Tag Based Policies

1 Introduction

Apache Ranger provides centralized security for Enterprise Hadoop ecosystem, including fine-grained access control and centralized auditing. As of 0.5 version, Apache Ranger policies enable authorization for specific set of resources – like HDFS files/directories, Hive databases/tables/columns, HBase tables/column-families/columns, etc. Features like centralized policy management for many Hadoop components, ability to specify resource sets using wildcards and delegated administration model, make the security administration of Hadoop resources much simpler to manage.

Ability to authorize access based on tags associated with the resources, instead of the resources themselves, offers many advantages. One of the important advantage is the separation of resource-classification from access-authorization. For example, resources (HDFS file/directory, Hive database /table/column etc.) containing sensitive data like social-security-number/credit-card-number/sensitive-health-care-data can be tagged with PII/PCI/PHI – either as the resource enters the Hadoop ecosystem or any time later. Once a resource is tagged, the authorization for the tag would automatically be enforced, thus eliminating the need to create or update of policies for the resource. Also, a single authorization policy for a tag can be used to authorize access to resources across various Hadoop components – which eliminates the need to create separate policies in each component.

The goal of this document is to provide an overview of tag-based policies implementation in Apache Ranger.

2 Tag-based policy

Apache Ranger introduces a new service-type called ‘tag’ to work with tag-based policies. The new service-type ‘tag’ is similar to other existing service-types – HDFS, Hive, HBase, YARN, Strom, etc. With this approach, the users can use existing/familiar resource-based policy UI for tag-based policies as well. In addition, this also enables reuse of existing infrastructure that deal with Ranger Policies – like REST APIs, persistence, custom conditions, policy engine, etc.

2.1 Ranger Admin UI

Apache Ranger provides a new UI page, named ‘Tag Based Policies’, to work with tag based policies. The workflow to create/update tag-based policies is essentially same as with the existing ‘Resource Based Policies’. Start by adding a tag service instance, in which tag-based policies can be created. Multiple tag service instances can be created – like tag-dev/tag-test/tag-prod, to group tag-based policies for different clusters.

Policy UI for tag-based policy looks very similar to existing resource-based policies. The name of the tag should be specified at the top half of the page; the bottom half of the page provides the UI to specify permissions for users and groups. Following are few differences from resource-based policies UI:

- Permissions UI lists the permissions available in all the service-types. This allows policy authors to restrict type of accesses users/groups can perform on tagged resources
- Wildcards are not allowed in tag names. Also only one tag can be entered per policy
- Delegated Admin is not available for tag-based policies. Currently only an administrator can work with tag-based policies

2.2 Update component services for tag-based policies

Apache Ranger plugins enforce the authorization policies defined in the component service – like hive-dev/hive-test/hive-prod. For the plugins to also enforce tag-based policies, the component service must be updated to refer to a specific tag service instance (like tag-dev/tag-test/tag-prod). Follow the steps below:

- go to ‘Resource Based Policies’ page
- click on the Edit button of the component service that needs to be updated
- select appropriate tag service name from the list of services shown in ‘Select Tag Service’

3 Tag Store

Details of tags associated with resources are stored in a tag store. Apache Ranger plugins retrieve the tag details from the tag store for use during policy evaluation. To minimize the performance impact during policy evaluation (in finding tags for resources), Apache Ranger plugins cache the tags and periodically poll the tag store for any changes. On detecting change, the plugins update the cache. In addition, the plugins store the tag details in a local cache file – just as the policies are stored in a local cache file. On component restart, the plugins will use the tag data from the local cache file if the tag store is not reachable.

In the current release, Apache Ranger plugins download the tag details from the store managed by Ranger Admin. Ranger Admin persists the tag details in its policy store and provides a REST interface for the plugins to download the tag details.

4 Tag Sync

Apache Ranger introduces a new module, ranger-tagsync, to populate the tag store from the tag details available in an external system like Apache Atlas. Tag sync is a daemon process similar to ranger-usersync process.

