IdentitySnapshot, but while snapshots are meant to be kept forever and therefore do not contain hard references to other objects in the environment which could be deleted at a later point, IdentityArchives are expected to be shorter-lived and can have those hard references.

The **BundleArchive** and **IdentityArchive** objects both extend the **Archive** class, adding no specific attributes of their own, so they look like this:

Attribute	Description
sourceId	ID of the original object being archived
name	Name of the original object being archived at the time the archive was created
creator	Name of the identity who created the archive

version	Numeric version number; used for archive objects which need to be ordered and presented as numbered versions
archive	XML blob containing the serialized object which has been archived

The **WorkItemArchive** object is an archive of a WorkItem – a unit of work assigned to a given user in the system. WorkItems only exist until they are completed by the assigned user (e.g. until the certification has been signed off, until the approval has been completed, until the form as been submitted, etc.). When work item archiving is turned on through the system config, WorkItemArchive objects can be created to store archived versions of the workItems after completion. These can be seen from the Manage -> WorkItems -> Archive page in the user interface. WorkItemArchive extendes SailPointObject, not the Archive object, and it contains these attributes:

Attribute	Description
ownerName	Name of the original work item owner
assignee	Name of a workgroup member that was assigned the item; only set if the workitem owner is a workgroup
requester	Name of the identity who originated the work item (can be null for system-generated items)
completer	Name of the identity who completed the work item
type	WorkItem Type (enumeration value – see Javadocs) for this work item
handler	Class to be notified when a change to the work item is persisted; not used in the archive object
renderer	URL fragment to the JSF include that will render the work item details
state	Workitem State (enumeration value) indicating the completion state of this work item
level	WorkItem Level (enumeration value) of the item; indicates significance; if null, implies “Normal”
targetClass	Database object class of the associated object (e.g. the associated Approval, Signoff, PolicyViolation object to which this work item pertains)
targetId	Database ID of an associated object
targetName	Display name of the associated object
identityRequestId	The name of the IdentityRequest that caused this workItem (can be null if note related to an identity request
attributes	Attributes map of workItem attributes
systemAttributes	Attributes map of system attributes
archived	Date the archive object was created
signed	Flag indicating whether the work item was electronically signed and therefore whether the archive can or cannot be deleted by the system (electronically signed objects are immutable in IdentityIQ)

The **CertificationArchive** object is an archive of a Certification object (which represents an access review). CertificationArchive objects cannot be viewed through the IdentityIQ user interface but are available for reporting and export. They are created by a Perform Maintenance task when the certification archive option is turned on.

Attribute	Description
certificationId	Unique ID of the original certification object; preserved as a soft reference to a certification; allows the certification to be archived and deleted without violating a Hibernate integrity constraint; provides a way to locate the archive that replaced a deleted certification
childCertificationIds	IDs of archived child certifications of the top level certification in this archive; child certifications are archived within their parents, so this list is flattened and stored when the archive is created so it can be searched for the archive that holds a child certification
certificationGroupId	ID of the CertificationGroup (the collection of access reviews, sometimes described as the “certification campaign”) that contained this certification
creatorName	Name of the user who generated the certification
ownerName	Name of the user responsible for this certification
signed	Date the certification process was completed
expiration	Date the certification expired or was due
comments	Comments entered by the certifier in the certification process
archive	XML blob containing the serialized Certification object

Other Historical Records

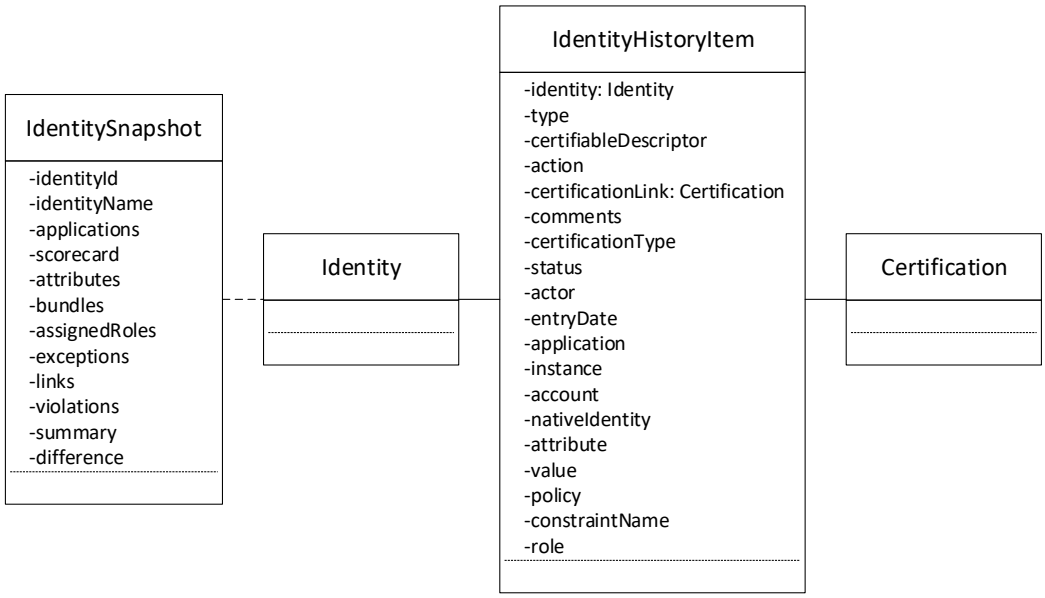


Figure 40: Historical-record objects and their relationships to other objects

The **IdentitySnapshot** object is a different type of object for recording identity history data. The system can be configured to auto-capture a snapshot of Identities on a regular basis so its state can be tracked at various points in time. The IdentitySnapshot is a simplified representation of a full identity object. Its attributes and their usages are described in the table below. It contains the ID and name of the Identity Cube to which it relates to allow it to be connected back to that Identity object.

