KIP 577: Allow HTTP Response Headers to be Configured for Kafka Connect

Status

Current state: Accepted

Discussion thread: *here*

Vote thread: *here*

JIRA: ⚠ Unable to render Jira issues macro, execution error.

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Kafka users have reported that their security scanners have identified some missing HTTP headers from Connect REST API response. Specific headers that customers have asked about include:

- X-XSS-Protection
- X-Content-Type-Options
- Strict-Transport-Security
- X-Frame-Options
- Content-Security-Policy

We should provide a mechanism to return the desired headers. We can't, in general, anticipate what specific headers their scanners will demand now, much less in the future, so we must make this mechanism configurable. Also we don't foresee the requirement to set different headers for different paths or mime types (since Connect API only return application/json).

Public Interfaces

This proposal adds a new configuration property to customize HTTP response headers. The following section has detailed description.

response.http.headers.config

Type: Comma separated string

Valid Values: Values must use same format as headerConfig defined in Jetty's HeaderFilter documentation:

```
[action] [header name]: [header value]
```

`[action]` can be one of `set`, `add`, `setDate`, or `addDate` which specify an action will perform on header.

- `set` action is same as setHeader function in HttpServletResponse, it will set a response header with the given name and value. If the header had already been set, the new value overwrites the previous one.
- add action is same as addHeader function in HttpServletResponse, it will add a new value to the header. Responses headers could have multiple values.
- `setDate` action is same as setDateHeader function in HttpServletResponse. It will set HTTP header need date value. Such as `setDate Expires: 31540000000` which indicates the header will be expired approximately one year in the future.

- `addDate` action is same as addDateHeader function in HttpServletResponse. It will add a response header with the given name and date-value. Such as `addDate Last-Modified: 0` which indicates the Last-Modified date is same as current system date.

`[header name]` name of header (For examples, see: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers)

`[header value]` value for the header. We need put double quotes around the value if the value contains commas due to we use comma as separator for different headers. Default Value: ""

Example:

```
response.http.headers.config="add Cache-Control: no-cache, no-store, must-revalidate", add X-XSS-Protection: 1;
mode=block, add Strict-Transport-Security: max-age=31536000; includeSubDomains, add X-Content-Type-Options:
nosniff
```

Output of Response Header:

```
< HTTP/1.1 200 OK
< Date: Sat, 07 Mar 2020 17:33:39 GMT
< Strict-Transport-Security: max-age=31536000;includeSubDomains
< X-XSS-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< Cache-Control: no-cache, no-store, must-revalidate
< Content-Type: application/json
< Vary: Accept-Encoding, User-Agent
< Content-Length: 136
```

Proposed Changes

Since we're using Jetty to serve this, we can take advantage of the Jetty HeaderFilter class to allow the configuration of these headers. We add configuration property `response.http.headers.config` in the `org.apache.kafka.connect.runtime.WorkerConfig` class and update org. apache.kafka.connect.runtime.rest.RestServer class. During initialization process, Connect REST server will read all header configurations from the property `response.http.headers.config`, and create a FilterHolder with HeaderFilter class and add the filter holder to the Servlet context handler. We only need to provide a single HeaderFilter for the entire server, because that HeaderFilter can set as many headers as the customer needs. B oth `set` and `add` action will add a header and value to the response header if the header is not already in the response. When the header is there, `set` action overwrites the existing value, whereas `add` action adds an additional value to the header value. So it is the Kafka user's applications and server administrator responsibility to manage headers configured and existing headers.

The following is flow how this will be implemented in RestServer class:

- loads configuration file
- parses the property `response.http.headers.config` in configuration file sing WorkerConfig
- creates FilterHolder with HeadFilter class.
- adds the FilterHolder into ServletContextHandler

Pseudocode

```
private void configureHttpResponsHeaderFilter(ServletContextHandler context)
  String headerConfig = workerConfig.getString(WorkerConfig.RESPONSE_HTTP_HEADERS_CONFIG);
  FilterHolder headerFilterHolder = new FilterHolder(HeaderFilter.class);
  headerFilterHolder.setName("default");
  headerFilterHolder.setInitParameter("headerConfig", headerConfig);
  context.addFilter(headerFilterHolder, "/*", EnumSet.of(DispatcherType.REQUEST));
}
```

Resources

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

https://www.eclipse.org/jetty/documentation/current/header-filter.html

https://www.eclipse.org/jetty/javadoc/9.4.24.v20191120/org/eclipse/jetty/servlets/HeaderFilter.html

Compatibility, Deprecation, and Migration Plan

This is a new feature that add HTTP headers based on configuration property `response.http.headers.config`. The default value for `response. http.headers.config` is empty, so it is backward compatible to old version.

Rejected Alternatives

Another implementation would be writing a customized filter extension to intercept and set HTTP response headers and we have to define same configuration property described in this proposal. Ultimately the purpose of this design will allow users to set HTTP response headers. Using this alternative approach make implementation much complex and doesn't gain any benefits.

Supporting multiple header filters add complexity to configuration properties and cause site administrator confusions on configuring Connect REST Server. Most important thing is we do not see applicable customer scenarios at this time. One header filter is sufficient for all HTTP response headers based on existing customer application scenarios and concern. We could expand existing implementation easily if customer need it in the future. For instance, we could add multiple header filters by adding name of header filter between prefix response.http.headers and config such as response.http.headers.{name}.config. We still keep response.http.headers.config as default header filter and internally set name of header filter to default if there is nothing between the response.http.headers and config.

Supporting other optional parameters, that are `response.http.headers.included.paths`, `response.http.headers.included.mime.types`, `response.http.headers.included.methods`, `response.http.headers.excluded.paths`, `response.http.headers.excluded.mime.types`, `response.http.headers.excluded.methods`, don't add value based on existing customer complain about missing header in HTTP response headers. Customer will be happy as long as the headers they want in HTTP response header. This will simply configuration properties lot. We could easily add these optional parameters if customer need it in the future.

Another option would be to try to determine what the right headers are in all cases, and always send them. This is appealing because it would not require any end-user customization. However new security headers are regularly added by the web development community, and it would be difficult for us to anticipate all of our users' needs. By providing a configurable option, users can implement the headers that make sense in their own security environments.