In the current release, ranger-tagsync supports receiving tag details from Apache Atlas via change notifications. As tags are added/updated/deleted to resources in Apache Atlas, ranger-tagsync would receive notifications and update the tag store.

5 Tags

Tags in Apache Ranger can have attributes. Tag attribute values can be used in Ranger tag-based-policies to influence the authorization decision.

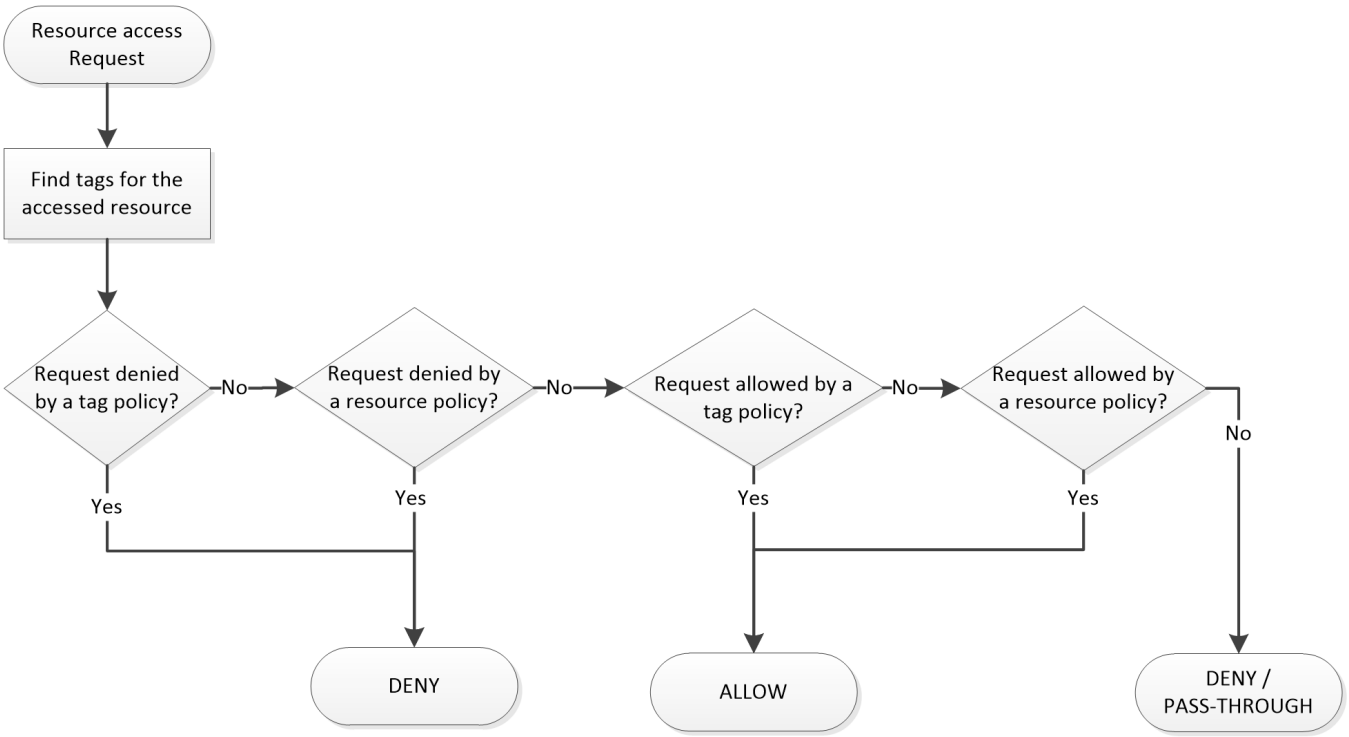
For example, to deny access to a resource after a specific date:

- add EXPIRES_ON tag to the resource
- add a tag attribute, named expiry_date, with its value set to the expiry date
- create a Ranger policy for EXPIRES_ON tag
- add a condition in this policy to deny the access when the date specified in expiry_date tag attribute is later than the current date

In fact, the above detailed EXPIRES_ON tag policy is created as the default policy in tag service instances.

