Cloudera Runtime 7.3.2

Schema Registry Reference

Date published: 2019-08-22
Date modified: 2026-03-31

https://docs.cloudera.com/

Legal Notice

Contents

SchemaRegistryClient properties reference

Review the following reference for a comprehensive list of the SchemaRegistryClient properties.

Table 1: SchemaRegistryClient properties

Property Name	Description	Type	Default Value
schema.registry.url	The URL of the Schema Registry server which this client connects to.	String	http://localhost:9090/api/v1
schema.registry.client.local.jars.path	The local directory path to which downloaded JARs are copied to.	String	/tmp/schema-registry/local-jars
schema.registry.client.class.loader.cache.size	The maximum size of the classloader cache.	Int	1024
schema.registry.client.class.loader.cache.expiry.interval.secs	The expiry interval (in seconds) of an entry in the classloader cache.	Int	3600 sec
schema.registry.client.schema.version.cache.size	The maximum size of schema text cache.	Int	1024
schema.registry.client.schema.version.cache.expiry.interval.secs	The expiry interval (in seconds) of an entry in schema version cache.	Int	300 sec
schema.registry.client.schema.metadata.cache.size	Maximum size of schema metadata cache.	Int	1024
schema.registry.client.schema.metadata.cache.expiry.interval.secs	Expiry interval (in seconds) of an entry in schema metadata cache.	Int	300 sec
schema.registry.client.schema.text.cache.size	Maximum size of schema text cache.	Int	1024
schema.registry.client.schema.text.cache.expiry.interval.secs	Expiry interval (in seconds) of an entry in schema text cache.	Int	300 sec
schema.registry.client.url.selector	Schema Registry URL selector class.	String	com.hortonworks.registries.schemaregistry.client.Fail
sasl.jaas.config	Schema Registry Dynamic JAAS configuration for SASL connection.	String	null
schema.registry.auth.username	Username for basic authentication.	String	null
schema.registry.auth.password	Password for basic authentication.	String	null
schema.registry.hash.function	Hashing algorithm for generating schema fingerprints.	String	MD5
schema.registry.auth.type	The type of authentication the client should use. If the value is oauth2, it will be configured for oauth login, otherwise it will use Kerberos.	String	kerberos
schema.registry.oauth.client.id	Client ID for OAuth server in case of oauth login.	String	empty
schema.registry.oauth.secret	Secret for OAuth server in case of oauth login.	String	empty
schema.registry.oauth.server.url	OAuth server URL in case of oauth login.	String	empty
schema.registry.oauth.scope	OAuth scope in case of oauth login.	String	empty

Property Name	Description	Type	Default Value
schema.registry.oauth.request.method	HTTP method for requesting the oauth token.	String	post
connector.provider.class	Classname of a Jersey connector provider. (For example: org.glassfish.jersey.apache.connector.ApacheConnectorProvider) Make sure the class is on classpath. If this is set, Backoff policy might need to be set to com.hortonworks.registries.schemaregistry.retry.policy.ExponentialBackoffPolicy.	String	empty
schema.registry.client.ssl	Schema Registry SSL configuration. Expects a map of SSL properties. For available properties, see SSL configuration properties on page 5.	Object	null

schema.registry.client.retry.policy	Expects a map containing the retry policy configuration. The map accepts a className key to specify the policy class and a config key for policy-specific properties. Both keys are optional. className defaults to `ExponentialBackoffPolicy` if omitted, and config uses policy defaults if omitted. For available policies and properties, see Retry policies and retry policy configuration properties on page 7.	Object	`ExponentialBackoffPolicy` with the following default properties:
			`sleepTimeMs=1000`
			`maxAttempts=5`
			`timeoutMs=90000`

schema.registry.client.doAs	The user principal to send requests on behalf of. When configured, the principal name is included in the doAs query parameter for every request.	String	empty

SSL configuration properties

The schema.registry.client.ssl property expects a map of SSL properties that control SSL/TLS connections between the Schema Registry client and server.

Example configuration in YAML

The following example shows a typical mTLS configuration with both truststore properties (for validating the server) and keystore properties (for client authentication):

```
#...
schema.registry.client.ssl:
  protocol: [***PROTOCOL NAME***]
  trustStoreType: [***STORE TYPE***]
  trustStorePath: [***TRUSTSTORE PATH***]
  trustStorePassword: [***TRUSTSTORE PASSWORD***]
  keyStoreType: [***STORE TYPE***]
  keyStorePath: [***KEYSTORE PATH***]
  keyStorePassword: [***KEYSTORE PASSWORD***]
```

Table 2: SchemaRegistryClient SSL configurations properties

Property Name	Description	Type	Default Value
protocol	The SSL/TLS protocol name used to create the SSLContext. Using generic protocol names like "SSL" or "TLS" allows the JVM to negotiate the best available protocol version. Specifying a specific version restricts the connection to that protocol.	String	null (uses JVM default)
hostnameVerifierClass	Fully qualified class name of a custom HostnameVerifier implementation for validating server hostnames during the SSL handshake.	String	null (uses default HTTPS hostname verification)
keyStoreType	Format of the client keystore file.	String	null (uses JVM default, typically "JKS")
keyStorePath	Absolute or relative file path to the keystore containing the client's private key and certificate. Required for mutual TLS (mTLS) authentication.	String	null (client authentication disabled)
keyStorePassword	Password required to access the keystore file specified in keyStore Path.	String	null
keyPassword	Password for the client's private key within the keystore. Only needed if different from keyStore Password.	String	null (uses keyStorePassword)
keyStoreProvider	Name of the Java security provider for the keystore.	String	null (uses default provider)
keyManagerFactoryAlgorithm	Algorithm used by the `KeyManagerFactory` for managing client authentication credentials. This is an advanced property that rarely needs to be configured.	String	null (uses default algorithm)
keyManagerFactoryProvider	Provider name for the `KeyManagerFactory` for managing client authentication credentials. This is an advanced property that rarely needs to be configured.	String	null (uses default provider)
trustStoreType	Format of the truststore file. Required for validating the server's certificate.	String	null (uses JVM default)
trustStorePath	Absolute or relative file path to the truststore containing trusted CA certificates for validating the server's SSL certificate.	String	null (uses JVM's default cacerts)
trustStorePassword	Password required to access the truststore file specified in trustSto rePath.	String	null
trustStoreProvider	Name of the Java security provider for the truststore.	String	null (uses default provider)

Property Name	Description	Type	Default Value
trustManagerFactoryAlgorithm	Algorithm used by the `TrustManagerFactory` for managing server certificate validation. This is an advanced property that rarely needs to be configured.	String	null (uses default algorithm)
trustManagerFactoryProvider	Provider name for the `TrustManagerFactory` for managing server certificate validation. This is an advanced property that rarely needs to be configured.	String	null (uses default provider)

Retry policies and retry policy configuration properties

The schema.registry.client.retry.policy property expects a map containing the retry policy configuration. The map accepts a className key to specify the policy class and a config key for policy-specific properties. Both keys are optional. className defaults to `ExponentialBackoffPolicy` if omitted, and config uses policy defaults if omitted. The retry policy controls how the client handles failed requests to the Schema Registry server.

The following retry policies are available:

- `com.hortonworks.registries.schemaregistry.retry.policy.ExponentialBackoffPolicy` – The default policy. Retries with exponentially increasing delays between attempts.
- `com.hortonworks.registries.schemaregistry.retry.policy.FixedTimeBackoffPolicy` – Retries with a fixed delay between attempts.
- `com.hortonworks.registries.schemaregistry.retry.policy.NOOPBackoffPolicy` – No retry attempts are made.

Example configuration in YAML

```
#...
schema.registry.client.retry.policy:
  className: [***POLICY CLASS***]
  config:
    sleepTimeMs: [***MILLISECONDS***]
    maxAttempts: [***ATTEMPT COUNT***]
    timeoutMs: [***TIMEOUT MILLISECONDS***]
```

Table 3: SchemaRegistryClient retry policy configuration properties

Property Name	Description	Type	Default Value
sleepTimeMs	The initial delay in milliseconds between retry attempts. For `ExponentialBackoffPolicy`, this value increases exponentially with each retry.	Long	`ExponentialBackoffPolicy` – 1000
			`FixedTimeBackoffPolicy` – 1000
			`NOOPBackoffPolicy` – 0
maxAttempts	The maximum number of retry attempts before failing the request.	Int	`ExponentialBackoffPolicy` – 5
			`FixedTimeBackoffPolicy` – 5
			`NOOPBackoffPolicy` – 1

Property Name	Description	Type	Default Value
timeoutMs	The total timeout in milliseconds for all retry attempts combined. The client will stop retrying once this timeout is exceeded, even if maxAttempts has not been reached.	Long	`ExponentialBackoffPolicy` – 90000 `FixedTimeBackoffPolicy` – 60000 `NOOPBackoffPolicy` – 0

Related Information

Compatibility policies

KafkaAvroSerializer properties reference

Review the following reference for a comprehensive list of the KafkaAvroSerializer properties.