Attribute	Description
identityId	Identity ID (GUID) of the original identity

identityName	Name of the Identity cube; stored instead of an identity reference because the identity itself can be deleted while the snapshots remain in the system
applications	CSV list of application names on which the identity has accounts at the time of the snapshot
scorecard	The Identity's risk scorecard at the time of the snapshot
attributes	Attributes map of Identity attributes
bundles	List of BundleSnapshots (simplified representations of roles) for the detected roles held by the identity
assignedRoles	List of RoleAssignmentSnapshots (simplified representation of assigned roles) for the identity's assigned roles
exceptions	List of EntitlementSnapshots (simplified representation of entitlements) held by the identity outside any role associations (sometimes called exception entitlements)
links	List of LinkSnapshots (simplified representation of links) for the accounts held by the identity
violations	List of PolicyViolation objects for the policy violations detected for the user at the time of the snapshot
summary	String value which summarizes the contents of the snapshot, used for fast display in a summary table
difference	String summarizing the differences between this snapshot and the previous one

IdentityIQ can also track certification history and comment history through **IdentityHistoryItem** objects. This table describes the attributes of the IdentityHistoryItem object and their usages.

Attribute	Description
identity	The Identity to which this history item pertains
type	Type of the history item (enumeration: either Decision or Comment)
certifiableDescriptor	CertifiableDescriptor object representing a unique identifier for the identity history item (identifies a specific role, entitlement, or policy violation from a specific certification
action	Action taken on the given certification item (only used if the Type of this IdentityHistoryItem is Decision)
certificationLink	Pointer back to the original certification containing the item (only if Type = Decision)
comments	Comments made on a given entitlement (only if Type = Comment)
certificationType	Type of the certification item (enumeration CertificationItem.Type – see javadocs)
status	Status of the action recorded by this history item (CertificationAction.Status enumeration)
actor	Name of the identity or process responsible for the history item
entryDate	Date the activity associated with the history item was performed (certification decision made or comment entered)
application	Attributes recorded for certification history items, depending on the type of certification
instance	
account	
nativeIdentity	
attribute	
value	
policy	

constraintName	
role	

Configuration Objects

IdentityIQ stores system and user interface configuration details in various configuration objects in the database. It also supports creation of custom objects for storing data required by a specific installation's processes. This collection of objects is described below.

These configuration objects are standalone objects without interconnections to other system objects, so no diagram appears here to illustrate them.

System Configuration Objects

The **Configuration** objects in IdentityIQ are a varied set of objects, storing attributes for the system configuration. The types of data recorded in Configuration objects are varied so the attributes map for these objects varies greatly from one to the next. The most commonly use configuration objects are System Configuration, Connector Registry, and Identity Selector Configuration. Occasionally, customers have implemented their own custom Configuration objects. This is generally not recommended; instead, the ***Custom*** object should be used for custom configuration objects to avoid namespace conflicts.

Attribute	Description
systemConfigCache	Static cache for the system configuration object; UI uses getSystemConfig method to access sysconfig frequently so this prevents the need to fetch it from the database each time
identitySelectorConfigCache	Static cache for the identity selector configuration object
attributes	Attributes map of configuration attributes relevant to the specific configuration object

The **UIConfig** object contains parameters which tell the system what attributes to display on various user interface pages. Most of the UIConfig entries affect columns in grid displays in the UI, but they can also specify attributes for non-grid UI pages.

Attribute	Description
cache	Static CacheReference so this object can be cached and not require repeated database hits to retrieve it each time it is needed
attributes	Attributes map which holds all of the configurations for rendering most of the grids and tables of data in the user interface

The **DatabaseVersion** object maintains a persistent version number for the object model and associated Hibernate schema. It is used during system startup to verify that the database being connected to has a schema which is compatible with the java classes in the system. These values only change during system upgrades and are used to ensure that the upgrade process is run properly – that the database and the schema have been updated in sync with each other.

Attribute	Description
systemVersion	Persisted database version identifier
schemaVersion	Persisted Hibername schema version identifier

Custom Data Storage Objects

The **Custom** object is an object type specifically created to allow installations to build their own custom configuration objects or objects containing data they need for various system operations. Each custom object is an attributes map so it can be used to hold whatever data is needed. It stores the map as an XML blob. These are intended to be exportable classes in relatively small numbers, and they should not be used for things like a custom audit log which could result in thousands of instances.

Attribute	Description
attributes	The attributes map which contains the needed data

CustomGlobal is a class used to maintain a static map of custom attributes, meant to be used for maintaining a set of global variables across calls to custom rules. There can be one CustomGlobal object in an installation. It should be populated and cleared as needed. Unlike Custom objects, this is not a top-level object which can be exported or viewed through the console or debug pages.

Attribute	Description
attributes	The attributes map which contains the needed data

These two objects define components and layouts available for the IdentityIQ Dashboards (the default page users see after logging in to IdentityIQ).

