[Prop Data Multi-Tenant Implementation Using S3 and Federation Query Engine](https://confluence.marketintelligence.spglobal.com/display/TPM/Prop+Data+Multi-Tenant+Implementation+Using+S3+and+Federation+Query+Engine)

* Created by Unknown User (kiran.sahoo2@spglobal.com), last modified on [Oct 12, 2023](https://confluence.marketintelligence.spglobal.com/pages/diffpagesbyversion.action?pageId=168821135&selectedPageVersions=29&selectedPageVersions=30)

**PropData Implementation**

**Prop data platform will take a metadata driven approach  and will provide necessary hooks/ interfaces for extending/modifying the base functionalities.**

**Prop Channels**- Prop Data channels for storing/retrieving prop data will be  designed to be extendable and will support the following channels as part of initial phases

* API
* GraphQL
* Embeddable SDK

**Prop Auth Service**- Is an extendable authentication module

* **IAMABACAuthService**- to use ABAC authorization patterns of  **AWS IAM .**The implementation will use  **session tags**, a feature of IAM and AWS Security Token Service (STS), to implement ABAC through tags passed to an AWS IAM Role. **A Custom Fine grained policy Engine** to be applied on top of the data returned from the API calls.

**Prop Data Connectors**- To be built to support connection to various data stores for read/write using

* AWS Boto SDK
* Spark APIs
* Custom Connectors using Eclipse Microprofile or Similar Frameworks
* Athena Federation Engine

Future Phases may include support for

* Eclipse Data Connectors

**Prop Data Formats**-  Prop data formats will have support for.

* **Full Schema Evolution**
* **Partitioning**
* **Time Travel and Versioning**
* **Compression**
* **Mutation**

**Prop Data Source**-First phase to  support the following data sources for storing prop data

* S3 (Queryable via Redshift Spectrum/Athena/Snowflake)
* MS SQL

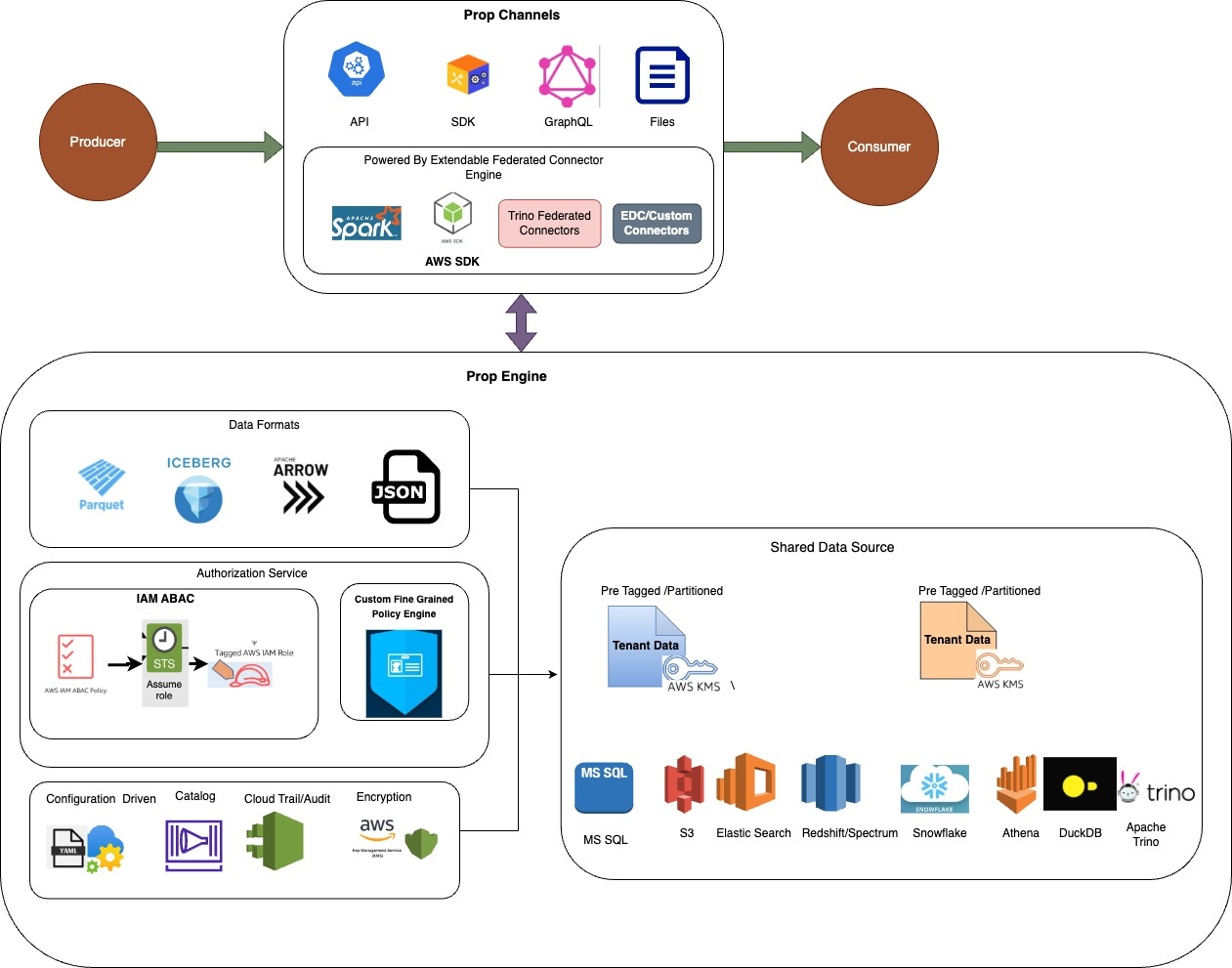
**Prop Data Catalog:**- To leverage Glue Data Catalog to store and retrieve table metadata for the Amazon S3 data

