# HBase Plugin



## Introduction

This post explains details around configuring HBase plugin along with a few recommendations.

## Prerequisites

### Installation

- Install, configure and start Ranger Admin.
- Install, configure and start Ranger UserSync.
- In Ranger Admin, create a service instances (aka repository) for each Hadoop service that will have Ranger authorization - e.g. HDFS, Hive, HBase, ..
- Install Ranger plugin on the target Hadoop service, e.g. Hive, HBase, HDFS, etc. and restart the service.

If you are unsure about how to do the above then please refer to the following:

- Installing Ranger manually and Verifying Ranger installation.
- Installing Ranger using Ambari

### Verification and preparation for plugin

This document assumes that you have successfully performed the above steps. In addition, please ensure that you have done the following:

- Log into Ranger Admin with a user having admin role.
- Test the connectivity between Ranger Service and its target service. This isn't essential but it would ease the policy authoring process.
- Ensure that you can see users and their group mappings in Ranger Admin.

This document would further assume that you are familiar with Ranger specific terms like Service, Audit, etc. If you are unsure about these then please consult the introductory material about Ranger at the product wiki page.

## HBase Policy

Let us look at the specifics of HBase policies.

### HBase policy Resource

Hbase policies, like all ranger policies are specific to a resource. Resource is the primary target of authorization.

1. HBase policy resource can have following levels:
    a. HBase Table
    b. HBase Column Family
    c. HBase Column
2. Each resource field can take in multiple values. For example, for Table one could enter: `tab1`, `tab2`. As you start typing Policy Manager contacts the service for metadata and shows the valid table/column-family values. This helps during policy authoring to avoid misspelled resource names.
3. Include/exclude flag can be specified at each of these resource levels. Default is include. By turning the flag to exclude inverts the resource definition.

    a. For example, if you have a resource setup as follows: Table=`tbl1`, Column Family=`cf1`, Column=`col1`. and Column level is set to Exclude, then it means that means the resource is effectively referring to all columns of column family `cf1` of table `tbl1` except the column `col1`.
    b. Use excludes flag in resource definition when it makes simplifies the policy definition. Indiscriminate use of include/exclude flag can make reasoning about authorization challenging.
4. Auditing is specified at the resource level.

### Policy Item(s)

Each policy can have zero or multiple policy items.

- Policy item specify which sets of users or user-groups are allowed to perform what sort of operations on the policy resource.

Sometimes policy items can also specify other constraints about access, e.g. ip address from which access is to be granted, etc.

### Hbase Access Permissions

1. Hbase plugin support the following Permissions:
    a. Read
    b. Write
    c. Create
    d. Admin
2. A policy item can specify multiple permissions.

### Delegated administration

The Delegate Admin flag at policy item level can be used to delegate the administration responsibility for a policy to users or user-groups specified on that policy item.

1. This is a handy way to free the corporate administrator from having to deal with low level administration details that are best left to department level super-users.
2. If you check grant delegated admin flag at a policy level then those users and user-group members would be able to grant access privileges to other users at a resource level below the policies resource.
3. This feature isn't specific to HBase but it is common to all plugins.

### Audit specification

The policy can specify if access to the policy resource should be audited or not. Audit specification provides for aggregating the audit events such that similar events within a configurable timeframe would be logged as a single audit along with the total count. This can be particularly useful when audit volume is high.

## HBase Authorization with Apache Phoenix

Apache Phoenix is a popular relational database layer over HBase. Phoenix requires all access to tables that start with name `SYSTEM.`. To enable this access, please create a Ranger policy with the following values:

1. Table : `SYSTEM.*`
2. Column Family : *
3. Column : *
4. Groups : `public`
5. Permissions : `Read`, `Write`, `Create`, `Admin`

## Grant and revoke

HBase shell supports grant and revoke commands to tweak authorization. Those commands are mapped to ranger policies such that behind the scenes Ranger intercepts the grant/revoke calls and adds or edits ranger policies to mimic the resource and user or user-group specified in grant/revoke. This can be quite useful, say, if you have existing batch jobs or scripts that rely on grant and revoke features of HBase Shell to work correctly.

### Mechanics of grant/revoke

It is worth going to into a bit of detail about how HBase grant/revoke works with ranger because there are situations when its behaviors might be surprising and even undesirable.

1. User running grant or revoke should either have admin role in policy manager or have delegated admin privilege on the subject resource. This no different from when a user authors policies via the policy manager.
2. HBase shell would do checks to validate that table or column-family on which grant is being attempted do exist and would deny grant/revoke if the resources are absent. Whereas Ranger supports creation of policies for non-existent resources in anticipation of their creation at a future date.
3. Ranger looks for a policy that exactly matches the resources of grant/revoke. Due to this exact match policies can be in an unexpected state. Consider the following scenario.
    a. Say, a policy exists in Ranger that grants `read` access to user `user1` on tables `tab1` and `tab2`.
    b. Admin uses grant command to also give both `read` and `write` permission to `user1` on table `tab1`. The command admin uses would be: `grant 'user1', 'RW', 'tab1'`.

