Ranger Plugin Service (RPS)

[1. Introduction](#_ye64z2lkbj2g)

[2. Architecture](#_wlzogy6ibbyk)

[3. Design](#_y7wjok655utf)

[3.1 Resources and access types](#_oif6djowlesk)

[3.2 HAWQ Service Type Configurations](#_vgv3jbatkajh)

[3.3 Definition of HAWQ Service Type JSON](#_vwfyebc05tas)

[3.3 Ranger Authorizer](#_k4nv3efegyqc)

[3.4 Resource Lookup](#_vsngmhhg36ko)

[3.5 XML configuration files](#_lhzezolnyf62)

[3.6 Dependent JAR files](#_wvcmxifqvt3p)

[3.7 Deployment considerations and High Availability](#_to8l4e9oms34)

[4. RPS REST API Specification](#_a51z8xeak7u1)

[4.1 API Request](#_2niqtj9g7qy8)

[4.2 API Normal Response](#_he8txmt2dr15)

[4.3 API Error Response](#_45c5xv4zxs0i)

[5. Management Requirements](#_7mokih29dvr8)

[5.1 Register HAWQ Ranger JSON with the Ranger Admin](#_xrie7n8c8x6)

[5.2 Place ranger-service-hawq.jar into Ranger Classpath](#_iuoqt2oy15ac)

[5.3 Update ranger xml configuration files in RPS](#_rqz9chon6igb)

[5.4 Management Scripts](#_rhm2ullyg9pf)

[6. Questions and Answers](#_63sr8e14wjq)

# 1. Introduction

The Ranger Plugin Service (RPS) is a RESTFul service that provides integration between Apache HAWQ (incubating) and Apache Ranger.

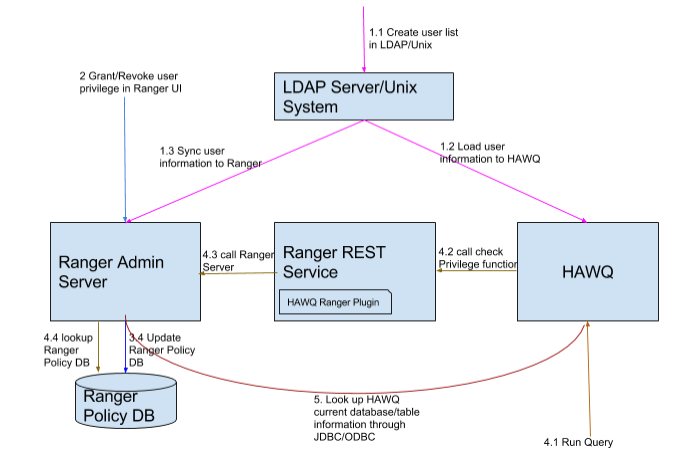
Apache Ranger offers a centralized security framework to manage fine-grained access control in Hadoop ecosystem. It provides security administrators with Admin portal where they can easily manage policies for access to files, folders, databases, tables, columns or other service-specific resources. These policies can be set for individual users or groups and then enforced by the services running in the Hadoop ecosystem.

For HAWQ to take advantage of Ranger, we must define a Ranger Plugin that is a Java library run as a part of the Java process. The plugin uses Ranger Plugin library to communicate to Ranger Admin server and to evaluate security policies.

The Ranger Plugin will be shipped as a jar file with embedded web server, thus creating a Ranger Plugin Service. HAWQ will use the REST endpoints exposed by the RPS to request evaluation of security policies whenever such a decision is required by HAWQ.

# 2. Architecture

The following picture illustrates how the Ranger Plugin Service facilitates integration between HAWQ and Ranger.



Ranger REST Service in the diagram above is now called Ranger Plugin Service (RPS) and contains HAWQ Ranger Plugin that extends RangerBasePlugin.