6 Tags in policy evaluation

While authorizing an access request, Apache Ranger plugin evaluates applicable Ranger policies for the resource being accessed. Following diagram shows the details of the policy evaluation flow. More details on these steps are explained in the sections below.



Apache Ranger Policy Evaluation Flow with Tags

6.1 Finding tags

Apache Ranger stack model, introduced in Ranger 0.5, supports a service to register context enrichers, which are used to update context data to the access request.

Tag service, which is introduced in tag-based policies feature, adds a context enricher named RangerTagEnricher. This context enricher is responsible for finding tags for the requested resource and adding the tag details to the request context. This context enricher keeps a cache of the available tags; while processing an access request, it finds the tags applicable for the requested resource and adds the tags to the request context. The context enricher keeps the cache updated by periodically polling Ranger Admin for changes.

6.2 Evaluating tag-based-policies

Once the list of tags for the requested resource are found, Apache Ranger policy engine will evaluate the tag-based-policies applicable for the tags. If a policy for one of these tag results in deny, the access will be denied. If none of the tags are denied and if a policy allows for one of the tags, the access will be allowed. If there is no result for any tag or if there are no tags for the resource, the policy engine will evaluate the resource-based policies to make the authorization decision.

6.3 Using tags in conditions

Apache Ranger stack model allows use of custom conditions while evaluating the policies for authorization. Apache Ranger policy engine makes various request details - like user, groups, resource and context, available to the conditions. Tags in the request context, which are added by the enricher, are available to the conditions and can be used to influence the authorization decision.

The default policy in tag service instances, for EXPIRES_ON tag, uses such condition to check if the request date is later than the value specified in tag attribute expiry_date.

7 Setting up

7.1 Apache Atlas

Ensure that Apache Atlas is setup in the environment and configured to send entity and trait notifications to Apache Atlas clients via Kafka. Please refer to appropriate Apache Atlas documentation for more details.

7.2 Build Apache Ranger

Apache Ranger with support for tag-based policies is available in Apache branch named tag-policy (https://github.com/apache/incubator-ranger/tree/tag-policy). Instructions to build Apache Ranger from this branch using a Linux/Unix shell are given below. Please remember to set JAVA_HOME environment variable to appropriate value before executing these commands:

$ git clone git://git.apache.org/incubator-ranger.git

$ cd incubator-ranger

$ git checkout tag-policy

$ git pull

$ mvn clean compile package install assembly:assembly

Once the build completes, archive files containing the binaries should be available under target directory, as shown below:

$ ls –l target/*.tar.gz

target/ranger-0.5.0-admin.tar.gz

target/ranger-0.5.0-hive-plugin.tar.gz

target/ranger-0.5.0-tagsync.tar.gz

target/ranger-0.5.0-usersync.tar.gz

7.3 Install Apache Ranger

Install Ranger components (Admin, Usersync and plugins) using the instructions available at Apache Ranger wiki page. Then following the instructions below to install and configure TagSync:

$ cd /usr/local

$ sudo tar xvfz ranger-0.5.0.tagsync.tar.gz

$ sudo ln –s ranger-0.5.0.tagsync ranger-tagsync

$ cd /usr/local/ranger-tagsync

Review install.properties and update as necessary; especially review the following:

TAGADMIN_ENDPOINT = http://localhost:6080

TAGSYNC_ATLAS_KAFKA_ENDPOINTS = localhost:6667

TAGSYNC_ATLAS_ZOOKEEPER_ENDPOINT = localhost:2181

TAGSYNC_ATLAS_CONSUMER_GROUP = entityConsumer

Once install.properties is updated, run the setup script with the following command:

$ ./setup.sh

7.4 Start Apache Ranger

After completing the installation, start Apache Ranger components with the following commands:

$ service ranger-admin start

$ service ranger-usersync start

$ service ranger-tagsync start

In addition, restart the components where Ranger plugin was installed.