c. This would not only create a new policy with a single resource of `tab1` for user `user1` granting her both read and write permissions but leave the old policy as is.
d. Now let's say admin wants to revoke permissions for `user1` on table `tab1`. The command admin uses would be: `revoke 'user1', 'tab1'`.
e. This would remove the policy created during step # 3.b above but since it leaves the original policy for tables `tab1` and `tab2` as is, the end result is that user still has read permissions on the table `tab1`.
f. The root of this problem is that policies are being set both via Policy Manager and hbase shell. One way to avoid these problems is to restrict policy authoring to one system -- either do it via Ranger Policy Manager or via grant/revoke commands of hbase shell, not both. Another way is to disable policy authoring via grant revoke in the hbase plugin itself.

### Disabling policy edits via grant/revoke route

It is possible to restrict grant revoke commands from altering the Ranger policies. This is controlled via the following property in `ranger-hbase-security.xml`.

```
<property>
    <name>xasecure.hbase.update.xapolicies.on.grant.revoke</name>
    <value>true</value>
    <description>
        Should HBase plugin update Ranger policies for updates to permissions done using GRANT/REVOKE?
    </description>
</property>
```

## Frequently asked questions

You can do a lot with information provided above. However, the hbase plugin and Ranger's policy framework supports many other features for advanced functionality. This section lists those in the form of a FAQ.

### General questions about policies

1. Can I create two policies for the same resource?
    a. Yes, but repeating resources in a policy is discouraged since it can lead to a configuration that is hard to understand and reason about.
    b. Ranger policies are resource based and policy resources must be unique across a service. But since a Resource can have multiple values it is possible to create two different policies that share some common resource in them thereby having multiple policies for a single resource.
2. But looks like I do have two policies with exact same resource! How come?
    a. Resource uniqueness depends on the include/exclude flag value at each level. Changing the exclude flag materially alters the resource definition.
    b. In addition the recursive flag at resource level also determines policy resource uniqueness. For example, you can have two policies for same resources one with recursive turned on and one without. This provision exists to allow one to keep the auditing and authorization concerns separate. But this can make the policy set complex so it should be used with caution.
    c. In the end policies should be such that it is easy to reason about an authorization request by looking at the various policy values.
3. Can I create a policy just to do auditing?
    a. Yes. Create a policy for the right resource level where you want to do auditing and then turn auditing on. Don't define any policy items.
4. Can I have a policy without any policy items?
    a. Yes if the intention is to do just auditing.
5. Can I create a policy for a resource that does not exist yet?
    a. Yes, you can create a policy for a non-existent resource in anticipation of its creation.
6. Can I create a policy for users or user-groups that don't exist in Ranger yet?
    a. No. User or user-group must exist before they can be used in a policy item.
7. Can I administer the policies using a REST API?
    a. Yes. Please refer to Ranger wiki for details.
8. What is the "public" group? Where did it come from?
    a. This is an internal group that serves as a wildcard for all system users.

### HBase policies

1. Can I create a policy to control access by HBase row id?
    a. No. Ranger does not support row-level access control for HBase.
2. Is Ranger authorizer installed only on HBase Master?
    a. Ranger authorizer is installed on both HBase Master and all Regionservers.
3. Do I need to Install Ranger HBase Plugin on Region Servers, too?
    a. Yes. Ranger plugin must be deployed as a coprocessor on both Master and Regionservers.
4. We want to expand out our hbase cluster by adding new nodes. Do we need to do anything as far as Ranger is concerned.
    a. Yes. Make sure to install the ranger plugin on each of the new nodes. Some additional steps may be required if you are using wire encryption. Refer to the Ranger installation links mentioned in pre-requisites.
5. Why do you have two different types of admin privileges for Hbase policy? How does HBase Admin permission differ from Delegated Admin?
    a. Admin permission is HBase level permission which controls the ability to do administrative tasks on a table or column family such as balancing a cluster or reassigning regions.
    b. Whereas Delegated Admin is a flag that controls if Ranger Policy Manager would allow users to create or manage policies at a finer level.

    c. For example, a user with delegated admin privilege at Table=`tbl1`, Column-family=`cf1` can define new policies for a sub-set of columns of table `tbl1`, column-family `cf1`.
6. How does HBase do its authorization?
    a. Ranger plugin is deployed as a Coprocessor on both HBase Master and Region Server.
7. We have other Coprocessors installed on our HBase cluster. Will Ranger conflict with those?
    a. No it should not. HBase coprocessors are designed such that multiple of those can be installed.
    b. Depending on the type of your coprocessors you may have to review the order in which coprocessors are configured.
8. Will Apache Ranger change the memory profile of my Hbase master and region server processes?
    a. Yes but the change should be very minimal if at all.
9. How can I benchmark the additional memory footprint due to Ranger plugin?
    a. Run a test load with and without the ranger plugin and see any changes in the memory footprint.
    b. Ranger holds hbase policies in memory. In addition it buffers audit logs in memory which get flushed to audit store periodically. Neither of these is expected to be much.