# 3. Design

The Ranger Wiki page documents what needs to be done to create a Ranger Plugin for a service: <https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=53741207>

In summary, the following items need to be addressed:

* Define HAWQ service resources, access types and configurations in a JSON file.
* Load the above JSON file into Ranger (using Ranger REST API).
* Implement Ranger Authorization Plugin that will provide:
  + init() method that is called during instance creation, initializes a policy engine with policies retrieved from the Ranger Admin Service and triggers a background thread to periodically update policies from the Ranger Admin Service
  + isAccessAllowed() method to obtain an access decision from the policy engine when an authorization decision is needed by HAWQ
  + an audit handler to submit audit information to the Ranger Admin Service
* Implement a subclass of RangerBaseService and provide implementation of lookupResource() and validateConfig() methods. Package the JAR file of this class and dependencies and place it into ranger-plugins/<service-type> directory in CLASSPATH of Ranger Admin.

## 

## 

## 

## 3.1 Resources and access types

The following are HAWQ resource and access types that can be applied to those resources (based on <http://hdb.docs.pivotal.io/201/hawq/reference/sql/GRANT.html>) :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | select | insert | update | delete | references | usage | create | connect | execute | temp |
| database |  |  |  |  |  |  | Y | Y |  | Y |
| tablespace |  |  |  |  |  |  | Y |  |  |  |
| schema |  |  |  |  |  | Y | Y |  |  |  |
| table | Y | Y | Y | Y | Y |  |  |  |  |  |
| sequence | Y |  | Y |  |  | Y |  |  |  |  |
| function |  |  |  |  |  |  |  |  | Y |  |
| language |  |  |  |  |  | Y |  |  |  |  |
| protocol | Y | Y |  |  |  |  |  |  |  |  |

Resources form the following hierarchy that will be used by Ranger UI when discovering them:

|--database

| |--schema

| | |--table

| | |--sequence

| | |--function

| |

| |--language

|

|--tablespace

|--protocol

To specify a resource, the values for all resource parents must be provided, for example:

{

"database": "finance",

"schema": "us",

"table": "sales"

}

describes sales table in us schema in finance database. Internally, the same resource might be represented as “finance/us/sales”.

Due to restrictions of Ranger, it is not possible to specify policies directly on non-leaf nodes of the resource hierarchy -- in our case database and schema. To get around this restriction, the following solution will be adopted:

* Additional access types will be defined: create-schema, usage-schema
* The policies will need to be defined on leaf resources, such as tables, with value set to “\*” and with the special permissions defined above, for example:
  + database=”A”/schema=”\*”/table=”\*” create-schema -- to allow creation of schemas in database A
  + database=”A”/schema=”B”/table=”\*” usage-schema -- to allow usage of schema B in database A

## 3.2 HAWQ Service Type Configurations

There are a number of configuration values that Ranger Admin UI will need to be able to connect to HAWQ Master and discover available resources. These configuration parameters are defined in HAWQ Service Type JSON file. A cluster administrator will define values for these parameters when enabling HAWQ as a supported service in Ranger.

The following parameters are required:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Label | Default |
| username | string | HAWQ User Name | gpadmin |
| password | string | HAWQ User Password |  |
| hostname | string | HAWQ Master Host |  |
| port | int | HAWQ Master Port | 5432 |

## 3.3 Definition of HAWQ Service Type JSON

This file defines resources of a service, along with other details like type of resource accesses (read/write/create/delete/submit/…), configuration needed to connect to the service (url, username, password, …) , custom conditions to evaluate in policies (IP range, …), etc.