**Encryption**-  AWS KMS to be leveraged for encryption objects in S3. Each Tenant can bring their own key or can be assigned a key. Tenant will be granted access to S3 objects only if the tag-key and tag-values on the **AWS IAM Role** attached to it match the specific tag-key and tag-values on the Amazon S3 object **AND** the corresponding **AWS KMS Key** used to encrypt the S3 object.

**Auditing and Governance**-

* **AWS CloudTrail and  DataDog to** be used for monitoring/auditing to ensure Compliance  and Governance
* **Secure  tags using an AWS Organizations service control policy -**Because the system will be relying on tags for authorization to data,  limiting who is able to tag data on Amazon S3 and your IAM principals will be handled through service control policy (SCP) . The SCP will prevents setting tags on s3 data and AWS IAM principals.

**Implementation Diagram**



**PropData Authentication/Authorization Service.**

ABAC mechanisms to be used to gain fine-grained access to S3 objects/Data  **AND** an AWS KMS Key. The prop application will be granted access to S3 objects/Data ONLY if the tag-key and tag-values on the **AWS IAM Role** attached to it match the specific tag-key and tag-values on the Amazon S3 object **AND** the corresponding **AWS KMS Key** used to encrypt the S3 object.

An **AWS IAM Policy** will be added to the **AWS IAM Role** attached to the Prop application and this **AWS IAM Role** will be tagged with the same tag-key and tag-values as seen on the Amazon S3 object/Data **AND** the corresponding AWS KMS Key.

A **condition block** in the **AWS IAM Policy** JSON document enforces this authorization by matching the tag-key and tag-values defined on the Amazon S3 object/ Data and the AWS KMS Key to the tag-key and tag-values defined on the AWS IAM Role.