Property Name	Description	Type	Default Value
schema.compatibility	The default compatibility of the new schemas created by the serializer. For more information about the compatibility options, see the *Compatibility policies* documentation.	String	BACKWARD
schema.group	The default group of the new schemas created by the serializer.	String	kafka
schema.name.key.suffix	The provided string will be appended to the end of the schema name created for record keys. The schema name is the topic name initially, so as a result, you will have topic name + provided suffix as schema name.	String	:k
schema.name.value.suffix	The provided string will be appended to the end of the schema name created for record values. The schema name is the topic name initially, so as a result, you will have topic name + provided suffix as schema name.	String	null
null.passthrough.enabled	If this is enabled, the serializer will return immediately with null value, without putting anything in the serialized output. Having null values can be important in compact topics. Having null as the latest value for a key indicates for Kafka that the corresponding key can be deleted during log compaction.	Boolean	false

Property Name	Description	Type	Default Value
slogical.type.conversion.enabled	If this is enabled, Avro record fields can be serialized that are instances of Java classes (BigDecimal, UUID, LocalDate, and so on) that can be serialized in Avro using logical types. For more information, see the Logical Types section in the official Avro documentation. This does not have to be enabled if SpecificRecords are used as part of generated classes. SpecificDatumWriter is used for serializing SpecificRecord instances and that writer handles logical types automatically.	Boolean	false
serdes.protocol.version	The protocol version to be used. The available default registered protocols are the following: • CONFLUENT_VERSION_PROTOCOL = 0x0; • METADATA_ID_VERSION_PROTOCOL = 0x1; • VERSION_ID_AS_LONG_PROTOCOL = 0x2; • VERSION_ID_AS_INT_PROTOCOL = 0x3; • JSON_PROTOCOL = 0x4;	Byte	0x3 (VERSION_ID_AS_INT_PROTOCOL)
store.schema.version.id.in.header	If this is set to true, the schema version ID will be stored in the record header instead of the value.	Boolean	false
key_schema_version_id_header_name	If the schema of the record key is stored in the header, the header name can be configured.	String	key.schema.version.id
value_schema_version_id_header_name	If the schema of the record value is stored in the header, the header name can be configured.	String	value.schema.version.id

The KafkaAvroSerializer contains an embedded SchemaRegistryClient. The configuration for SchemaRegistryCl ient should also be provided for the KafkaAvroSerializer. For more information about configurable properties for the *Schema Registry Client properties* reference documentation.

Related Information

Schema Registry Client properties reference

KafkaAvroDeserializer properties reference

Review the following reference for a comprehensive list of the KafkaAvroDeserializer properties.

Property Name	Description	Type	Default Value
schemaregistry.reader.schema.versions	This is for setting which version of the schema should be used for which topic.[1]	Map<String, Integer>	Empty map
specific.avro.reader	If this is enabled, SpecificDatumReader will be used for deserialization.	Boolean	false
logical.type.conversion.enabled	If this is enabled, Avro record fields that have logical types can be deserialized directly into their corresponding Java classes (BigDecimal, UUID, LocalDate, and so on). If not enabled, the record field will be deserialized as raw type, without handling the logical type. In this case, probably the same logical type related conversion will be needed to get a valid value. If specific.avro.reader is set to true, this does not have any effect.	Boolean	false
key_schema_version_id_header_name	If the schema of the record key is stored in the header, the header name can be configured.	String	key.schema.version.id
value_schema_version_id_header_name	If the schema of the record value is stored in the header, the header name can be configured.	String	value.schema.version.id

The KafkaAvroDeserializer contains an embedded SchemaRegistryClient. The configuration for SchemaRegistryClient should also be provided for the KafkaAvroDeserializer. For more information about configurable properties for the *Schema Registry Client properties* reference documentation.

Related Information

Schema Registry Client properties reference

[1] Example:

```
Map<String, Object> configs = new HashMap<>()
Map<String, Integer> readerVersions = new HashMap<>();
readerVersions.put("clicks", 2);
readerVersions.put("users", 1);
configs.put(“schemaregistry.reader.schema.versions”, readerVersions);
```