Examples of service type definitions are here: <https://github.com/apache/incubator-ranger/tree/master/agents-common/src/main/resources/service-defs>

{

"name": "hawq"

"implClass": "org.apache.hawq.ranger.service.RangerServiceHawq",

"label": "HAWQ",

"description": "HAWQ",

"guid": "5b710438-edcf-4e20-834c-a9a267b5b963",

"resources":

[

{

"itemId": 1,

"name": "database",

"type": "string",

"level": 10,

"parent": "",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Database",

"description": "HAWQ Database"

},

{

"itemId": 2,

"name": "schema",

"type": "string",

"level": 20,

"parent": "database",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Schema",

"description": "HAWQ Schema"

},

{

"itemId": 3,

"name": "table",

"type": "string",

"level": 30,

"parent": "schema",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Table",

"description": "HAWQ Table"

},

{

"itemId": 4,

"name": "sequence",

"type": "string",

"level": 30,

"parent": "schema",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Sequence",

"description": "HAWQ Sequence"

},

{

"itemId": 5,

"name": "function",

"type": "string",

"level": 30,

"parent": "schema",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Function",

"description": "HAWQ Function"

},

{

"itemId": 6,

"name": "language",

"type": "string",

"level": 20,

"parent": "database",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Language",

"description": "HAWQ Language"

},

{

"itemId": 7,

"name": "tablespace",

"type": "string",

"level": 10,

"parent": "",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Tablespace",

"description": "HAWQ Tablespace"

},

{

"itemId": 8,

"name": "protocol",

"type": "string",

"level": 10,

"parent": "",

"mandatory": **true**,

"lookupSupported": **true**,

"recursiveSupported": **false**,

"excludesSupported": **true**,

"Matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",

"matcherOptions": {"wildCard":**true**, "ignoreCase":**true**},

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "Protocol",

"description": "HAWQ Protocol"

}

],

"accessTypes":

[

{

"itemId": 1,

"name": "select",

"label": "select"

},

{

"itemId": 2,

"name": "insert",

"label": "insert"

},

{

"itemId": 3,

"name": "update",

"label": "update"

},

{

"itemId": 4,

"name": "delete",

"label": "delete"

},

{

"itemId": 5,

"name": "references",

"label": "references"

},

{

"itemId": 6,

"name": "usage",

"label": "usage"

},

{

"itemId": 7,

"name": "create",

"label": "create"

},

{

"itemId": 8,

"name": "connect",

"label": "connect"

},

{

"itemId": 9,

"name": "execute",

"label": "execute"

},

{

"itemId": 10,

"name": "temp",

"label": "temp"

},

{

"itemId": 11,

"name": "all",

"label": "All",

"impliedGrants": [

"select",

"insert",

"update",

"delete",

"references",

"usage",

"create",

"connect",

"execute",

"temp"

]

}

],

"configs":

[

{

"itemId": 1,

"name": "username",

"type": "string",

"mandatory": **true**,

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "HAWQ User Name",

"defaultValue": "gpadmin"

},

{

"itemId": 2,

"name": "password",

"type": "password",

"mandatory": **true**,

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "HAWQ User Password"

},

{

"itemId": 3,

"name": "hostname",

"type": "string",

"mandatory": **true**,

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "HAWQ Master Hostname"

},

{

"itemId": 4,

"name": "port",

"type": "int",

"mandatory": **true**,

"validationRegEx": "",

"validationMessage": "",

"uiHint": "",

"label": "HAWQ Master Port",

"defaultValue": 5432

}

],

"enums": [],

"contextEnrichers": [],

"policyConditions": [],

"dataMaskDef": {},

"rowFilterDef": {}

}

## 3.3 Ranger Authorizer

The Authorizer is a Java class that receives an access verification request from HAWQ and uses Ranger Plugin APIs to come up with the access decision.

A static/global instance of RangerBasePlugin class should be created and initialized during the initialization of the service. The reference to the instantiated plugin object should be preserved for later reference during authorization of access requests.