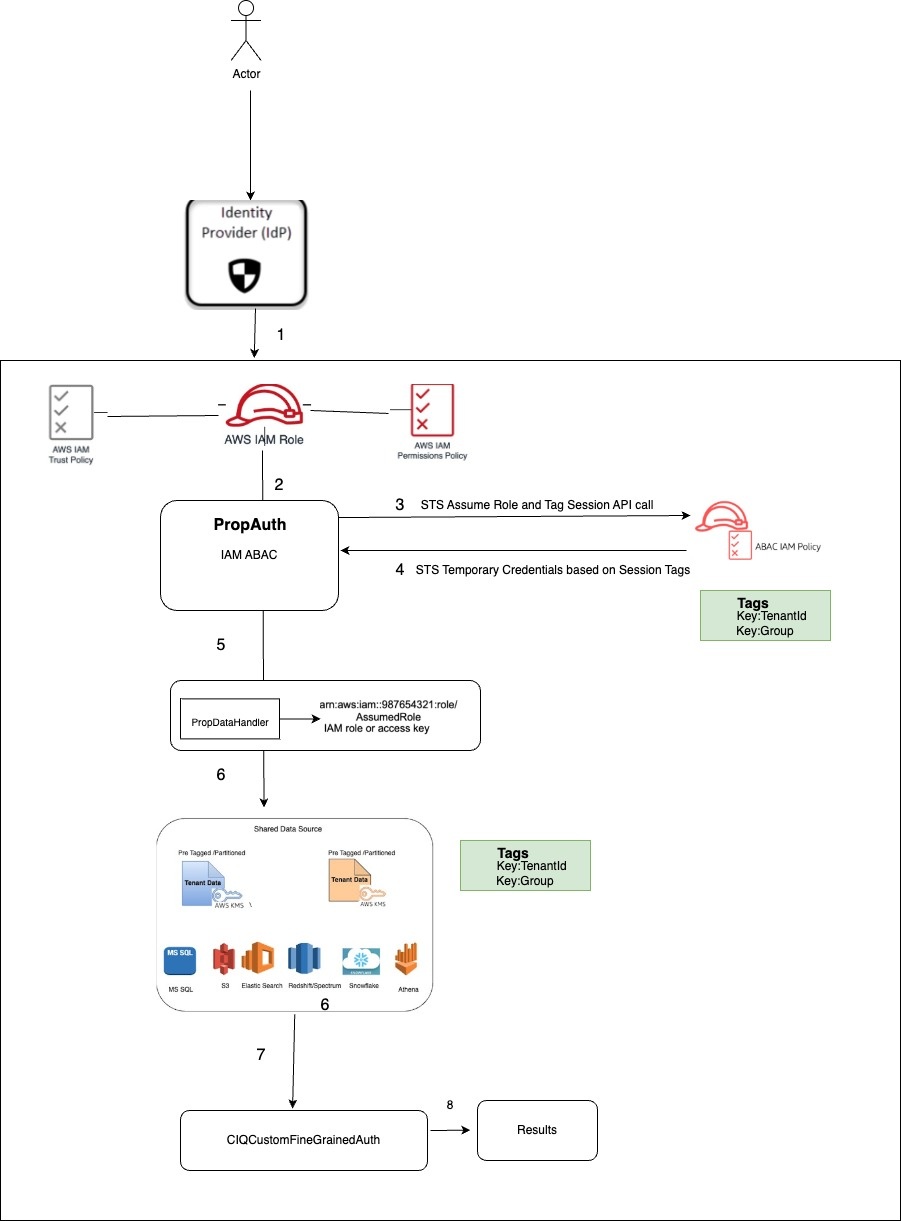
By adopting this mechanism, it will allow one to use a single AWS IAM Role that will grant the defined access  to single or multiple AWS resources. Additionally, it allows tenants to grow with fewer changes to AWS IAM policies.

ABAC policies are more flexible than traditional AWS IAM policies, which require to list each individual resource.