Dashboard Configuration Objects

NOTE: Dashboards still exist in versions 7.0+ but are no longer the default view for users. They are accessed through the Quicklink menu **Dashboards** option.

The **DashboardContent** objects define each of the component widgets which can be included on each dashboard, including its source and any widget-specific parameters.

Attribute	Description
source	Source of this content item; currently a URL like “contentReport.xhtml”
regionSize	default size of the content object; used to identify which column the content object should reside in
rights	optional list of SPRights for restricting access to this content item
parent	Parent DashboardContent object; only set if this content is a subclass of another
arguments	Attributes map of arguments to the DashboardContent object
title	Title of the content object; can be a string or a localizable messageKey

required	Flag indicating the content object is a required part of the dashboard it is part of and cannot be removed, though it can be dragged around to different areas of the dashboard
enablingAttributes	Map of identity attributes which grant access to this resource, keyed off attribute name; if specified, the dashboard content object will only be displayed for users who have that matching attribute value
type	Type enumeration which associates the content with a specific dashboard type (values are My, Lifecycle, or Compliance)

The **DashboardLayout** objects define the different organizations of components which can be selected for a user's dashboards (e.g. one column, two column, and two column with a one-column area at the top or bottom of the page).

Attribute	Description
type	Human readable layout name (e.g. Two Columns, etc.)
regions	List of the regions contained in the layout object

Widgets

Starting in version 7.0, the landing page for all users changed from being the "Dashboard" to being the "Home" page. The Home page is made up of two sets of objects: Quicklink Cards and Widgets. Through the UI, Widgets can be added, removed, and rearranged per user. Out of the box, most Widgets are available for display on the Home page of any user, though a few include identity selectors that allow them to be seen only be users with the appropriate permissions or status in IdentityIQ. You can further customize the visibility of widgets by adding or changing these identity selectors to the widget definitions.

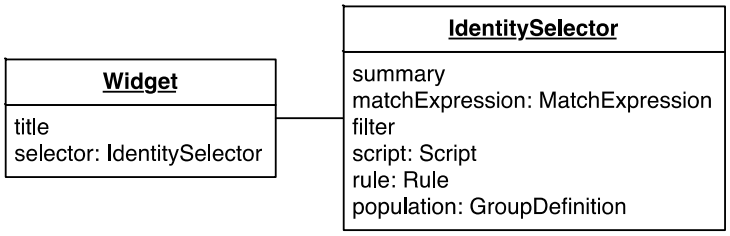


Figure 41: Widget Object and its Attribute Relationships

The **Widget** object is a simple object whose behavior is built into the core code of IdentityIQ and cannot be modified. The names of widgets are specified either in the system configuration or in each user's UIPreferences object to cause them to display on the user's Home page. The title displays in the UI and the selector specifies which users have permissions to see the widget.

Attribute	Description
title	Localizable message catalog key that is displayed as the name of the widget in the UI

selector	Identity Selector that specifies the requirements for a user to be able to see this widget on their Home page; users who do not meet the selection criteria will not be able to add this widget to their Home page at all

In this context, attributes of the **IdentitySelector** object and their descriptions are listed in the table below. IdentitySelector represents different ways sets of Identities are allowed access to a given Widget to be able to optionally include them on the user's Home page. IdentitySelector objects are used in several contexts in IdentityIQ, including as part of Quicklink population specifications, in role assignment rules, in lifecycle event "included identities" specifications, and in advanced policy definitions.

Attribute*	Description
summary	Brief summary of what the selector is doing
matchExpression	Simple list of attribute value or permission matches
filter	Compound filter that can accommodate Boolean operators; may reference both identity and link attributes in any combination
script	Inline script; returns true if the role should be assigned to the user
rule	External rule reference; returns true if the role should be assigned to the user
population	Population reference; filter only applies to identity cube attributes

***** IdentitySelector does not contain the attributes discussed in the *Objects' Shared Attributes* section.

ObjectConfig

Extended attribute definitions for several different types of objects (Identity, Link, Bundle, etc.) are stored in the ObjectConfig object. The contents of the objectConfig are needed frequently by the persistence layer so they are cached by a system cache service. Both the cache service and the owning object are given a CacheReference to the objectConfig to interact with it.

The attribute values for each Identity, Link, Bundle, etc. are stored in that object's attributes map. The ObjectConfig is the extended attribute *definition*, and only one exists for each object type (e.g. there is a single Identity ObjectConfig, Link ObjectConfig, etc. that specifies the extended attribute configuration for all objects of that specific type).