During initialization, the plugin will load policies from local cache, if exists, and setup a policy refresher to get the updated policies from Ranger admin.

public class RangerHawqAuthorizer {

private static RangerBasePlugin plugin = null;

public void init(Configuration conf) {

plugin = new RangerBasePlugin("hawq", "hawq");

plugin.init(); // initialize policy engine and policy refresher

plugin.setDefaultAuditHandler(new RangerDefaultAuditHandler());

}

public boolean checkPermission(entity, accessType, userInfo) {

RangerAccessRequestImpl request = new RangerAccessRequestImpl();

RangerResourceImpl resource = new RangerResourceImpl();

resource.setValue(entity.getType(), entity.getName());

request.setResource(resource);

request.setAccessType(getRangerAccessType(accessType));

request.setUser(userInfo.getShortUserName());

request.setUserGroups(Sets.newHashSet(userInfo.getGroupNames()));

request.setAccessTime(new Date());

request.setClientIPAddress(getRemoteIp());

RangerAccessResult result = plugin.isAccessAllowed(request);

return result == null ? false : result.getIsAllowed();

}

}

The full set of properties that can be set on the RangerAccessRequest can be seen here: <https://github.com/apache/incubator-ranger/blob/master/agents-common/src/main/java/org/apache/ranger/plugin/policyengine/RangerAccessRequestImpl.java>

## 3.4 Resource Lookup

When authoring policies in Ranger admin, the users enter the name of the resources whose access need to be protected. To make it easier for users to enter the resource names, Ranger admin provides autocomplete feature, which looks up the available resources in the service that match the input entered so far.

To facilitate the autocomplete feature, Ranger Admin requires the plugin to provide the implementation of RangerBaseService interface. The implementation class should be registered with Ranger in service type definition and be made available in the CLASSPATH of Ranger Admin.

public class RangerServiceHawq extends RangerBaseService {

public HashMap<String, Object> validateConfig() throws Exception {

// TODO: connect to HAWQ Master; throw Exception on failure

}

public List<String> lookupResource(ResourceLookupContext context) throws Exception {

// TODO: retrieve the resource list from HAWQ Master via JDBC

}

}

## 3.5 XML configuration files

The Ranger Base Plugin expects to find its configuration in specific XML configuration files, that will need to become part of the RPS definition. These files need to be on the CLASSPATH of the RPS service.

The following are the files and expected parameters:

* ranger-hawq-audit.xml
* ranger-hawq-security.xml
* ranger-policymgr-ssl.xml

## 3.6 Dependent JAR files

The Ranger Base Plugin needs the following JAR files to be available on the CLASSPATH of the RPS service:

* ranger-plugins-audit-<version>.jar
* ranger-plugins-common-<version>.jar
* ranger-plugins-cred-<version>.jar

The above JAR files need to be added to the RPS classpath, which will also require co-location of RPS and Ranger Client on the same host.

## 3.7 Deployment considerations and High Availability

RPS will be deployed and running on HAWQ Master host (both Active and Standby). This will provide High-Availability of the RPS service and eliminate a single-point-of-failure. HAWQ engine will need to be able to fallback on calling RPS on the host where Standby Master is deployed if the RPS on the local host is not reachable (process down).

Cluster administrator will configure ranger integration by providing a value of true to the property enable\_ranger in hawq-site.xml.It is not expected that RPS will validate the value for the property, but the management tool responsible for starting the service should make sure the Ranger integration is turned on before starting the service.

# 4. RPS REST API Specification

The Ranger Plugin Service will provide REST APIs to HAWQ to request evaluation of access policies. The API is designed to support a batch request, where access to multiple resources might be requested at once so that HAWQ doesn’t have to call the REST API multiple times for a single query.

## 4.1 API Request

POST /authorize

{

"requestId": 1,

"user": "joe",

"groups": ["admin","us"],

"clientIp": "123.0.0.21",

"context": "SELECT \* FROM sales",

"access":

[

{

"resource":

{

"database": "finance"

},

"privileges": ["connect"]

},

{

"resource":

{

"database": "finance",

"schema": "us",

"table": "sales"

},

"privileges": ["select, insert"]

}

]

}

## 4.2 API Normal Response

{

"requestId": 1,

"access":

[

{

"resource":

{

"database": "finance"

},

"privileges": ["connect"]

"allowed": true

},

{

"resource":

{

"database": "finance",

"schema": "us",

"table": "sales"

},

"privileges": ["select, insert"]

"allowed": false

}

]

}

Privileges will be evaluated as ALL, meaning all of them must be granted to the user in order for the access to be allowed. The decision is a cumulative one, the response does not detail which particular privilege was declined.