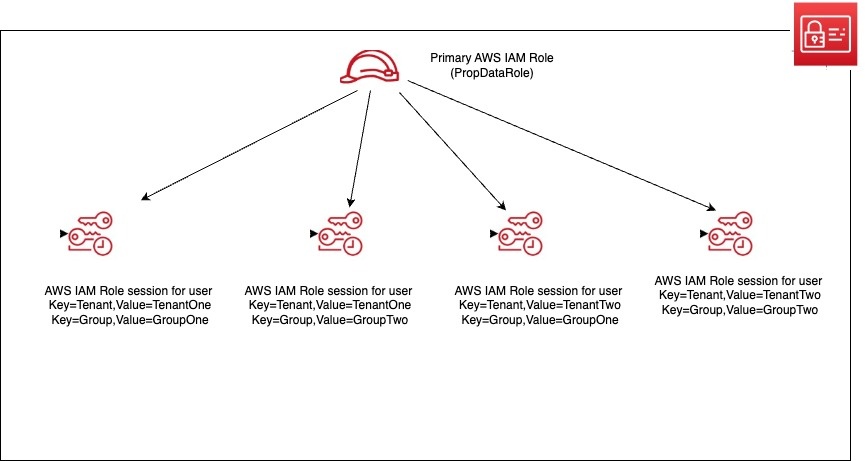
The code  will use  session tags, a feature of IAM and AWS Security Token Service (STS), to implement ABAC through tags passed to an **AWS IAM Role**.

Session tags are key-value pair attributes to be passed to assume an **AWS IAM role** in **AWS STS**. This can be done by making AWS API request through AWS STS.  AWS STS generates temporary  security credentials and a session. Sessions expire and have credentials, such as an access key pair and a session token. Session credentials will be used for  subsequent request to access AWS resource, the request context also includes the **aws:PrincipalTag** context key. **aws:PrincipalTag** key in the Condition element of **AWS IAM policy** to allow or deny access to AWS resources based on those tags.

The diagram below shows the authz/authn flow



As seen in the below diagram there is only a single IAM role deployed in AWS (PropData), but the assigned session tags are unique to each user assuming the role



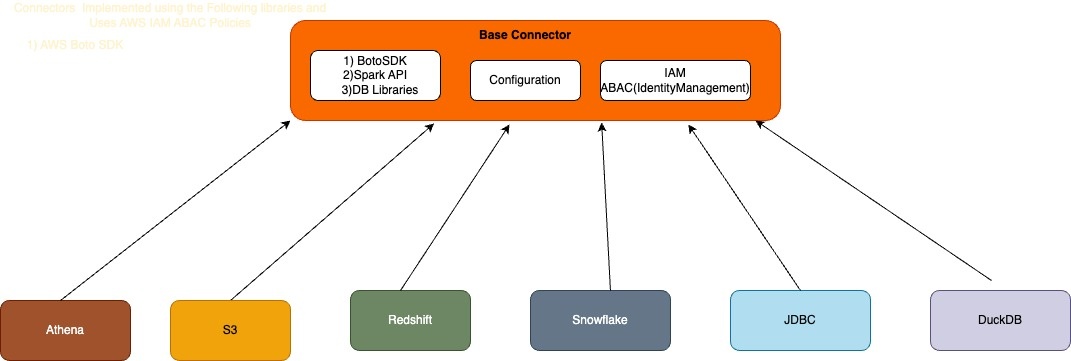
**Prop Connectors Implementation**

**Build a interactive query engine to federate multiple data sources together. With the ability to query multple workloads, It becomes a single point of access and allows access to  data wherever it lives.**

1) Base  connectors to use IAM ABAC for Authorization/Authentication

2) Connector module will be metadata driven.

3) Connector module to expose read/write methods  and will leverage a combination of following SDKS(**Boto,Spark,DB Libraries or Athena Federation Engine, Trino Federation Connectors**) to read and write .

****

**Prop Data Model**

**TBD**

**Prop API/Schema Specification**

**TBD**