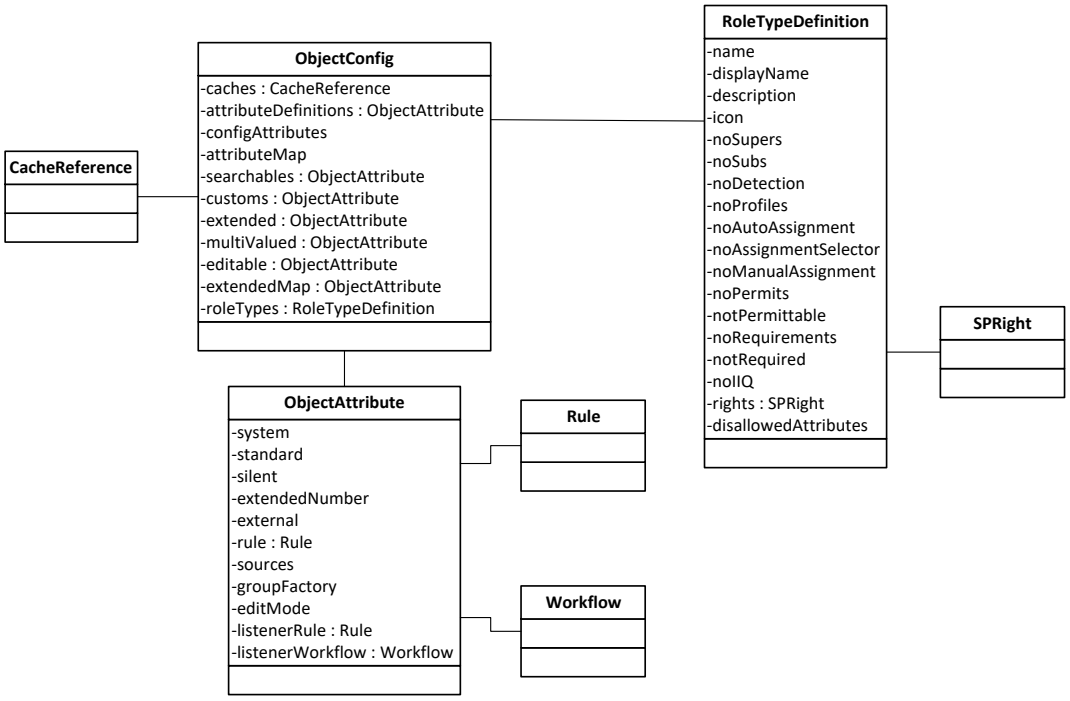


Figure 42: ObjectConfig Object and its Attribute Relationships

These are the attributes of the **ObjectConfig** object and their descriptions.

Attribute	Description
caches	Map of cached ObjectConfigs keyed to the full name of the class
attributesDefinition	List of ObjectAttribute objects; definitions for the abstract attributes
configAttributes	Other configuration attributes (map of name/value pairs)
attributesMap	Runtime map used to quickly locate and ObjectAttribute by name (name/value pairs; value is ObjectAttribute object)
searchables	List of attributes that can be used in searches (objectAttribute objects)
customs	List of only custom attributes (objectAttribute objects)
extended	List of extended attributes (with reserved column number) (objectAttribute objects)
multi-valued	List of attributes that are multi-valued (objectAttribute objects)
editable	List of attributes that can be edited through the UI (objectAttribute objects)
extendedMap	runtime map used to quickly locate extended attributes (those with a reserved column number) by name (name/value pairs; value is objectAttribute object)
roleTypeDefinition	List of RoleTypeDefinition objects (only used by the Bundle objectConfig); these objects define the available role types and the features enabled for each (e.g. whether Permitted Roles can be assigned for the role, whether the role can be manually assigned to an Identity, etc. – options defined in System Setup -> Role Configuration)

These are the attributes of the **ObjectAttribute** object and their descriptions. These are specified for each attribute within the ObjectConfig to tell the system how to store/populate that attribute. These are the

attributes which are set through the system configuration pages (e.g. Identity Mappings, Account Mappings, etc.).

Attribute	Description
system	Boolean indicating whether this is a system-defined variable (e.g. username, detected roles, last refresh)
standard	Boolean indicating if this is one of the default identity attributes (e.g. manager, email, firstname)
silent	Boolean indicating whether this is an attribute which should be included as a normal attribute or if it is a special case that should be suppressed from those views (e.g. capabilities is an attribute of an identity but is not considered an "identity attribute" that would be displayed on the Attributes tab of the identity view, so it is marked as "silent")
extendedNumber	Column number of the placeholder database columns (as opposed to named columns) into which this attribute's values should be stored (e.g. 1 = extended1, 2 = extended2, etc.); set for searchable attributes only
namedColumn	Boolean indicating the attribute should be stored in a named column (matching the name of the attribute) rather than a placeholder column; set for searchable attributes only
external	Not currently used
rule	Rule reference to the rule used to calculate the attribute value
sources	The application sources used to populate the attribute value
groupFctory	Boolean indicating whether the attribute is available to use in group generation
editMode	String specifying whether the attribute is UI-editable and how the system should treat it once altered: ReadOnly, Temporary, Permanent
listenerRule	Rule reference to the rule which can auto-execute when the attribute's value changes
listenerWorkflow	Workflow reference to a workflow which can auto-execute when the attribute's value changes

ColumnConfig

The ColumnConfig object specifies the source and formatting of columns displayed in grids throughout the IdentityIQ UI. They are specified for each grid in the UIConfig configuration object.

These are standalone objects. No diagram is provided because there are no connections to illustrate.

This table describes the attributes on the **ColumnConfig** object.

Attribute*	Description
headerKey	Text to display as the column header; often a message key in the iiqMessages.properties file
property	Object property that will be pulled from the database for this column
name	Name of the column; often same as dataIndex
dataIndex	JSON-safe name for column; can be same as property

sortProperty	Column by which data should be sorted when this column is selected as the sort column; typically same as property, but when column is a calculated field, it requires a different sort column (or should be marked as not sortable)
secondarySort	Used to sort rows when column can have non-unique values (so sortProperty will not completely sort list by unique values)
sortable	Flag indicating whether the grid data can be sorted by this column
hideable	Flag indicating whether the column can be hidden in the grid
noEscape	Flag to disable HTML escaping of the column value; intended primarily for use in JSON data sources to permit embedding of s for multi-line text in table cells. (infrequently used)
localize	Flag to enable localization of the column value (if true); intended for pages containing columns of message keys to avoid the overhead of message localization for every cell in a large datastore (infrequently used)
renderer	ext render function to use to render this column
editorClass	ext xtype of the editor component to use in this column
dateStyle	ext date style for this column
timeStyle	ext time style for this column
pluginClass	ext xtype of the plugin class for this column
percentWidth	Column width expressed as a percentage of the grid
width	Column width expressed in pixels
flex	Column width in ratios (all flex numbers are added up and the total divided by the number of columns to create a percentage width)
hidden	Flag indicating the column should be hidden from the grid by default
fieldOnly	Flag indicating the column should be in the datasource but never displayed in the grid
evaluator	Name of the java class that evaluates the value of this column (only applies to the certification detail view grids)

***** ColumnConfig does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Full Text Index

Full text index is an option that can be turned on for Lifecycle Manager in the LCM Configuration system setup page, beginning in version 6.0. IdentityIQ's full text index capabilities are based on the Apache Lucene text search engine library. The Full Text Index Refresh task builds (or rebuilds) the full text index on ManagedAttributes, Roles, or both (default is both). The indexes themselves are stored in the file system. The FullTextIndex objects record information about the indexing process, including the last refresh date, any errors encountered, and data about the index (storage path location, indexed fields, etc.).

The default location for recording indexes is the WEB-INF directory of the IdentityIQ installation (in 6.0 GA, it was the base installation directory but this was altered in 6.0p1). This location is a configurable option set through the indexPath attribute in each FullTextIndex object's attributes map.

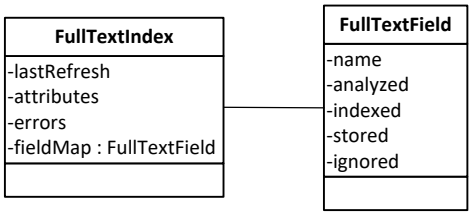


Figure 43: FullTextIndex Object and its Attribute Relationships

A **FullTextIndex** object is created to represent each index (ManagedAttribute and Bundle). Its attributes are listed in this table:

Attribute	Description
lastRefresh	Date this index was last refreshed
attributes	Attributes map (stored as XML blob) for index; includes: indexPath, fields list
errors	List of error messages captured in last refresh, if any
fieldMap	transient field lookup cache; map of name – FullTextField pairs

A **FullTextField** object represents each field named in the index. Its attributes are listed in this table:

Attribute*	Description
name	name of the attribute to be indexed
analyzed	flag indicating field will be broken up and indexed for full text search with substring matching
indexed	flag indicating field will be stored and can be used in filters but cannot be use dwiht substring matching
stored	flag indiating field will be returned in serach results but cannot be used in full text search or filters
ignored	flag indicating field will be ignored if sent down in a filter

***** FullTextField does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Server

Beginning in version 6.2, Server objects are created automatically for every IdentityIQ server that connects to the IdentityIQ database. Once created, they can be used to store server-specific configuration information. These objects are used for controlling request and task processing, for reporting server status to administrators in the IdentityIQ UI, and for recording server health. They are particularly important to the processing health of the system in a multi-server environment.

These are standalone objects. No diagram is provided because there are no connections to illustrate.

The attributes of the **Server** object and their usages are shown in this table:

Attribute	Description
attributes	an Attributes object (expansion of a Map object) containing configuration and status attributes; attributes are:

	• **maxRequestThreads** – maximum number of processing threads to use for request processing on this server; overrides the Request ServiceDefinition maxThreads value for this server • **taskThreads**, **requestThreads** –number of active task/request processing threads at last heartbeat check • **cpuUsage** - % CPU utilization at last heartbeat check • **requestServiceStarted** – flag indicating whether the server is running as a request server
heartbeat	time of last heartbeat
inactive	flag indicating the Heartbeat Service believes the server to be inactive (no heartbeat updated within last 2 minutes)

ServiceDefinition

The ServiceDefinition objects have existed for a while, but their importance increased in the 6.2 release because starting from that release, they are now used to configure the Request and Task hosts. Prior to that version, that configuration was managed through the iiq.properties file; while that method is still supported for backward compatibility, customers are encouraged to migrate to the new mechanism for specifying request and task hosts with this version. Specifically, the "hosts" attribute on the Request and Task ServiceDefinition objects should list the names of the hosts on which each service should run; by default, all attached hosts will be used as Request and Task hosts.

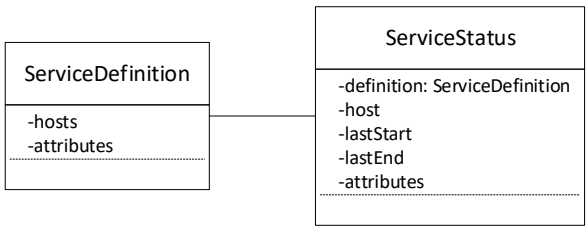


Figure 44: ServiceDefinition Object and its Attribute Relationships

The attributes of the **ServiceDefinition** object and their usages are shown in this table:

Attribute	Description
hosts	The list of hostnames on which the service runs; set to "global" by default for most services, but can be constrained to run on a subset of attached servers by specifying a comma separated list of host names
attributes	An attributes map, which varies by service

The attributes of the **ServiceStatus** object and their usages are shown in this table. ServiceStatus tracks status information about each ServiceDefinition: host it is running on, last run date info, and service-specific attributes. Multiple ServiceStatus objects can exist per ServiceDefinition as they track the service's execution on different hosts.

Attribute	Description
definition	ServiceDefinition object to which this status information pertains

host	Specific host the service is running on; also tracked in the name of this ServiceStatus
lastStart	Date the service last began executing
lastEnd	Date the service last ended execution
attributes	An attributes map of service-specific status attributes, which varies by service

Electronic Signatures

Electronic signatures were introduced in IdentityIQ version 6.1. Each is represented through an ESignatureType object, which contains the signature meaning: the legal text to which the signer is agreeing when they affix their electronic signature. ESignatureType objects do not get directly connected to other objects; they are referenced by name in the attributes map of certain configuration objects (like CertificationDefinition), but when they are applied at the time of sign-off, the meaning text is copied into the associated object. This ensures clarify of the legal meaning of the person's sign-off without any referential integrity requirements or consequences.

The **ESignatureType** object contains these attributes:

Attribute*	Description
name	Internal name of the signature object, used to reference the signature from in workflows, certification configurations, etc.
displayName	Optional display name to show in selection boxes in the UI
meanings	Map of meaning texts, keyed by locale (for localizing based on the user's browser language settings)

***** ESignatureType does not contain the attributes discussed in the *Objects' Shared Attributes* section.

Login Configuration Objects

IdentityIQ supports three methods of authentication for login: single sign-on, pass-through authentication, and native (internally-stored password based) authentication. When configured for pass-through authentication, IdentityIQ allows users to reset their pass-through application password if they forget it by authenticating through authorization questions; they must have previously recorded answers to their authentication questions and they must provide matching answers to those previously recorded values to authenticate and reset the password.

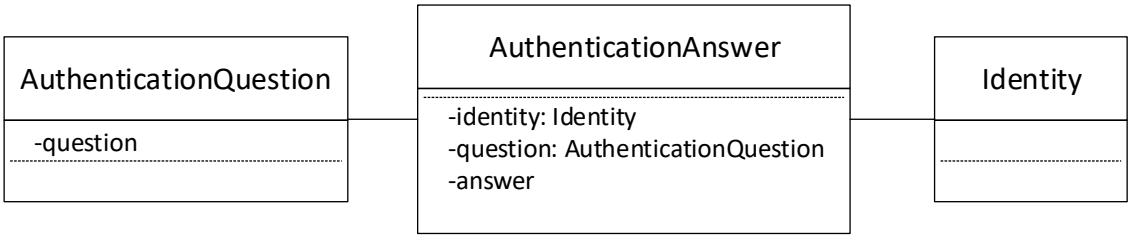


Figure 45: Authentication Questions/Answers and their object relationships

The **AuthenticationQuestion** objects record all of the questions which are available for users to answer. IdentityIQ ships with a list of 7 questions, but additional ones can be configured for any installation and the provided ones can be modified or removed as needed.

Attribute	Description
question	Question text or a localizable message key which contains the question text in the messages catalogs

The **AuthenticationAnswer** objects contain user-specific answers to the authentication questions. These are not top-level objects which can be exported or seen from the console or debug pages since they contain secure (encrypted) data.

Attribute	Description
identity	The Identity object to which this answer applies (the user who recorded the answer)
question	The AuthenticationQuestion to which this answer is connected
answer	String value entered by the user as the answer to this question

The **Dictionary** object contains a collection of words or strings which should not be allowed as passwords or as substring of the password. This applies only to internally-stored passwords for native authentication.

Attribute	Description
name	Name of the object (always PasswordDictionary)
terms	List of terms which are disallowed

User Specific Configuration Objects

IdentityIQ allows various customizable user experience configurations in these objects.

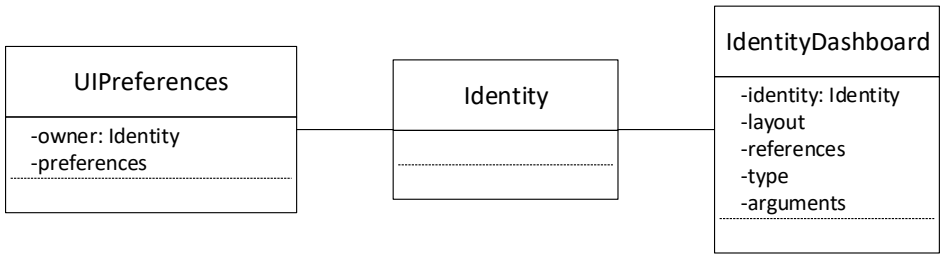


Figure 46: User Interface Customization objects and their object relationships

The **UIPreferences** object records UI preferences associated with an identity – UI layout preferences and options they have selected while using IdentityIQ (e.g. preferred certification layout, whether to continue to show certification pop-up help, sort preferences on various grids).

Attribute	Description
owner	Identity to whom this set of preferences relates; property inherited from SailPointObject
preferences	an attributes map which specifies the preferences; this is made up of a set of values which correspond to constants and enumerations in this class that ultimately drive the UI display choices

The **IdentityDashboard** object records the user's dashboard preferences – the widgets which should be displayed on their IdentityIQ Dashboards and the order in which they should be displayed. There is a default IdentityDashboard object for each of the dashboards which is used for all users who have not set up their own custom dashboard layout configurations.

Attribute	Description
identity	Identity object which owns this custom dashboard configuration
layout	DashboardLayout which represents the physical layout for this dashboard
references	List of DashboardReferences, which are map the regions to the DashboardContent which should be displayed there
type	Dashboard type: an enumeration which can be My (for My Dashboard), Compliance, or Lifecycle
arguments	List of arguments which apply to a given dashboard

Environment Monitoring

Beginning in version 7.3, IdentityIQ offers an Administrator Console page where administrators can monitor the health of their servers. The MonitoringStatistic objects define the statistics that can be collected and point to the system classes that know how to collect those statistics; these are configuration objects that are not modified by the installation. The ServerStatistics objects contain the snapshots of statistical data for a given server and monitoring statistic set; this data is what is shown on the Administrator Console -> Environment -> Hosts page when the administrator clicks the Host Name link and views the sets of statistics.

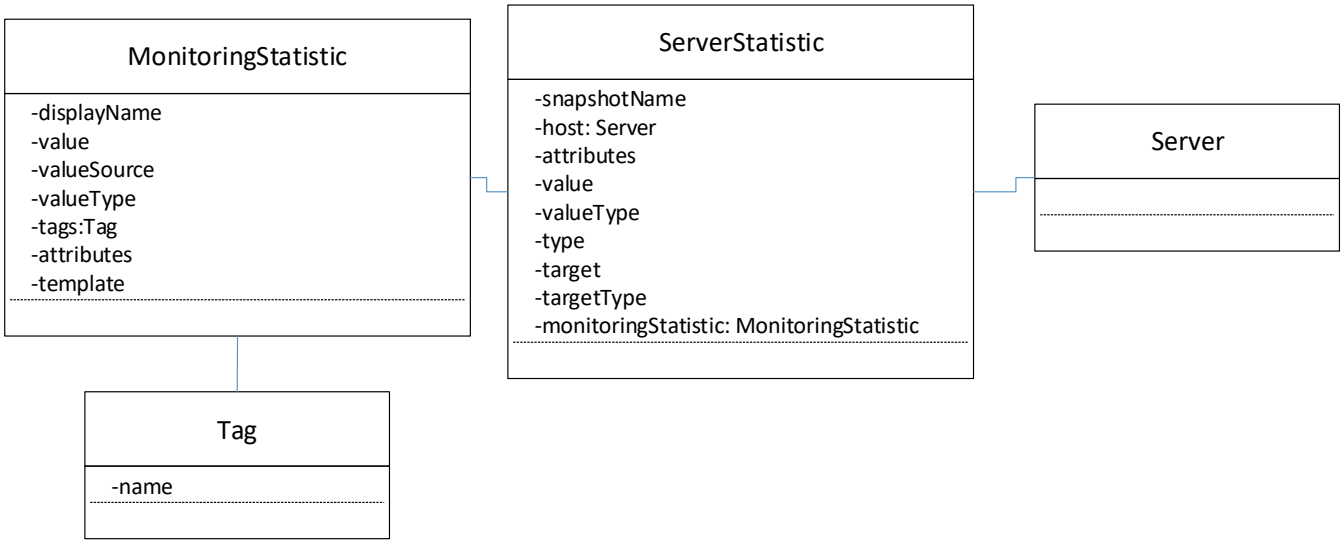


Figure 7: MonitoringStatistics objects and their object relationships

The **MonitoringStatistic** object contains these attributes.

Attribute	Description
displayName	Name to show when displaying
value	Scriptlet for how to obtain value
valueSource	Parsed value field into scriptlet

valueType	Type of object returned when valueSource is evaluated
tags	Type of statistic
attributes	Map of related attribute values
template	True if this is a template that requires customization

The **ServerStatistic** object contains these attributes.

Attribute	Description
snapshotName	Name of the snapshot this was captured with; used to group statistics
host	Server object representing the host from which this statistic was captured
attributes	Map of related attribute values
value	Capture statistical value
valueType	Type of the value – copied from MonitoringStatistic at time of creation
type	Type of statistic – copied from MonitoringStatistic at time of creation
target	Name/ID of the object referenced
targetType	Type of object referenced
monitoringStatistic	Reference to the associated MonitoringStatistic object

Plugin

The Plug-in Framework, introduced in a 7.0 patch version and formalized in 7.1, allows for rich functionality developed by third parties to be integrated into the IdentityIQ product. For more on the available plugins and development guidance, see https://community.sailpoint.com/community/plugins.

These are standalone objects. No diagram is provided because there are no connections to illustrate.

The **Plugin** object contains these attributes. It is created during the Plugin installation process.

Attribute	Description
installDate	Last date the plugin was installed or updated
displayName	Name to display to the UI/end user
version	Plugin version number
disabled	Flag to enable or disable the plugin functionality
rightRequired	SPRight required for the plugin to work
minSystemVersion	Lowest IdentityIQ version the plugin will run on
maxSystemVersion	Highest IdentityIQ version the plugin will run on
attributes	Attributes map containing the PluginConfiguration and any configurable settings needed by the framework
certificationLevel	Level of certification given to this plugin (enumeration: None, Bronze, Silver, Gold)
position	Numeric position of this plugin, relative to other plugins, to be used in determining start order
File	Persisted Zip file reference to the Plugin's original archive

Classification

New in version 8.1. Classifications allow the flagging and categorization of roles and entitlements, Classifications can be used in certifications and policies, to help with monitoring and controlling the access users have to sensitive data. Access requests, approvals, and access reviews can be configured to show a classification icon with any role or entitlement that grants access to sensitive data, so that the users responsible for making access decisions can see which entitlements allow potentially risky access. Classifications can be added manually to Roles and Entitlements via the UI, imported via File Access Manager aggregation, or added automatically to cloud-enabled entitlements via the Cloud Access Management integration. There is no limit to the number of Classifications that Roles and Entitlements can reference.

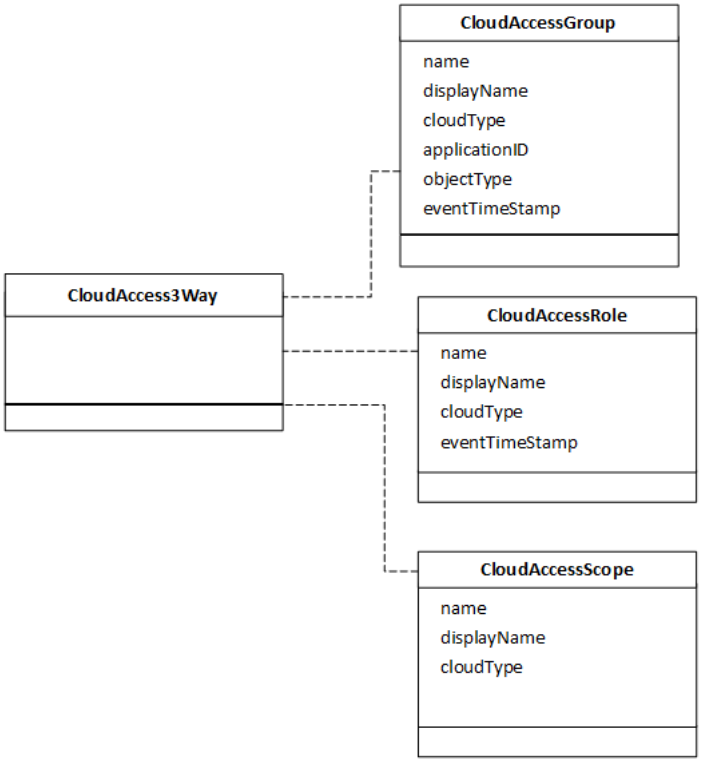
The Classification object holds classification data for a defined Classification. ObjectClassification objects act as intermediary objects to store meta data regarding the relationship of a given SailPointObject to a Classification. ObjectClassification objects are referenced on Bundles (Roles) and ManagedAttributes (Entitlements).

This table lists attributes for the Classification object.

Attribute	Description
id	The internal SailPoint Id of the classification
Date created	Date on which the classification was created
Date modified	Date on which the classification was last modified
name	Name of the classification. This must be unique
displayName	DisplayName of the classification
displayableName	The displayName if it is non-null; otherwise returns the name
attributes	Attributes map used to store any extra information. The localized descriptions will be stored in the attributes map, similar to other Describable objects.
origin	Origin from which the classification was sourced. In the case of Application promotion, this will be the name of the application. In the case of File Access Manager, this will be FileAccessManager
type	Type of the classification. This can be used to further aid in grouping/filtering classifications

Cloud Access Management Objects

New in version 8.2. Cloud Access Management is a governance offering for multi-cloud environments that is used to identify who has access to what, how that access is being granted, and to implement pre-configured policies that automate detection of compliance violations. **Cloud Access Management is sold separately from IdentityIQ, so your instance of IdentityIQ may not include these objects.**



- CloudAccessGroup: represents the concept of Cloud Access Management group. Attributes are:
  - name: the unique identifier of the CloudAccessGroup provided by Cloud Access Management
  - displayName: the display name as provided by Cloud Access Management
  - cloudType: the cloud type, such as Amazon Web Services, Azure, or Google Cloud Platform
  - applicationId: matches the ManagedAttribute's application
  - objectType: matches the ManagedAttribute's objectType
  - eventTimeStamp: when the info was sent from Cloud Access Management
- CloudAccessRole: represents the concept of Cloud Access Management role. Attributes are:
  - name: the unique identifier of the CloudAccessRole provided by Cloud Access Management
  - displayName: the display name as provided by Cloud Access Management
  - cloudType: the cloud type, such as Amazon Web Services, Azure, or Google Cloud Platform
  - eventTimeStamp: when the info was sent from Cloud Access Management
- CloudAccessScope: represents the concept of Cloud Access Management scope. The definition of a Cloud Access Management scope is specific to the Cloud Service Provider. Amazon Web Services scopes are limited to Amazon Web Services accounts within an org. Azure scopes refer to management groups, subscriptions, and resource groups. Google Cloud Platform scopes refer to folders and projects.

  Attributes are:
  - name: the unique identifier of the CloudAccessScope provided by CAM
  - displayName: the display name as provided by Cloud Access Management
  - cloudType: the cloud type, such as Amazon Web Services, Azure, or Google Cloud Platform

- CloudAccess3Way: Because a CloudAccessRole can have different CloudAccessScopes depending on the context of the CloudAcesssGroup, the relationships are modeled using this association table. This association model provides the flexibility of using both the CloudAccessGroup and CloudAccessRole to find the list of CloudAccessScope.