If a requested resource is not defined in Ranger, then access will be denied, since we should be using a security model of “denied unless allowed”.

## 4.3 API Error Response

In case the RPS can not fulfill authorization request from HAWQ, an HTTP error will be returned. The following is the list of error responses:

|  |  |
| --- | --- |
| HTTP 400 | Submitted request is incorrect. Should not happen during runtime. |
| HTTP 500 | If the RPS encountered an unhandled exception. |

The error message returned by the RPS will have a JSON payload containing the HTTP error status and the message, for example:

{

"status": 400,

"message": "resource value is missing for key=database in the request"

}

The actual exception with the stack trace will be logged into an RPS log file.

# 

# 5. Management Requirements

There are a few steps that a cluster administrator needs to perform to enable HAWQ-Ranger integration. If the cluster is administered by Ambari, then HAWQ Ambari Plugin needs to be extended to support this functionality.

## 5.1 Register HAWQ Ranger JSON with the Ranger Admin

HAWQ service type definition JSON file must be registered with Ranger using REST API provided by Ranger Admin. Once registered, the cluster administrator can use Ranger UI to create service instances and access policies for HAWQ.

The REST API can be invoked using curl command as shown in the example below:

curl -u admin:admin -X POST -H "Accept: application/json" -H "Content-Type: application/json" –d @ranger-servicedef-hawq.json http://ranger-admin-host:port/service/plugins/definitions

## 5.2 Place ranger-service-hawq.jar into Ranger Classpath

The RangerServiceHawq class with required dependencies and the Postgres JDBC driver need to be packaged into a JAR file ranger-service-hawq.jar. This JAR file and Postgres JDBC driver JAR file need to be placed into ranger-plugins/hawq directory in CLASSPATH

of Ranger Admin.

## 5.3 Update ranger xml configuration files in RPS

The RPS needs to be able to configure itself so that it can connect to Ranger Admin and read values for related properties. These properties are contained in ranger-hawq-security.xml file that will become part of RPS installation:

|  |  |  |
| --- | --- | --- |
| Configuration | Default Value | Comments |
| ranger.plugin.hawq.service.name | hawq | Name of the service containing policies for the plugin |
| ranger.plugin.hawq.policy.source.impl | org.apache.ranger.admin.client.RangerAdminRESTClient | Name of the class used to retrieve policies. |
| ranger.plugin.hawq.policy.rest.url | **No default value.** | URL to Ranger Admin |
| ranger.plugin.hawq.policy.rest.ssl.config.file | **No default value.**  This configuration must be provided if SSL is enabled between plugin and Ranger Admin. | Path to the file containing SSL details to contact Ranger Admin |
| ranger.plugin.hawq.policy.cache.dir | /var/rps ? OR  /etc/ranger/hawq/policycache  **No default value.**  If no valid value is specified, local caching of policies will not be done. | Directory where Ranger policies are cached after successful retrieval from the source |
| ranger.plugin.hawq.policy.pollIntervalMs | 30000 | How often to poll for changes in policies? |

At least the value of ranger.plugin.hawq.policy.rest.url property needs to be updated with the valid URL where Ranger Admin is located.

## 5.4 Management Scripts

RPS will provide scripts to start and stop the service. The scripts themselves will not check whether the service should be running or not., assuming the check has been perfromed by the tools invoking the script.

It is possible to have “hawq start master” and “hawq stop master” HAWQ commands to check the value of the property in hawq-site.xml and start RPS, if needed, in addition to starting postgres processes. This will make RPS existence and management transparent to users and will make service lifecycle easier for the administrators.