8 Examples

In this section, we will go through use of tag-based policies to implement the following two usecases:

- Access to objects tagged with PII should only be allowed to users in audit group
- Access to objects tagged with EXPIRES_ON should not be allowed after the date specified in tag attribute expiry_date

8.1 Hive Schema

We will use the Hive schema given below to implement the usecases:

Database	Table	Columns
finance	**tax_2010**	**ssn**, fed_tax, state_tax, local_tax
hr	employee	id, name, **ssn**, join_date, locatiion

Following Hive SQL statements can be used to create this schema using beeline command line:

create database finance;

use finance;

create table tax_2010(ssn STRING, fed_tax INT, state_tax INT, local_tax INT);

create database hr;

use hr;

create table employee(id INT, name STRING, ssn STRING, join_date DATE, location STRING);

8.2 Tags

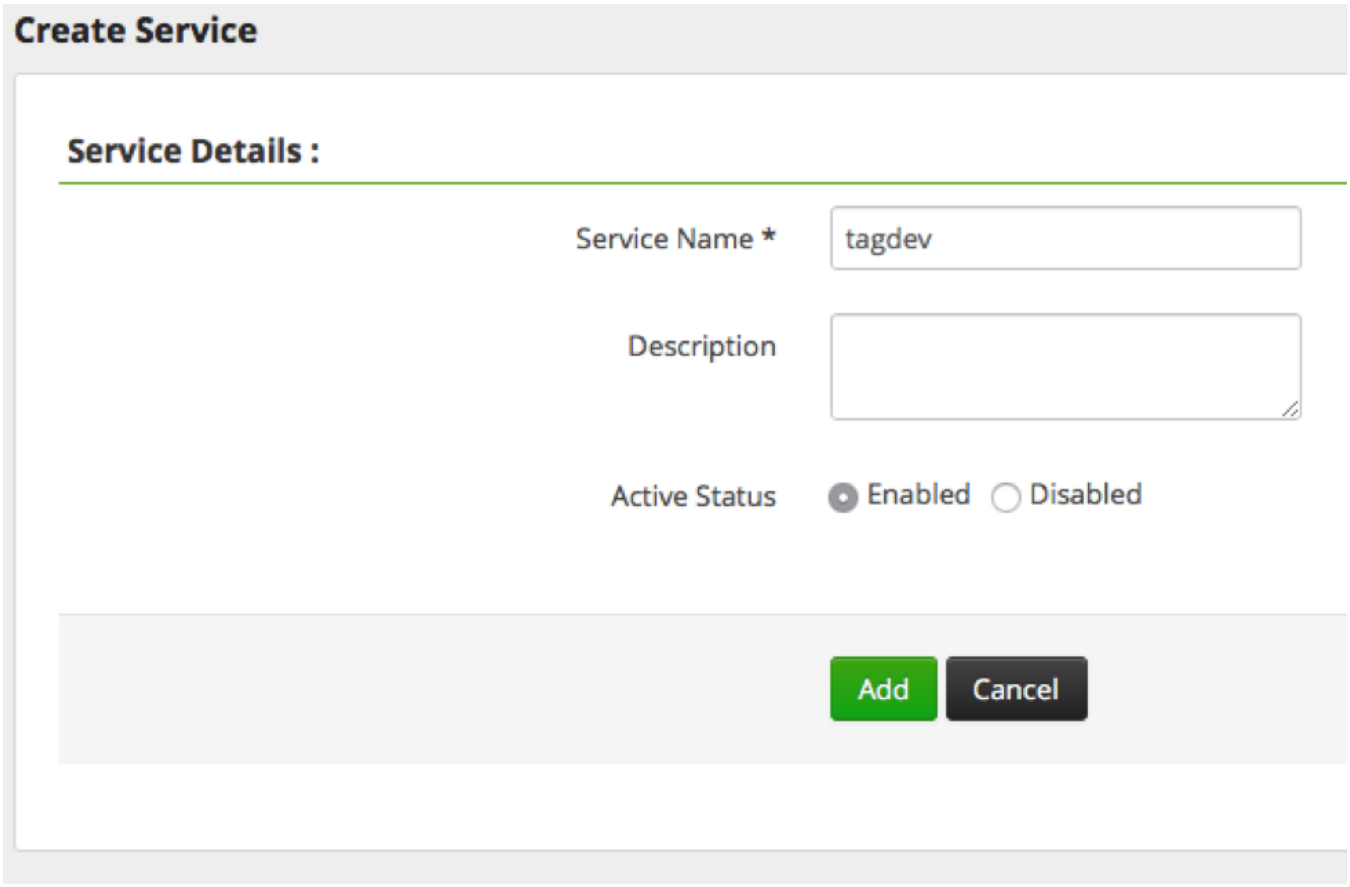
Usecases require Hive objects to be tagged as shown below. Please ensure to add these tags in Apache Atlas:

Objects	TAG	Tag attributes
Column: **finance.tax_2010.ssn** Column: **hr.employee.ssn**	PII	
Table: **finance.tax_2010**	EXPIRES_ON	expiry_date='2015/08/31'

8.3 Tag service

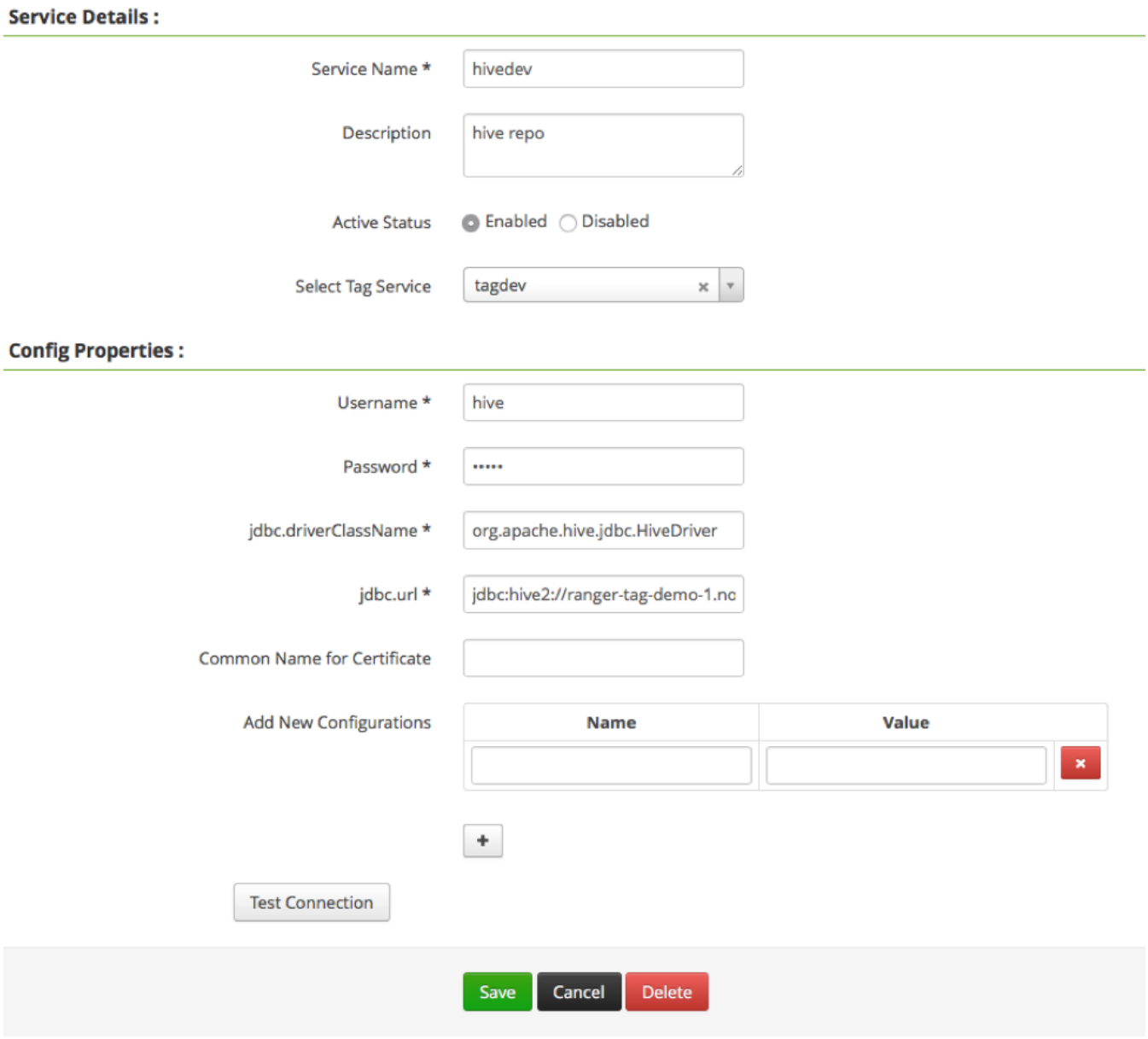
As mentioned earlier in this document, tag-based policies will be created in a tag service-instance. Follow the steps given below to create a tag service-instance named tagdev:

1. Login to Ranger Admin
2. Select menu: Access Manager è Tag Based Policies
3. Click the + icon next to TAG
4. In ‘Service Name’ field, enter tagdev and click ‘Add’



Component service instances, like hivedev, must be updated to enforce the tag-based policies available in a tag service instance. Follow the steps given below to update Hive service instance in your environment:

1. Login to Ranger Admin
2. Select menu: Access Manager è Resource Based Policies
3. Click on the edit icon next to your hive service instance, like hivedev,
4. In ‘Select Tag Service’ field, select tagdev and click ‘Save’

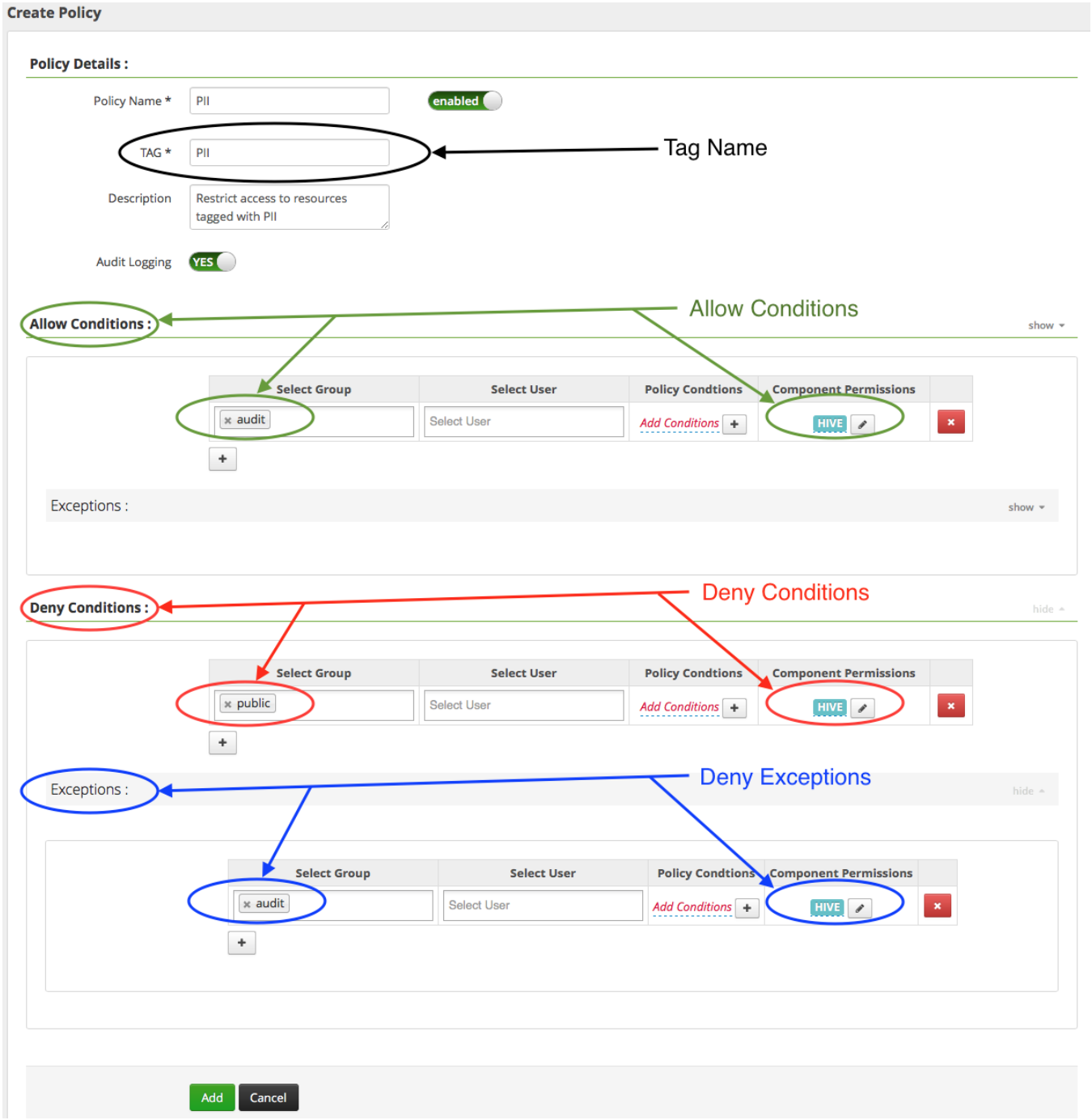


8.4 Tag-based policy: PII

Follow the steps give below to create a tag-based policy, which allows only audit group users to access objects tagged with PII. This policy will deny access to all other users.

1. Login to Ranger Admin
2. Select menu: Access Manager è Tag Based Policies
3. Select service instance tagdev
4. Click ‘Add New Policy’
5. In ‘Policy Name’ field, enter PII
6. In ‘TAG’ field, enter PII
7. In Allow conditions:
    a. In ‘Select Group’, pick audit
    b. In ‘Component Permissions’, enter hive as component name and pick ‘All’
8. In Deny conditions:
    a. In ‘Select Group’, pick public
    b. In ‘Component Permissions’, enter hive as component name and pick ‘All’
9. In Deny Exceptions:
    a. In ‘Select Group’, pick audit
    b. In ‘Component Permissions’, enter hive as component name and pick ‘All’
10. Click ‘Add’

Here is the screenshot of the tag-based policy for PII tag:



8.5 Tag-based policy: EXPIRES_ON

A policy for EXPIRES_ON tag is created automatically when a tag service instance, like tagdev, is created. This default policy denies access to objects tagged with EXPIRES_ON after the expiry date specified tag attribute. Please review the default policy with the following steps:

1. Login to Ranger Admin
2. Select menu: Access Manager è Tag Based Policies
3. Select service instance tagdev
4. Select policy ‘tagdev-EXPIRES_ON’

Here is the screenshot of the tag-based policy for EXPIRES-ON tag: