**package** com.mastercard.testing.gdp.api.helper;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.ContextConstants.***RESPONSE***;

**import** **static** io.restassured.RestAssured.given;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.InputStreamReader;

**import** java.io.StringWriter;

**import** java.io.UnsupportedEncodingException;

**import** java.nio.charset.Charset;

**import** java.nio.charset.StandardCharsets;

**import** java.sql.Timestamp;

**import** java.text.SimpleDateFormat;

**import** java.time.Instant;

**import** java.time.LocalDateTime;

**import** java.time.ZoneId;

**import** java.time.ZoneOffset;

**import** java.time.format.DateTimeFormatter;

**import** java.util.Arrays;

**import** java.util.Base64;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Objects;

**import** java.util.Random;

**import** java.util.Set;

**import** java.util.SimpleTimeZone;

**import** org.apache.commons.io.FileUtils;

**import** org.apache.commons.io.IOUtils;

**import** org.apache.http.HttpEntity;

**import** org.apache.http.HttpResponse;

**import** org.apache.http.auth.AuthScope;

**import** org.apache.http.auth.UsernamePasswordCredentials;

**import** org.apache.http.client.ClientProtocolException;

**import** org.apache.http.client.CredentialsProvider;

**import** org.apache.http.client.methods.HttpPost;

**import** org.apache.http.entity.ContentType;

**import** org.apache.http.entity.StringEntity;

**import** org.apache.http.entity.mime.MultipartEntityBuilder;

**import** org.apache.http.entity.mime.content.FileBody;

**import** org.apache.http.impl.client.BasicCredentialsProvider;

**import** org.apache.http.impl.client.CloseableHttpClient;

**import** org.apache.http.impl.client.HttpClientBuilder;

**import** org.apache.http.impl.client.HttpClients;

**import** org.json.JSONArray;

**import** org.json.JSONObject;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.core.env.Environment;

**import** org.springframework.http.MediaType;

**import** com.fasterxml.jackson.core.JsonParseException;

**import** com.fasterxml.jackson.databind.DeserializationFeature;

**import** com.fasterxml.jackson.databind.JsonMappingException;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**import** com.mastercard.quality.engineering.mtaf.jbehave.context.TestContextProvider;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

**import** com.mastercard.testing.gdp.api.helper.constants.Constants;

**import** io.restassured.builder.RequestSpecBuilder;

**import** io.restassured.response.Response;

**import** io.restassured.specification.RequestSpecification;

**public** **class** CommonFunctions {

**private** **static** **final** String ***DATE\_FORMATTER\_MMM\_DD\_YYYY*** = "MMM dd, yyyy";

**private** LocalDateTime timeBeforeTestBegin;

**private** LocalDateTime timeAfterTestEnd;

**private** Random generator = **new** Random();

@Autowired

**private** ObjectMapper mapper;

@Autowired

**private** Environment env;

@Value("${rest.response.unwraproot:false}")

**private** Boolean unwrapRootValue;

@Autowired

**protected** TestContextProvider testContextProvider;

**private** **static** **final** Logger ***logger*** = LoggerFactory.getLogger(CommonFunctions.**class**);

**protected** String baseJSONUrl;

**private** HttpResponse httpResponse;

@Value("${aker.proxy.per.request.timeout}")

**private** String akerProxyPerRequestTimeout;

**public** **void** setBaseURL(String baseURL) {

baseJSONUrl = baseURL;

}

**public** RequestSpecification getReqSpecBuilder(String resPath) {

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(resPath)

.addHeader(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***HEADER\_NAME\_ACCCEPT***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout).build();

}

**public** RequestSpecification getReqSpecBuilderWithEmptyBody(String resPath) {

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath("").setBody("")

.addHeader(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***HEADER\_NAME\_ACCCEPT***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout).build();

}

**public** RequestSpecification getPutRequestBuilder(String resourcePath) {

**return** getReqSpecBuilderWithEmptyBody(resourcePath);

}

**public** RequestSpecification getGetRequestBuilder(String resourcePath, String reqParam) {

StringBuilder resPathSB = **new** StringBuilder(resourcePath);

String reqParameter = reqParam;

String resPath;

**if** (reqParameter == **null** || reqParameter.equals("")) {

resPath = resPathSB.toString();

} **else** {

resPath = resPathSB.append(reqParameter).toString();

}

**return** getReqSpecBuilder(resPath);

}

**public** RequestSpecification getGetRequestBuilderWithDefaultAndGivenHeadersAndQueryParams(String resourcePath,

String reqParam, Map<String, String> headers, Map<String, Object> queryParam) {

StringBuilder resPathSB = **new** StringBuilder(resourcePath);

String resPath;

String reqParameter = reqParam;

**if** (reqParameter == **null** || reqParam.equals("")) {

resPath = resPathSB.toString();

} **else** {

resPath = resPathSB.append(reqParam).toString();

}

headers.put(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***);

headers.put(Constants.***HEADER\_NAME\_ACCCEPT***, Constants.***APPLICATION\_JSON***);

headers.put(Constants.***CORRELATION\_ID***, getCurrentDate());

headers.put(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout);

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(resPath).addHeaders(headers)

.addQueryParams(queryParam).build();

}

**public** RequestSpecification getGetRequestBuilderWithHeaders(String resourcePath, String reqParam,

Map<String, String> headers) {

StringBuilder resPathSB = **new** StringBuilder(resourcePath);

String resPath;

String reqParameter = reqParam;

**if** (reqParameter == **null** || reqParam.equals("")) {

resPath = resPathSB.toString();

} **else** {

resPath = resPathSB.append(reqParam).toString();

}

headers.put(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***);

headers.put(Constants.***HEADER\_NAME\_ACCCEPT***, Constants.***APPLICATION\_JSON***);

headers.put(Constants.***CORRELATION\_ID***, getCurrentDate());

headers.put(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout);

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(resPath).addHeaders(headers).build();

}

**public** RequestSpecification getGetRequestBuilderWithQueryParams(String resourcePath, String reqParam,

Map<String, Object> queryParam) {

StringBuilder resPathSB = **new** StringBuilder(resourcePath);

String resPath;

String reqParameter = reqParam;

**if** (reqParameter == **null** || reqParameter.equals("")) {

resPath = resPathSB.toString();

} **else** {

resPath = resPathSB.append(reqParameter).toString();

}

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(resPath).addQueryParams(queryParam).build();

}

**public** RequestSpecification getPostRequestBuilderWithCustomHeaders(String link, String requestBody,

String[] reqHeader) {

Map<String, String> listOfHeaders = **new** HashMap<>();

**for** (**int** count = 0; count < reqHeader.length; count++) {

String key = reqHeader[count].split("=")[0];

String value = reqHeader[count].split("=")[1];

listOfHeaders.put(key, value);

}

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(link).setBody(requestBody).addHeaders(listOfHeaders)

.addHeader(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout).build();

}

**public** RequestSpecification getPostRequestBuilderWithCustomHeaders(String resourcePath, String requestBody,

String pathToUrl, Map<String, String> headers) {

RequestSpecBuilder specBuilder = getRequestSpecBuilder(resourcePath, **null**, pathToUrl);

**return** specBuilder.addHeaders(headers).setBody(requestBody).build();

}

**public** RequestSpecBuilder getRequestSpecBuilder(String resourcePath, String reqParam, String pathToUrl) {

StringBuilder resPathSB = **new** StringBuilder(resourcePath);

String resPath;

String reqParameter = reqParam;

**if** (reqParameter == **null** || reqParameter.equals("")) {

resPath = resPathSB.toString();

} **else** {

resPath = resPathSB.append(reqParameter).toString();

}

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(resPath)

.addHeader(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***HEADER\_NAME\_ACCCEPT***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout);

}

**public** RequestSpecification getPostRequestBuilder(String link, String requestBody, String baseJSONUrl) {

**return** **new** RequestSpecBuilder().setBaseUri(baseJSONUrl).setBasePath(link).setBody(requestBody)

.addHeader(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout).build();

}

**public** RequestSpecification getDeleteRequestBuilder(String resourcePath, String reqParam) {

StringBuilder resPathSB = **new** StringBuilder(resourcePath);

String resPath;

String reqParameter = reqParam;

**if** (reqParameter == **null** || reqParameter.equals("")) {

resPath = resPathSB.toString();

} **else** {

resPath = resPathSB.append(reqParameter).toString();

}

**return** getReqSpecBuilder(resPath);

}

**public** RequestSpecification getPutRequestBuilder(String resourcePath, String reqParam) {

**return** getDeleteRequestBuilder(resourcePath, reqParam);

}

**public** RequestSpecification getPutRequestBuilder(String resourcePath, **int** reqParam, String requestBody) {

String resPath = resourcePath;

**if** (reqParam != 0) {

resPath = resourcePath + reqParam;

}

**return** **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(resPath).setBody(requestBody)

.addHeader(Constants.***HEADER\_NAME\_CONTENT\_TYPE***, Constants.***APPLICATION\_JSON***)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout).build();

}

**public** <T> T parseResponse(String content, Class<T> returnType) {

**this**.mapper.configure(DeserializationFeature.UNWRAP\_ROOT\_VALUE, **this**.unwrapRootValue);

**if** (content == **null**) {

**return** **null**;

} **else** {

**try** {

**return** **this**.mapper.readValue(content, returnType);

} **catch** (JsonParseException | JsonMappingException jsonParseException) {

**throw** **new** GDPAPIException(jsonParseException.getMessage(), jsonParseException);

} **catch** (IOException ioException) {

**throw** **new** GDPAPIException(ioException.getMessage(), ioException);

}

}

}

**public** <T> T parseResponse(Object content, Class<T> returnType) {

**this**.mapper.configure(DeserializationFeature.UNWRAP\_ROOT\_VALUE, **this**.unwrapRootValue);

**if** (content == **null**) {

**return** **null**;

} **else** {

**try** {

String contentString = **this**.mapper.writeValueAsString(content);

**return** **this**.mapper.readValue(contentString, returnType);

} **catch** (JsonParseException jsonParseException) {

**throw** **new** GDPAPIException(jsonParseException.getMessage(), jsonParseException);

} **catch** (JsonMappingException jsonMappingException) {

**throw** **new** GDPAPIException(jsonMappingException.getMessage(), jsonMappingException);

} **catch** (IOException ioException) {

**throw** **new** GDPAPIException(ioException.getMessage(), ioException);

}

}

}

**private** Object setEntityInPost(HttpPost post, MultipartEntityBuilder entity) **throws** IOException {

post.setEntity(entity.build());

StringWriter stringWriter = **new** StringWriter();

Object jsonObject = **null**;

**try** {

CloseableHttpClient client = HttpClientBuilder.create().build();

jsonObject = invokeAPIAndGetResponse(post, stringWriter, client);

} **catch** (GDPAPIException exp) {

**throw** **new** GDPAPIException(exp.getMessage(), exp);

}

**return** jsonObject;

}

**private** **void** logRequestInformation(HttpPost post) {

**if** (***logger***.isInfoEnabled()) {

***logger***.info("Request Method: {}", post.getMethod());

***logger***.info("Request Endpoint: {}", post.getURI());

***logger***.info("Request Headers: {}", Arrays.asList(post.getAllHeaders()));

}

}

**private** Object invokeAPIAndGetResponse(HttpPost post, StringWriter writer, CloseableHttpClient client)

**throws** IOException {

String successResponse;

Object jsonObject = **null**;

logRequestInformation(post);

httpResponse = client.execute(post);

String encoding = StandardCharsets.***UTF\_8***.name();

IOUtils.copy(httpResponse.getEntity().getContent(), writer, encoding);

**if** ("".equals(writer.toString())) {

successResponse = "{\"data\":\"success\"}";

} **else** {

successResponse = writer.toString();

}

**try** {

jsonObject = **new** JSONObject(successResponse);

} **catch** (Exception e) {

***logger***.info("Response is not Json Object Proceeding converting to JsonArray");

**try** {

jsonObject = **new** JSONArray(successResponse);

} **catch** (Exception e1) {

***logger***.info("Response is not JsonArray");

}

}

**return** jsonObject;

}

**public** **void** printResponseBody(Response response) {

**if** (***logger***.isInfoEnabled()) {

***logger***.info("API Response Body is : {} " + response.toString());

***logger***.info("API Response Status Code is : {} " + response.getStatusCode());

}

}

**public** String getCurrentDate() {

SimpleDateFormat completedDateFormat = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss.sss");

completedDateFormat.setTimeZone(**new** SimpleTimeZone(SimpleTimeZone.UTC\_TIME, Constants.***STLOUIS\_TIMEZONE***));

**return** completedDateFormat.format(**new** Date());

}

**public** Response executeGetRequest(RequestSpecification reqSpec) {

Response response = given().spec(reqSpec).log().all().when().get();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executeGetRequest(String link, String reqParam) {

RequestSpecification builder = getGetRequestBuilder(link, reqParam);

Response response = given().spec(builder).log().all().when().get();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executeGetRequestWithGivenHeadersAndQueryParams(String resourcePath, String reqParam,

Map<String, String> headers, Map<String, Object> queryParam) {

RequestSpecification builder = getGetRequestBuilderWithDefaultAndGivenHeadersAndQueryParams(resourcePath,

reqParam, headers, queryParam);

Response response = given().spec(builder).log().all().when().get();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executeGetRequestWithGivenHeaders(String resourcePath, String reqParam,

Map<String, String> headers) {

RequestSpecification builder = getGetRequestBuilderWithHeaders(resourcePath, reqParam, headers);

Response response = given().spec(builder).log().all().when().get();

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executeGetRequestWithQueryParams(String link, String reqParam, Map<String, Object> queryParam) {

RequestSpecification builder = getGetRequestBuilderWithQueryParams(link, reqParam, queryParam);

Response response = given().spec(builder).log().all().when().get();

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executeGetRequest(String link) {

RequestSpecification builder = **new** RequestSpecBuilder().setBaseUri(env.getProperty(baseJSONUrl)).setBasePath(link)

.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate())

.addHeader(Constants.***AKER\_PROXY\_REQUEST\_TIMEOUT***, akerProxyPerRequestTimeout).build();

Response response = given().spec(builder).log().all().when().get();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executePostRequestwithCustomHeadersArray(String link, String requestBody, String... reqHeader) {

RequestSpecification builder = getPostRequestBuilderWithCustomHeaders(link, requestBody, reqHeader);

Response response = given().spec(builder).log().all().when().post();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executePostRequestWithCustomHeaders(String link, String requestBody, Map<String, String> headers) {

RequestSpecification builder = getPostRequestBuilderWithCustomHeaders(link, requestBody, baseJSONUrl, headers);

Response response = given().spec(builder).log().all().post();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executePostRequest(String link, String requestBody) {

RequestSpecification builder = getPostRequestBuilder(link, requestBody, baseJSONUrl);

Response response = given().spec(builder).log().all().when().post();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executePutRequest(String resourcePath, String pathParameter) {

RequestSpecification builder = getPutRequestBuilder(resourcePath, pathParameter);

Response response = given().spec(builder).log().all().when().put();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executePutRequest(String resourcePath) {

RequestSpecification builder = getPutRequestBuilder(resourcePath);

Response response = given().spec(builder).log().all().when().put();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executePutRequestWithRequestBody(String resourcePath, **int** pathParameter, String requestBody) {

RequestSpecification builder = getPutRequestBuilder(resourcePath, pathParameter, requestBody);

Response response = given().spec(builder).log().all().when().put();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response executeDeleteRequest(String link, String reqParam) {

RequestSpecification builder = getDeleteRequestBuilder(link, reqParam);

Response response = given().spec(builder).log().all().when().delete();

printResponseBody(response);

testContextProvider.get().put(***RESPONSE***, response);

**return** response;

}

**public** Response getActualStatusCodeFromContextProvider() {

**return** testContextProvider.get().get(***RESPONSE***);

}

**public** Response getResponseFromContextProvider() {

**return** testContextProvider.get().get(***RESPONSE***);

}

**public** <T> T getActualPojoClassesFromContext(Class<T> returnType) {

ResponseConverter<?> response = testContextProvider.get().get(***RESPONSE***);

**return** parseResponse(response.getResponseBody(), returnType);

}

**public** **int** getActualStatusCodeFromContext() {

ResponseConverter<?> response = testContextProvider.get().get(***RESPONSE***);

**return** response.getStatusLine().getStatusCode();

}

**public** String getResponseFromContext() {

ResponseConverter<?> response = testContextProvider.get().get(***RESPONSE***);

**return** response.getResponseBody();

}

**public** **int** getActualStatusCodeFromClientResponse() {

**return** testContextProvider.get().get(***RESPONSE***);

}

**public** String getResponseBody() {

**try** {

**return** getResponseFromContext();

} **catch** (ClassCastException ex) {

***logger***.info("failed to get responseBody from getResponseFromContext ");

}

**try** {

**return** getResponseFromContextProvider().body().asString();

} **catch** (ClassCastException ex) {

***logger***.info("failed to get responseBody from getResponseBody ");

**throw** ex;

}

}

**public** Object sendAttachmentInPost(String link, String filePath) **throws** IOException {

HttpPost httpPost = **new** HttpPost(link);

File fileToUpload = **new** File(filePath);

MultipartEntityBuilder entity = MultipartEntityBuilder.create();

entity.addPart(Constants.***FILE***, **new** FileBody(fileToUpload));

**return** setEntityInPost(httpPost, entity);

}

**public** Object sendAttachmentInPost(String link, String filePath, String username, String password)

**throws** IOException {

CredentialsProvider credentialsProvider = **new** BasicCredentialsProvider();

UsernamePasswordCredentials credentials = **new** UsernamePasswordCredentials(username, password);

credentialsProvider.setCredentials(AuthScope.ANY, credentials);

HttpPost httpPost = **new** HttpPost(link);

File fileUpload = **new** File(filePath);

MultipartEntityBuilder entityBuilder = MultipartEntityBuilder.create();

entityBuilder.addPart(Constants.***FILE***, **new** FileBody(fileUpload));

httpPost.setEntity(entityBuilder.build());

StringWriter stringWriter = **new** StringWriter();

Object jsonObject;

**try** (CloseableHttpClient client = HttpClientBuilder.create().setDefaultCredentialsProvider(credentialsProvider)

.build();) {

jsonObject = invokeAPIAndGetResponse(httpPost, stringWriter, client);

} **catch** (GDPAPIException exp) {

**throw** **new** GDPAPIException(exp.getMessage(), exp);

}

**return** jsonObject;

}

**public** **int** getStatusCodeFromHttpResponse() {

**return** httpResponse.getStatusLine().getStatusCode();

}

**public** String getHttpResponse() **throws** IOException {

**return** httpResponse.getEntity().getContent().toString();

}

**public** <T> T getActualPojoClasses(Class<T> returnType) {

Response response = testContextProvider.get().get(***RESPONSE***);

**return** parseResponse(response.getBody().asString(), returnType);

}

**public** <T> T getActualPojoClassesFromResponse(Class<T> returnType, String response) {

**return** parseResponse(response, returnType);

}

**public** String getResponseAsString() {

Response response = testContextProvider.get().get(***RESPONSE***);

**return** response.asString();

}

**public** **void** compareJSONStrings(String string, String responseBody)

**throws** JsonParseException, JsonMappingException, IOException {

mapper = **new** ObjectMapper();

Map<String, Object> actualJson = mapper.readValue(string, Map.**class**);

Map<String, Object> expectedJSON = mapper.readValue(responseBody, Map.**class**);

// assertThat(actualJson).containsAllEntriesOf(expectedJSON);

}

**public** String getPreviousFutureDate(**int** days) {

SimpleDateFormat completedDateFormat = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss.sss");

completedDateFormat.setTimeZone(**new** SimpleTimeZone(SimpleTimeZone.UTC\_TIME, Constants.***STLOUIS\_TIMEZONE***));

Calendar cal = Calendar.getInstance();

cal.add(Calendar.DATE, days);

Date date = cal.getTime();

**return** completedDateFormat.format(date);

}

**public** String getAuthorizationHeader(String userName, String password) {

**return** "Authorization=" + getAuthorizationHeaderValue(userName, password);

}

**public** String getAuthorizationHeaderValue(String userName, String password) {

Charset charSet = StandardCharsets.***UTF\_8***;

**return** "Basic " + Base64.getEncoder().encodeToString((userName + ":" + password).getBytes(charSet));

}

**public** **int** generateRandomNumberInRange(**int** lastIndex) {

**return** generator.nextInt();

}

**public** **void** setStartTime() {

timeBeforeTestBegin = getCurrentCSTLocalDateTime();

}

**public** **void** setEndTime() {

timeAfterTestEnd = getCurrentCSTLocalDateTime().plusSeconds(10);

}

**private** LocalDateTime getCurrentCSTLocalDateTime() {

**double** offsetHours;

**if** (ZoneId.of("America/Chicago").getRules().isDaylightSavings(Instant.now())) {

offsetHours = -5;

} **else** {

offsetHours = -6;

}

ZoneOffset offset = ZoneOffset.ofHours((**int**) offsetHours);

**return** Instant.now().atOffset(offset).toLocalDateTime();

}

**public** **boolean** verifyTimeStampIsInBetweenStartAndEndTimes(LocalDateTime time) {

**return** (timeBeforeTestBegin.isBefore(time) && timeAfterTestEnd.isAfter(time));

}

**public** JSONObject getJsonObjectByName(String fieldName) {

**return** **new** JSONObject(getResponseFromContextProvider().body().asString()).optJSONObject(fieldName);

}

**public** Map<String, Object> convertBodyToObjectMap(String requestBody) **throws** IOException {

**return** (Map<String, Object>) (mapper.readValue(requestBody, Map.**class**));

}

**public** HttpResponse executePostMethod(String link, String requestBody) **throws** ClientProtocolException, IOException {

CloseableHttpClient client = HttpClients.custom().build();

HttpPost post = **new** HttpPost(link);

post.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate());

StringEntity entity = **new** StringEntity(requestBody);

entity.setContentType(Constants.***APPLICATION\_JSON***);

post.setEntity(entity);

logRequestInformation(post);

**return** client.execute(post);

}

**public** **boolean** areTheFilesIdentical(String fileName1, String fileName2) **throws** IOException {

File file1 = **new** File(getAbsoluteFilePath(fileName1));

File file2 = **new** File(getAbsoluteFilePath(fileName2));

**return** FileUtils.contentEquals(file1, file2);

}

**private** String getAbsoluteFilePath(String fileName1) {

**return** String.format(Constants.***DOCUMENT\_DOWNLOAD\_PATH***, fileName1);

}

**public** String formatDateWithMonthFirst(Timestamp timestamp) {

LocalDateTime convertedTime = timestamp.toLocalDateTime();

**return** convertedTime.format(DateTimeFormatter.ofPattern(***DATE\_FORMATTER\_MMM\_DD\_YYYY***));

}

**public** Date getCurrentSystemTime() {

Calendar calendar = Calendar.getInstance();

calendar.setTimeInMillis(System.currentTimeMillis());

**return** calendar.getTime();

}

**public** **void** compareJSONArrays(String expetcedJson, String responseData) **throws** IOException {

JSONArray actualJsonArray = **new** JSONArray(expetcedJson);

JSONArray expectedJsonArray = **new** JSONArray(responseData);

**for** (**int** n = 0; n < actualJsonArray.length(); n++) {

compareJSONArrays(**new** JSONObject(actualJsonArray.getJSONObject(n).toString()).toString(),

**new** JSONObject(expectedJsonArray.getJSONObject(n).toString()).toString());

}

}

**public** **void** compareJSONArraysWithoutJsonObject(String expetcedJson, String responseData) {

JSONArray actualJsonArray = **new** JSONArray(responseData);

JSONArray expectedJsonArray = **new** JSONArray(expetcedJson);

// assertThat(actualJsonArray).containsAll(expectedJsonArray);

}

**public** **boolean** checkIfExpectedValuesPresentInActualMap(Map<String, Object> mergeDataAnyExpected,

Map<String, Object> mergeDataAnyActual) {

**boolean** val = **true**;

Set<String> keySet = mergeDataAnyExpected.keySet();

**for** (String key : keySet) {

**if** (mergeDataAnyActual.containsKey(key)) {

Object actualVal = mergeDataAnyActual.get(key);

Object expectedVal = mergeDataAnyExpected.get(key);

**if** (!actualVal.equals(expectedVal)) {

val = **false**;

***logger***.info("actualVal:" + actualVal + " But expectedVal" + expectedVal);

} **else** {

val = **true**;

}

}

}

**return** val;

}

**public** String getEncryptedValue(String stringToEncrypt) {

**return** **new** String(

java.util.Base64.getEncoder()

.encode(org.apache.commons.codec.binary.Base64

.encodeBase64(stringToEncrypt.getBytes(StandardCharsets.***ISO\_8859\_1***), **false**)),

StandardCharsets.***ISO\_8859\_1***);

}

**private** <T> ResponseConverter<T> getResponseValues(HttpResponse response) {

ResponseConverter<T> responseWrapper = **new** ResponseConverter<>();

**if** (response.getAllHeaders() != **null**)

responseWrapper.setHeaders(response.getAllHeaders());

String responseBody = **null**;

**try** {

**if** (response.getEntity().getContentType() != **null**)

responseWrapper.setContentType(response.getEntity().getContentType());

**if** (response.getEntity().getContent() != **null**)

responseBody = convertEntityToString(response.getEntity().getContent());

responseWrapper.setResponseBody(responseBody);

} **catch** (IOException e1) {

**throw** **new** GDPAPIException(e1.getMessage(), e1);

}

**if** (response.getStatusLine() != **null**)

responseWrapper.setStatusLine(response.getStatusLine());

**return** responseWrapper;

}

**private** <T2> ResponseConverter<T2> parseResponse(ResponseConverter<T2> response, Class<T2> returnType) {

mapper.configure(DeserializationFeature.UNWRAP\_ROOT\_VALUE, unwrapRootValue);

/\*\*

\* As we need both content type to determine which type and response length

\* should not be zero

\*/

**if** (response.getContentType() == **null** && response.getResponseBody().length() == 0) {

**return** **null**;

}

**if** (response.getContentType() == **null** && response.getResponseBody().length() == 0) {

**return** **null**;

}

**if** (response.getContentType().toString().contains(MediaType.APPLICATION\_JSON\_VALUE)) {

**try** {

response.setFormattedResponseBody(mapper.readValue(response.getResponseBody(), returnType));

} **catch** (IOException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

}

**return** response;

}

**private** String convertEntityToString(InputStream responseBody) **throws** UnsupportedEncodingException {

BufferedReader rd = **new** BufferedReader(**new** InputStreamReader(responseBody, StandardCharsets.***UTF\_8***.name()));

StringBuilder result = **new** StringBuilder();

String line = "";

**try** {

**while** ((line = rd.readLine()) != **null**) {

result.append(line);

}

} **catch** (IOException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

**return** result.toString();

}

**public** <T> ResponseConverter<T> executePostRequestWithtoUploadFiles(String resourcePath, String fileRepoId,

String userId, String file, String date, Class<T> returnType) **throws** IOException {

HttpEntity entity = buildFileUploadHttpEntity(fileRepoId, userId, file, date);

HttpPost post = **new** HttpPost(resourcePath);

post.addHeader(Constants.***CORRELATION\_ID***, getCurrentDate());

post.setEntity(entity);

ResponseConverter<T> responseWrapper = **null**;

logRequestInformation(post);

**try** (CloseableHttpClient client = HttpClientBuilder.create().build()) {

httpResponse = client.execute(post);

responseWrapper = getResponseValues(httpResponse);

responseWrapper = parseResponse(responseWrapper, returnType);

testContextProvider.get().put(***RESPONSE***, responseWrapper);

***logger***.info("Response Code: {}", Objects.requireNonNull(responseWrapper).getStatusLine().getStatusCode());

***logger***.info("Response Body: {}", responseWrapper.getResponseBody());

} **catch** (Exception e) {

***logger***.error("Error Response Body is: {}");

**throw** **new** GDPAPIException(e.getMessage(), e);

}

**return** responseWrapper;

}

**private** HttpEntity buildFileUploadHttpEntity(String fileRepoId, String userId, String file, String date) {

**return** MultipartEntityBuilder.create().addTextBody(Constants.***FILE\_REPO\_ID***, fileRepoId)

.addTextBody(Constants.***USER\_ID***, userId).addBinaryBody(Constants.***FILE***,

**new** File(getAbsoluteFilePath(file)), ContentType.create(Constants.***APPLICATION\_JSON***), file)

.addTextBody(Constants.***DATE***, date).build();

}

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** java.util.Arrays;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

@Component

**public** **class** DefaultTestDataProvider **implements** TestDataProvider {

@Autowired

**private** TestDataExpressionResolver expressionResolver;

**public** **static** **final** String ***PROVIDER\_PREFIX*** = "default";

**public** **static** **final** String ***VALUE\_KEY*** = "value";

**public** **static** **final** Pattern ***EXPRESSION\_PARAMS*** = Pattern.*compile*("[a-z]\\{[\\w.\\s]\*\\}");

@Override

**public** String getValue(String expression) {

String value = expression;

Matcher matcher = ***EXPRESSION\_PARAMS***.matcher(expression);

**while** (matcher.find()) {

String substitute = expressionResolver.resolveExpression(matcher.group());

value = value.replace("(" + matcher.group() + ")", substitute);

}

**return** value;

}

@Override

**public** List<String> getList(String expression) {

**return** Arrays.*asList*(expression);

}

@Override

**public** Map<String, Object> getMap(String expression) {

Map<String, Object> result = **new** HashMap<>();

result.put(***VALUE\_KEY***, expression);

**return** result;

}

@Override

**public** **boolean** support(String prefix) {

**return** ***PROVIDER\_PREFIX***.equals(prefix);

}

}

**package** com.mastercard.testing.gdp.api.aspect;

@FunctionalInterface

**public** **interface** StepArgumentProvider {

Object[] getLastStepArguments();

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** java.util.List;

**public** **interface** TestDataExpressionResolver {

String resolveExpression(String expression);

List<String> resolveListExpression(String expression);

}

//

// Source code recreated from a .class file by IntelliJ IDEA

// (powered by Fernflower decompiler)

//

**package** com.mastercard.quality.engineering.mtaf.jbehave.context;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.commons.lang3.StringUtils;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** TestContext {

**private** **static** Logger *LOG* = LoggerFactory.getLogger(TestContext.**class**);

**protected** Map<String, Object> contextMap = **new** HashMap();

**public** TestContext() {

}

**public** **void** put(String key, Object value) **throws** IllegalArgumentException {

**if** (StringUtils.isEmpty(key)) {

String errorMsg = "Object cannot be null.";

IllegalArgumentException ex = **new** IllegalArgumentException(errorMsg);

*LOG*.error(errorMsg, ex);

**throw** ex;

} **else** {

*LOG*.info(key + " added to context.");

**this**.contextMap.put(key, value);

}

}

**public** **void** remove(String key) {

**if** (StringUtils.isNotEmpty(key)) {

**this**.contextMap.remove(key);

}

}

@SuppressWarnings("unchecked")

**public** <T> T get(String key) {

Object obj = **this**.contextMap.get(key);

**if** (obj == **null**) {

*LOG*.warn("Object of key " + key + " was not found in context. Returning null.");

**return** **null**;

} **else** {

**return** (T) obj;

}

}

**public** **void** clear() {

**this**.contextMap.clear();

}

}

**package** com.mastercard.quality.engineering.mtaf.jbehave.context;

**public** **class** TestContextProvider {

**private** **static** **final** ThreadLocal<TestContext> ***threadLocalContext*** = **new** ThreadLocal<TestContext>();

**public** TestContext get() {

TestContext context = (TestContext) ***threadLocalContext***.get();

**if** (context == **null**) {

context = **new** TestContext();

***threadLocalContext***.set(context);

**return** context;

} **else** {

**return** context;

}

}

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** java.util.Arrays;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.commons.lang3.StringUtils;

**import** org.springframework.stereotype.Component;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

@Component

**public** **class** ContextVariableTestData **implements** TestDataProvider {

**public** **static** **final** String ***PROVIDER\_PREFIX*** = "v";

**private** ThreadLocal<Map<String, Object>> contextVariables = **new** ThreadLocal<>();

**private** **static** **final** String ***VARIABLE\_STRING*** = "Variable ";

**public** Object getContextVariable(String key) {

String[] pathElements = key.split("\\.");

**return** getVariableStorageByPath(pathElements).get(pathElements[pathElements.length - 1]);

}

**public** **void** setContextVariable(String key, Object value) {

String[] pathElements = key.split("\\.");

getVariableStorageByPath(pathElements).put(pathElements[pathElements.length - 1], value);

}

**public** Map<String, Object> getVariableStorageByPath(String[] pathElements) {

Map<String, Object> variableStorage = getVariableStorage();

**for** (**int** i = 0; i < pathElements.length - 1; i++) {

Object storageCandidate = variableStorage.get(pathElements[i]);

**if** (storageCandidate == **null**) {

storageCandidate = **new** HashMap<String, Object>();

variableStorage.put(pathElements[i], storageCandidate);

}

**if** (storageCandidate **instanceof** Map) {

variableStorage = (Map) storageCandidate;

} **else** {

**throw** **new** GDPAPIException(

***VARIABLE\_STRING*** + StringUtils.join(Arrays.copyOfRange(pathElements, 0, i), ".") + " class="

+ storageCandidate.getClass().getName() + " already exists and is is not a Map");

}

}

**return** variableStorage;

}

@Override

**public** String getValue(String expression) {

Object value = getContextVariable(expression);

**return** value != **null** ? value.toString() : **null**;

}

@Override

**public** List<String> getList(String expression) {

Object variable = getContextVariable(expression);

**if** (variable **instanceof** String) {

**return** Arrays.asList((String) variable);

}

**if** (variable **instanceof** List) {

**return** (List) variable;

} **else** {

**throw** **new** IllegalArgumentException(***VARIABLE\_STRING*** + expression + " is not list");

}

}

@Override

**public** Map<String, Object> getMap(String expression) {

{

Object variable = getContextVariable(expression);

**if** (variable **instanceof** Map) {

**return** (Map) variable;

} **else** {

**throw** **new** IllegalArgumentException(***VARIABLE\_STRING*** + expression + " is not map");

}

}

}

@Override

**public** **boolean** support(String prefix) {

**return** **false**;

}

**private** Map<String, Object> getVariableStorage() {

**if** (contextVariables.get() == **null**) {

contextVariables.set(**new** HashMap<String, Object>());

}

**return** contextVariables.get();

}

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** java.util.List;

**import** java.util.Map;

**public** **interface** TestDataProvider {

String getValue(String expression);

List<String> getList(String expression);

Map<String, Object> getMap(String expression);

**boolean** support(String prefix);

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** com.fasterxml.jackson.core.JsonProcessingException;

**import** com.fasterxml.jackson.databind.JsonNode;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**import** com.fasterxml.jackson.databind.node.TextNode;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

@Component

**public** **class** JsonTestData **implements** TestDataProvider {

**public** **static** **final** Pattern ***EXPRESSION\_PARAMS*** = Pattern.compile("[a-z]\\{[\\w.\\s]\*\\}");

**public** **static** **final** String ***PROVIDER\_PREFIX*** = "j";

@Autowired

**private** TestDataExpressionResolver expressionResolver;

**private** ObjectMapper objectMapper = **new** ObjectMapper();

**private** ThreadLocal<JsonNode> jsonTestDataTL = **new** ThreadLocal<>();

**public** **void** init(String path) {

**try** {

InputStream inputStream = **this**.getClass().getResourceAsStream(path);

jsonTestDataTL.set(objectMapper.readTree(inputStream));

} **catch** (IOException ex) {

**throw** **new** GDPAPIException(ex.getMessage(), ex);

}

}

**public** JsonNode getNodeByPath(String jsonPath) {

JsonNode jsonNode = jsonTestDataTL.get();

**for** (String path : jsonPath.split("\\.")) {

**if** (jsonNode == **null**) {

**break**;

}

jsonNode = jsonNode.get(path);

}

**return** jsonNode;

}

@Override

**public** String getValue(String jsonPath) {

JsonNode jsonNode = getNodeByPath(buildPath(jsonPath));

**if** (jsonNode **instanceof** TextNode) {

**return** jsonNode.asText();

}

**return** jsonNode != **null** ? jsonNode.toString() : **null**;

}

@Override

**public** List<String> getList(String jsonPath) {

JsonNode jsonNode = getNodeByPath(buildPath(jsonPath));

**if** (jsonNode != **null**) {

List<String> result = **new** ArrayList<>();

Iterator<JsonNode> childIterator = jsonNode.elements();

**while** (childIterator.hasNext()) {

result.add(childIterator.next().toString());

}

**return** result;

} **else** {

**return** **new** ArrayList<>();

}

}

@Override

**public** Map<String, Object> getMap(String jsonPath) {

JsonNode jsonNode = getNodeByPath(buildPath(jsonPath));

**if** (jsonNode != **null**) {

**try** {

**return** objectMapper.treeToValue(jsonNode, Map.**class**);

} **catch** (JsonProcessingException ex) {

**throw** **new** GDPAPIException(ex.getMessage(), ex);

}

} **else** {

**return** **new** HashMap<>();

}

}

@Override

**public** **boolean** support(String prefix) {

**return** ***PROVIDER\_PREFIX***.equals(prefix);

}

**public** String buildPath(String untouchedPath) {

Matcher matcher = ***EXPRESSION\_PARAMS***.matcher(untouchedPath);

String path = untouchedPath;

**while** (matcher.find()) {

String substitute = expressionResolver.resolveExpression(matcher.group());

path = path.replace("(" + matcher.group() + ")", substitute);

}

**return** path;

}

}

//

// Source code recreated from a .class file by IntelliJ IDEA

// (powered by FernFlower decompiler)

//

package com.mastercard.quality.engineering.mtaf.jbehave.steps;

import com.mastercard.quality.engineering.mtaf.jbehave.context.TestContextProvider;

import org.apache.log4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.env.Environment;

public abstract class AbstractSteps {

protected static Logger LOG = Logger.getLogger(AbstractSteps.class);

@Autowired

private TestContextProvider testContextProvider;

@Autowired

protected Environment env;

public AbstractSteps() {

}

protected void putInContext(String key, Object obj) {

this.testContextProvider.get().put(key, obj);

}

protected <T> T getFromContext(String key) {

return this.testContextProvider.get().get(key);

}

protected void removeFromContext(String key) {

this.testContextProvider.get().remove(key);

}

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** java.util.List;

**public** **interface** TestDataExpressionResolver {

String resolveExpression(String expression);

List<String> resolveListExpression(String expression);

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** java.util.List;

**import** java.util.Map;

**public** **interface** TestDataProvider {

String getValue(String expression);

List<String> getList(String expression);

Map<String, Object> getMap(String expression);

**boolean** support(String prefix);

}

**package** com.mastercard.testing.gdp.api.dbconnection;

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.ResultSetMetaData;

**import** java.sql.SQLException;

**import** javax.sql.DataSource;

**import** org.apache.commons.dbcp2.BasicDataSource;

**import** org.apache.commons.dbutils.ResultSetHandler;

**import** org.springframework.beans.factory.annotation.Value;

**public** **class** DataSourceConfig **implements** DataSourceConfigBase {

@Value("${db.url:local}")

**private** String dbUrl;

@Value("${db.user:local}")

**private** String dbUser;

@Value("${db.password:local}")

**private** String dbPassword;

**private** DataSource postgres = **null**;

**private** ResultSetHandler resultSetHandler = **null**;

@Override

**public** DataSource getDataSource() {

**if** (postgres == **null**) {

postgres = createDataSource(dbUrl, dbUser, dbPassword);

}

**return** postgres;

}

**private** DataSource createDataSource(String dbUrl, String dbUser, String dbPassword) {

BasicDataSource dataSource = **new** BasicDataSource();

dataSource.setDriverClassName("org.postgresql.Driver");

dataSource.setUrl(dbUrl);

dataSource.setUsername(dbUser);

**return** dataSource;

}

@Override

**public** Connection getConnecion() **throws** SQLException {

**return** **this**.getDataSource().getConnection();

}

@Override

**public** ResultSetHandler getResultSetHandler() {

**if** (resultSetHandler == **null**) {

resultSetHandler = createResultSetHandler();

}

**return** resultSetHandler;

}

@Override

**public** ResultSetHandler createResultSetHandler() {

**return** **new** ResultSetHandler<Object[]>() {

@Override

**public** Object[] handle(ResultSet recordSet) **throws** SQLException {

**if** (!recordSet.next()) {

**return** **new** Object[0];

}

ResultSetMetaData meta = recordSet.getMetaData();

**int** cols = meta.getColumnCount();

Object[] result = **new** Object[cols];

**for** (**int** i = 0; i < cols; i++) {

result[i] = recordSet.getObject(i + 1);

}

**return** result;

}

};

}

}

**package** com.mastercard.testing.gdp.api.dbconnection;

**import** java.sql.Connection;

**import** java.sql.SQLException;

**import** javax.sql.DataSource;

**import** org.apache.commons.dbutils.ResultSetHandler;

**public** **interface** DataSourceConfigBase {

**public** DataSource getDataSource();

**public** Connection getConnecion() **throws** SQLException;

**public** ResultSetHandler getResultSetHandler();

**public** ResultSetHandler createResultSetHandler();

}

**package** com.mastercard.testing.gdp.api.dbconnection;

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.ResultSetMetaData;

**import** java.sql.SQLException;

**import** javax.sql.DataSource;

**import** org.apache.commons.dbcp2.BasicDataSource;

**import** org.apache.commons.dbutils.ResultSetHandler;

**import** org.springframework.beans.factory.annotation.Value;

**public** **class** DataSourceConfigOracle **implements** DataSourceConfigBase {

@Value("${oracle.db.url:local}")

**private** String dbUrl;

@Value("${oracle.db.user:local}")

**private** String dbUser;

@Value("${oracle.db.password:local}")

**private** String dbPassword;

@Value("${oracle.db.schema:local}")

**private** String dbSchema;

**private** DataSource oracle = **null**;

**private** ResultSetHandler resultSetHandler = **null**;

@Override

**public** DataSource getDataSource() {

**if** (oracle == **null**) {

**return** createDataSource(dbUrl, dbUser, dbPassword, dbSchema);

}

**return** oracle;

}

@Override

**public** Connection getConnecion() **throws** SQLException {

**return** **this**.getDataSource().getConnection();

}

@Override

**public** ResultSetHandler getResultSetHandler() {

**if** (resultSetHandler == **null**) {

**return** createResultSetHandler();

}

**return** resultSetHandler;

}

@Override

**public** ResultSetHandler createResultSetHandler() {

**return** **new** ResultSetHandler<Object[]>() {

@Override

**public** Object[] handle(ResultSet recordSet) **throws** SQLException {

**if** (!recordSet.next()) {

**return** **new** Object[0];

}

ResultSetMetaData meta = recordSet.getMetaData();

**int** cols = meta.getColumnCount();

Object[] result = **new** Object[cols];

**for** (**int** i = 0; i < cols; i++) {

result[i] = recordSet.getObject(i + 1);

}

**return** result;

}

};

}

**private** DataSource createDataSource(String dbUrl, String dbUser, String dbPassword, String dbSchema) {

BasicDataSource dataSource = **new** BasicDataSource();

dataSource.setDriverClassName("oracle.jdbc.OracleDriver");

dataSource.setUsername(dbUser);

dataSource.setPassword(dbPassword);

dataSource.setDefaultSchema(dbSchema);

**return** dataSource;

}

}

**package** com.mastercard.testing.gdp.api.dbconnection;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.InputStreamReader;

**import** java.io.Reader;

**import** java.nio.charset.StandardCharsets;

**import** java.sql.Array;

**import** java.sql.Blob;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.commons.dbutils.QueryRunner;

**import** org.apache.commons.dbutils.ResultSetHandler;

**import** org.apache.commons.dbutils.handlers.ColumnListHandler;

**import** org.jbehave.core.annotations.AfterStories;

**import** org.jbehave.core.annotations.BeforeStories;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** com.ibatis.common.jdbc.ScriptRunner;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

**import** io.restassured.internal.util.IOUtils;

**public** **class** DBOperationBaseUtils {

**protected** QueryRunner queryRunner;

**protected** Connection conn = **null**;

**protected** PreparedStatement pstmt = **null**;

**protected** ResultSet rs = **null**;

**private** **static** **final** Logger ***logger*** = LoggerFactory.getLogger(DBOperationBaseUtils.**class**);

**protected** DataSourceConfigBase dataSourceConfigBase;

@AfterStories

**public** **void** closeDatabaseConnection() {

**try** {

***logger***.info("Closed the connection {}", conn);

conn.close();

} **catch** (SQLException sqlError) {

***logger***.info("SQL exception occurred {}", sqlError);

}

}

@BeforeStories

**public** **void** establishDatabaseConnection() {

**try** {

Connection conn = dataSourceConfigBase.getDataSource().getConnection();

queryRunner = **new** QueryRunner(dataSourceConfigBase.getDataSource());

***logger***.info("Connection object is {}", conn);

} **catch** (Exception sqlError) {

***logger***.info("SQL exception occurred {}", sqlError);

}

}

@SuppressWarnings("unchecked")

**public** **void** executeInsertStatement(String query) {

***logger***.info("INSERT\_QUERY :{}", query);

**try** {

queryRunner.insert(query, dataSourceConfigBase.getResultSetHandler());

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

}

**public** **void** performUpdate(String query) {

***logger***.info("UPDATE\_QUERY :{}", query);

**try** {

queryRunner.update(query);

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

}

**public** **int** executeDeleteStatement(String query) {

***logger***.info("DELETE\_QUERY :{}", query);

**return** executeUpdate(query);

}

**public** **int** executeUpdateStatement(String query) {

***logger***.info("UPDATE\_QUERY : {}", query);

**return** executeUpdate(query);

}

**private** **int** executeUpdate(String query) {

**int** numberOfRecordsEffected = -1;

**try** {

queryRunner.insert(query, dataSourceConfigBase.getResultSetHandler());

numberOfRecordsEffected = queryRunner.update(query);

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

**return** numberOfRecordsEffected;

}

**public** **int** getCountOfRecords(String query) {

**int** numberOfRecords = -1;

**try** {

Object[] result = (Object[]) queryRunner.query(query, dataSourceConfigBase.getResultSetHandler());

**long** recordsCount = (**long**) result[0];

numberOfRecords = (**int**) recordsCount;

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

**return** numberOfRecords;

}

**public** String getSingleDbValue(String query) {

***logger***.info("Query : {}", query);

String value = **null**;

**try** {

Object[] result = (Object[]) queryRunner.query(query, dataSourceConfigBase.getResultSetHandler());

**if** (result != **null** || result[0] != **null**) {

value = result[0].toString();

}

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

***logger***.info("Value fetched from database is {}", value);

**return** value;

}

**public** String getBlobValue(String query, String columnLabel) {

String blobValue = **null**;

**try** {

rs = dataSourceConfigBase.getConnecion().prepareStatement(query).executeQuery();

rs.next();

Blob myBlob = rs.getBlob(columnLabel);

**byte**[] byteArray = IOUtils.toByteArray(myBlob.getBinaryStream());

blobValue = **new** String(byteArray, StandardCharsets.***UTF\_8***);

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

} **catch** (IOException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

***logger***.info("Value fetched from database is {}", blobValue);

**return** blobValue;

}

**public** <T> List<T> getAllDbValuesForAColumn(String query, String columnName) {

List<T> returnList = **null**;

ColumnListHandler<T> handler = **new** ColumnListHandler<>(columnName);

**try** {

returnList = queryRunner.query(query, handler);

} **catch** (SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

**return** returnList;

}

**public** <T> List<T> getCompleteRowData(String query, Class clazz) {

Object obj;

**try** {

obj = clazz.newInstance();

**return** queryRunner.query(query, (ResultSetHandler<List<T>>) obj);

} **catch** (InstantiationException | IllegalAccessException | SQLException e) {

**throw** **new** GDPAPIException(e.getMessage(), e);

}

}

**public** **void** executeSqlScript(String sqlFilePath) {

**try** {

ScriptRunner runner = **new** ScriptRunner(conn, **false**, **true**);

InputStream stream = **this**.getClass().getResourceAsStream(sqlFilePath);

Reader reader = **new** InputStreamReader(stream, StandardCharsets.***UTF\_8***.name());

runner.runScript(reader);

conn.setAutoCommit(**false**);

conn.commit();

} **catch** (IOException | SQLException e) {

***logger***.error(String.format("Failed to Execute %s %s %s", sqlFilePath, " \nThe error is ", e));

**try** {

conn.rollback();

} **catch** (SQLException sqlException) {

***logger***.error("SQL Rollback exception :", sqlException);

}

**throw** **new** GDPAPIException(e.getMessage(), e);

}

}

**public** Map<String, Object> executeSelectStmt(String query, List<String> keys) {

**try** {

**try** (PreparedStatement stmt = conn.prepareStatement(query)) {

Array array = conn.createArrayOf("text", keys.toArray(**new** String[0]));

stmt.setArray(1, array);

DriverManager.registerDriver(**new** org.postgresql.Driver());

ResultSet resultSet = stmt.executeQuery();

**return** prepareKeyValuePairs(resultSet);

}

} **catch** (SQLException e) {

***logger***.error("SQL exception occurred ", e);

**throw** **new** GDPAPIException(e.getMessage(), e);

}

}

**public** Map<String, Object> prepareKeyValuePairs(ResultSet rs) **throws** SQLException {

Map<String, Object> row = **new** HashMap<>();

**while** (rs.next()) {

row.put(rs.getString(1).trim(), rs.getString(2));

}

**return** row;

}

}

**package** com.mastercard.testing.gdp.api.dbconnection;

**import** javax.annotation.PostConstruct;

**import** org.springframework.beans.factory.annotation.Autowired;

**public** **class** DBOperationOracleUtils **extends** DBOperationBaseUtils {

@Autowired

**private** DataSourceConfigOracle dataSourceConfigOracle;

@PostConstruct

**public** **void** init() {

**super**.dataSourceConfigBase = **this**.dataSourceConfigOracle;

}

}

**package** com.mastercard.testing.gdp.api.dbconnection;

**import** javax.annotation.PostConstruct;

**import** org.apache.commons.lang3.BooleanUtils;

**import** org.jbehave.core.annotations.BeforeStories;

**import** org.jbehave.core.annotations.BeforeStory;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Value;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

**import** com.mastercard.testing.gdp.api.helper.constants.QueryConstants;

**public** **class** DBOperationUtils **extends** DBOperationBaseUtils {

@Value("${truncatePrivacyRequest:false}")

**private** String truncateTableSwitch;

@Value("${metaFilters:local}")

**private** String metaFilters;

@Autowired

**private** DataSourceConfig dataSourceConfig;

@PostConstruct

**public** **void** init() {

**super**.dataSourceConfigBase = **this**.dataSourceConfig;

}

**private** **static** **final** Logger ***logger*** = LoggerFactory.getLogger(DBOperationUtils.**class**);

@BeforeStory

**public** **void** truncatePrivacyRequestTables() {

**try** {

**if** (***logger***.isDebugEnabled()) {

***logger***.info(String.format("Count of privacy requests %s",

getSingleDbValue(QueryConstants.***PRIVACY\_REQUEST\_COUNT***)));

***logger***.info(String.format("Count of service function privacy request %s",

getSingleDbValue(QueryConstants.***SERVICE\_FUNCTION\_PRIVACY\_COUNT***)));

}

**if** (BooleanUtils.toBoolean(truncateTableSwitch)) {

performUpdate(QueryConstants.***TRUNCATE\_SERVICE\_FUNCTION\_PRIVACY\_REQUEST***);

performUpdate(QueryConstants.***TRUNCATE\_SERVICE\_FUNCTION\_INTEGRATION***);

performUpdate(QueryConstants.***TRUNCATE\_PRIVACY\_REQUEST\_ACTION***);

performUpdate(QueryConstants.***TRUNCATE\_PRIVACY\_REQUEST***);

}

} **catch** (Exception exception) {

**throw** **new** GDPAPIException(exception.toString());

}

}

@BeforeStories

**public** **void** truncatePrivacyRequestTablesBeforeBatch() {

**if** (metaFilters.contains("+jaguarsTeam") && metaFilters.contains("+eaglesTeam")) {

performUpdate(QueryConstants.***TRUNCATE\_PRIVACY\_REQUEST***);

performUpdate(QueryConstants.***TRUNCATE\_PRODUCT\_DEF***);

performUpdate(QueryConstants.***TRUNCATE\_SERVICE\_FUNCTION\_INTEGRATION***);

}

}

}

**package** com.mastercard.testing.gdp.api.domain;

**import** lombok.Data;

**import** org.apache.commons.lang3.builder.EqualsBuilder;

**import** org.apache.commons.lang3.builder.HashCodeBuilder;

**import** org.apache.commons.lang3.builder.ReflectionToStringBuilder;

**import** org.apache.commons.lang3.builder.ToStringStyle;

@Data

**public** **class** DefaultPOJO {

@Override

**public** **int** hashCode() {

**return** HashCodeBuilder.*reflectionHashCode*(3, 31, **this**, **false**);

}

@Override

**public** **boolean** equals(Object obj) {

**return** EqualsBuilder.*reflectionEquals*(**this**, obj, **false**, **this**.getClass());

}

@Override

**public** String toString() {

**return** ReflectionToStringBuilder.*toString*(**this**, ToStringStyle.***SHORT\_PREFIX\_STYLE***);

}

}

**package** com.mastercard.testing.gdp.api.domain;

**import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;

**import** com.fasterxml.jackson.annotation.JsonProperty;

**import** lombok.Data;

@JsonIgnoreProperties(ignoreUnknown = **true**)

@Data

**public** **class** ApiError **extends** DefaultPOJO {

@JsonProperty("code")

**private** String code;

@JsonProperty("message")

**private** String message;

@JsonProperty("status")

**private** String status;

@JsonProperty("more")

**private** String more;

@JsonProperty("field")

**private** String field;

@JsonProperty("gdpErrorCode")

**private** String gdpErrorCode;

@JsonProperty("apiErrors")

**private** Object apiErrors;

}

**package** com.mastercard.testing.gdp.api.domain;

**import** org.apache.commons.lang3.builder.EqualsBuilder;

**import** org.apache.commons.lang3.builder.HashCodeBuilder;

**import** org.apache.commons.lang3.builder.ReflectionToStringBuilder;

**import** org.apache.commons.lang3.builder.ToStringStyle;

**import** lombok.Data;

@Data

**public** **class** DefaultPOJO {

@Override

**public** **int** hashCode() {

**return** HashCodeBuilder.reflectionHashCode(3, 31, **this**, **false**);

}

@Override

**public** **boolean** equals(Object obj) {

**return** EqualsBuilder.reflectionEquals(**this**, obj, **false**, **this**.getClass());

}

@Override

**public** String toString() {

**return** ReflectionToStringBuilder.toString(**this**, ToStringStyle.SHORT\_PREFIX\_STYLE);

}

}

**package** com.mastercard.testing.gdp.api.domain;

**import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;

**import** com.fasterxml.jackson.annotation.JsonProperty;

**import** lombok.Data;

@JsonIgnoreProperties(ignoreUnknown = **true**)

@Data

**public** **class** ErrorMessage **extends** DefaultPOJO {

@JsonProperty("code")

**private** String code;

@JsonProperty("message")

**private** String message;

@JsonProperty("status")

**private** String status;

@JsonProperty("more")

**private** String more;

@JsonProperty("field")

**private** String field;

@JsonProperty("timestamp")

**private** String timestamp;

@JsonProperty("path")

**private** String path;

@JsonProperty("error")

**private** String error;

@JsonProperty("gdpErrorCode")

**private** String gdpErrorCode;

}

**package** com.mastercard.testing.gdp.api.domain;

**import** java.util.List;

**import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;

**import** com.fasterxml.jackson.annotation.JsonProperty;

**import** lombok.Data;

@JsonIgnoreProperties(ignoreUnknown = **true**)

@Data

**public** **class** ErrorResponse **extends** DefaultPOJO {

@JsonProperty("code")

**private** String code;

@JsonProperty("message")

**private** String message;

@JsonProperty("status")

**private** String status;

@JsonProperty("more")

**private** String more;

@JsonProperty("field")

**private** String field;

@JsonProperty("gdpErrorCode")

**private** String gdpErrorCode;

@JsonProperty("apiErrors")

**private** List<ApiError> apiErrors;

}

**package** com.mastercard.testing.gdp.api.exception;

**public** **class** GDPAPIException **extends** RuntimeException {

**public** GDPAPIException(String message) {

**super**(message);

}

**public** GDPAPIException(String message, Exception exp) {

**super**(message, exp);

}

}

**package** com.mastercard.testing.gdp.api.helper;

**import** java.util.Arrays;

**public** **class** ResponseConverter<T> {

**private** org.apache.http.Header[] headers = **null**;

**private** org.apache.http.StatusLine statusLine = **null**;

**private** String responseBody = **null**;

**private** T formattedResponseBody = **null**;

**private** org.apache.http.Header contentType;

**public** T getFormattedResponseBody() {

**return** formattedResponseBody;

}

**public** **void** setFormattedResponseBody(T formattedResponseBody) {

**this**.formattedResponseBody = formattedResponseBody;

}

**public** org.apache.http.Header getContentType() {

**return** contentType;

}

**protected** **void** setContentType(org.apache.http.Header header) {

**this**.contentType = header;

}

**public** org.apache.http.Header[] getHeaders() {

**return** headers;

}

**protected** **void** setHeaders(org.apache.http.Header[] headers2) {

**this**.headers = headers2;

}

**public** org.apache.http.StatusLine getStatusLine() {

**return** statusLine;

}

**protected** **void** setStatusLine(org.apache.http.StatusLine statusLine2) {

**this**.statusLine = statusLine2;

}

**public** String getResponseBody() {

**return** responseBody;

}

**protected** **void** setResponseBody(String responseBody) {

**this**.responseBody = responseBody;

}

@Override

**public** String toString() {

**return** "Response [headers=" + Arrays.toString(headers) + ", statusLine=" + statusLine + ", responseBody="

+ responseBody + ", formattedResponseBody=" + formattedResponseBody + ", ContentType=" + contentType

+ "]";

}

}

**package** com.mastercard.testing.gdp.api.helper.constants;

**import** java.nio.charset.StandardCharsets;

**public** **final** **class** Constants {

**public** **static** **final** String ***COMMA\_SEPARATOR*** = "\\s\*,\\s\*";

**public** **static** **final** String ***SERVICE\_FUNCTION\_CODE*** = "serviceFunctionCode";

**public** **static** **final** String ***REQUEST\_ID*** = "requestId";

**public** **static** **final** String ***ERROR\_CODE*** = "errorCode";

**public** **static** **final** String ***REQUEST\_STATUS\_CODE*** = "requestStatusCode";

**public** **static** **final** String ***RESPONSE\_CODE*** = "responseCode";

**public** **static** **final** String ***ENCRYPTED\_CODE*** = "encryptedKey";

**public** **static** **final** String ***EMAIL\_PARAMETERS*** = "emailParameters";

**public** **static** **final** String ***URL\_SEPARATOR*** = ":";

**public** **static** **final** String ***MERGE\_DATA*** = "mergeData";

**public** **static** **final** String ***URL*** = "url";

**public** **static** **final** String ***PRODUCT\_NAME*** = "productName";

**public** **static** **final** String ***SERVICE\_FUNCTION\_NAME*** = "serviceFunctionName";

**public** **static** **final** String ***APPLICATION\_DATA*** = "applicationData";

**public** **static** **final** String ***MESSAGE\_DATA*** = "messageData";

**public** **static** **final** String ***SITE\_ID*** = "siteID";

**public** **static** **final** String ***CAMPAIGN\_ID*** = "campaignID";

**public** **static** **final** String ***TEMPLATE*** = "template";

**public** **static** **final** String ***SUBJECT*** = "subject";

**public** **static** **final** String ***RECIPIENT\_EMAIL*** = "recipientEmail";

**public** **static** **final** String ***SENDER\_EMAIL*** = "senderEmail";

**public** **static** **final** String ***GDP\_WEB\_MANUALRESPONSE\_REQUEST\_ID*** = "/gdp-web/manualresponse?requestId=";

**public** **static** **final** String ***FUNCTION\_CODE*** = "&functionCode=";

**public** **static** **final** String ***ENCRYPTED\_KEY*** = "&encryptedKey=";

**public** **static** **final** String ***ENCRYPTED\_KEY\_SEPARATOR*** = "-";

**public** **static** **final** String ***EMAIL\_TEXT\_REGEX*** = "^[\\w!#$%&'\*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'\*+/=?`{|}~^-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$";

**public** **static** **final** String ***YYYY\_MM\_DD*** = "yyyy-MM-dd";

**public** **static** **final** **int** ***RADIX*** = 16;

**public** **static** **final** **int** ***SIGNUM*** = 1;

**public** **static** **final** String ***APPLICATION\_JSON*** = "application/json";

**public** **static** **final** String ***HEADER\_NAME\_CONTENT\_TYPE*** = "Content-Type";

**public** **static** **final** String ***CORRELATION\_ID*** = "Correlation-ID";

**public** **static** **final** String ***HEADER\_NAME\_ACCCEPT*** = "Accept";

**public** **static** **final** String ***ZONE\_ID*** = "Canada/Central";

**public** **static** **final** String ***PARAMETER\_SEPARATOR\_AMPERSAND*** = "&";

**public** **static** **final** String ***PARAMETER\_SEPARATOR\_QUESTION\_MARK*** = "?";

**public** **static** **final** String ***HEADER\_SEPARATOR\_EQUAL\_TO*** = "=";

**public** **static** **final** String ***BUSINESS\_REVIEW\_DATA\_BEAN\_HEADER*** = "BusinessReviewDataBean";

**public** **static** **final** String ***CONSUMER\_REVIEW\_DATA\_BEAN\_HEADER*** = "ConsumerReviewDataBean";

**public** **static** **final** String ***INPUT\_TYPE*** = "inputType";

**public** **static** **final** String ***UNIQUE\_KEY\_SEPARATOR*** = "\_";

**public** **static** **final** String ***OK*** = "OK";

**public** **static** **final** **int** ***API\_RESPONSE\_OK*** = 200;

**public** **static** **final** String ***PRIVACY\_REQUEST\_STATUS*** = "I";

**public** **static** **final** String ***NULL*** = "null";

**public** **static** **final** String ***CALLING\_THE\_API\_WITH\_THE\_RESOURCE\_PATH*** = "Calling the API with the resource path : ";

**public** **static** **final** **int** ***DELTA\_IN\_MILLISECONDS*** = 40000;

**public** **static** **final** **int** ***MILLISECONDS\_FOR\_3\_SECONDS*** = 3000;

**public** **static** **final** String ***PATH\_PARAMETER\_DELIMITER*** = "/";

**public** **static** **final** String ***PRIVACY\_REQUEST\_ID\_PATH\_PARAMETER*** = "?privacyRequestId=";

**public** **static** **final** String ***SERVICE\_FUNCTION\_CODE\_PATH\_PARAMETER*** = "&serviceFunctionCode=";

**public** **static** **final** String ***ALGORITHM*** = "MD5";

**public** **static** **final** String ***SALT\_STRING*** = "GDPSECURESALTKEY";

**public** **static** **final** String ***RQST\_ENTRY*** = "RQST\_ENTRY";

**public** **static** **final** String ***RESP\_RECVD*** = "RESP\_RECVD";

**public** **static** **final** String ***RQST\_SENT*** = "RQST\_SENT";

**public** **static** **final** String ***PRDCT\_CD*** = "prdct\_cd";

**public** **static** **final** String ***PRDCT\_PRVCY\_RQST\_ID*** = "prdct\_prvcy\_rqst\_id";

**public** **static** **final** String ***RESPONSE\_RECORD\_NAME\_SEPARATOR*** = "#";

**public** **static** **final** String ***SENDER\_NAME*** = "senderName";

**public** **static** **final** String ***REPLY\_TO*** = "replyTo";

**public** **static** **final** String ***RETURN\_PATH*** = "returnPath";

**public** **static** **final** String ***CONTACT\_INFO*** = "contactInfo";

**public** **static** **final** String ***RECIPIENT\_FIRSTNAME*** = "recipientFirstname";

**public** **static** **final** String ***RECIPIENT\_LASTNAME*** = "recipientLastname";

**public** **static** **final** String ***PROCESSING\_PARAMETERS*** = "processingParameters";

**public** **static** **final** String ***SITE\_ID\_VALUE*** = "822";

**public** **static** **final** String ***M5\_BVM7*** = "1-1M5BVM7";

**public** **static** **final** String ***YES*** = "Y";

**public** **static** **final** String ***REPORTING*** = "reporting";

**public** **static** **final** String ***EMAIL\_DELIVERY\_CHANNEL*** = "emailDeliveryChannel";

**public** **static** **final** String ***MASTER\_CARD*** = "MASTER\_CARD";

**public** **static** **final** String ***CLIENT\_MESSAGE\_ID*** = "clientMessageID";

**public** **static** **final** String ***RQST\_PBLSH*** = "RQST\_PBLSH";

**public** **static** **final** String ***CHAR\_SET*** = StandardCharsets.***UTF\_8***.name();

**public** **static** **final** String ***DATE\_SUBMITTED*** = "dateSubmitted";

**public** **static** **final** String ***PARAM\_SEPARATOR*** = "/";

**public** **static** **final** String ***PRODUCT\_STATUS\_CHANGE*** = "PRD\_ST\_CHG";

**public** **static** **final** String ***AUD\_DESC*** = "aud\_desc";

**public** **static** **final** String ***PRODUCT\_REQUEST\_STATUS\_CHANGED\_TO*** = "Product Request status changed to";

**public** **static** **final** String ***EMAIL\_CURRENT\_TIME\_REGEX*** = "MM/dd/yy hh:mm:ss";

**public** **static** **final** String ***ERROR\_EMAIL\_CURRENT\_TIME*** = "createdDate";

**public** **static** **final** String ***DUE\_DAYS*** = "dueDays";

**public** **static** **final** String ***DUE\_DAYS\_MANUAL\_VALUE*** = "5";

**public** **static** **final** String ***DUE\_DAYS\_ERROR\_MESSAGE\_VALUE*** = "2";

**public** **static** **final** String ***REPLY\_EMAIL*** = "mailTo";

**public** **static** **final** String ***REPLY\_EMAIL\_VALUE*** = "biz-ops@mailbox.com";

**public** **static** **final** String ***PHONE\_NUMBER\_VALUE*** = "1-800-000-0000";

**public** **static** **final** String ***PHONE\_NUMBER*** = "callTo";

**public** **static** **final** String ***MANUAL\_RESPONSE\_SUBJECT\_LINE*** = "ACTION REQUIRED: GDPR Individual Data Access Request#";

**public** **static** **final** String ***ERROR\_RESPONSE\_SUBJECT\_LINE*** = "ACTION REQUIRED: Error encountered validating response for Request#";

**public** **static** **final** String ***ERROR\_TEMPLATE*** = "822/822\_GDPR\_error-email-template.xsl";// Needs to be modified

**public** **static** **final** String ***MANUAL\_TEMPLATE*** = "822/822\_GDPR\_en\_US.xsl";

**public** **static** **final** String ***ERROR\_DESCRIPTION*** = "errorData";

**public** **static** **final** String ***ERROR\_KEY*** = "errorKey";

**public** **static** **final** String ***ERROR\_VALUE*** = "errorValue";

**public** **static** **final** String ***COMPLETED\_DATE*** = "dateCompleted";

**public** **static** **final** String ***STLOUIS\_TIMEZONE*** = "UTC-06:00";

**public** **static** **final** String ***IN\_PROGRESS*** = "I";

**public** **static** **final** String ***COMPLETED*** = "C";

**public** **static** **final** String ***REQ\_TYPE\_DELETE*** = "D";

**public** **static** **final** String ***REQ\_TYPE\_UPDATE*** = "U";

**public** **static** **final** String ***REQ\_TYPE\_VIEW*** = "V";

**public** **static** **final** String ***SERV\_INTG\_TYPE\_PUBSUB*** = "P";

**public** **static** **final** String ***SERV\_INTG\_TYPE\_MANUAL*** = "M";

**public** **static** **final** String ***SERV\_INTG\_TYPE\_API*** = "A";

**public** **static** **final** String ***BUSINESS\_CNTXT\_B2B*** = "B2B";

**public** **static** **final** String ***BUSINESS\_CNTXT\_B2C*** = "B2C";

**public** **static** **final** String ***BUSINESS\_CNTXT\_B2E*** = "B2E";

**public** **static** **final** String ***DOMAIN\_BUSINESS*** = "Business";

**public** **static** **final** String ***DOMAIN\_CONSUMER*** = "Consumer";

**public** **static** **final** String ***INSERT\_INTO\_GDP\_OWNER\_RESP\_REC*** = "INSERT INTO resp\_rec(\r\n";

**public** **static** **final** String ***RESP\_REC\_ID\_SERV\_FUNC\_PRVCY\_RQST\_ID\_PRDCT\_PRVCY\_RQST\_ID\_RESP\_REC\_NAM\_RESP\_DATA\_TXT*** = " resp\_rec\_id, serv\_func\_prvcy\_rqst\_id, prdct\_prvcy\_rqst\_id, resp\_rec\_nam, resp\_data\_txt, \r\n";

**public** **static** **final** String ***RESP\_REC\_STAT\_CD\_RESP\_RVW\_TS\_DSCLSR\_SW\_DEL\_SW\_DEL\_PRDCT\_RQST\_ID\_HOLD\_RLSE\_DT*** = " resp\_rec\_stat\_cd, resp\_rvw\_ts, dsclsr\_sw, del\_sw, del\_prdct\_rqst\_id, hold\_rlse\_dt)\r\n";

**public** **static** **final** String ***RESPONSE\_RECORD\_ID*** = "responseRecordId";

**public** **static** **final** String ***PRODUCTS*** = "products";

**public** **static** **final** String ***PRODUCT\_PRIVACY\_REQUEST\_ID*** = "productPrivacyRequestId";

**public** **static** **final** String ***SERVICE\_FUNCTIONS*** = "serviceFunctions";

**public** **static** **final** String ***SERVICE\_FUNCTION\_PRIVACY\_REQUEST\_ID*** = "serviceFunctionPrivacyRequestId";

**public** **static** **final** String ***RECORDS*** = "records";

**public** **static** **final** String ***RECORD\_ID*** = "recordId";

**public** **static** **final** String ***PRIVACY\_REQUEST\_ID*** = "privacyRequestId";

**public** **static** **final** String ***PRODUCT\_IDS*** = "productIds";

**public** **static** **final** String ***FILTER\_BY*** = "filterBy";

**public** **static** **final** String ***LOCALE*** = "locale";

**public** **static** **final** String ***APP\_AUTH\_USERNAME\_GDPUSER*** = "app.auth.username.gdpuser";

**public** **static** **final** String ***APP\_AUTH\_KEY*** = "app.auth.password";

**public** **static** **final** String ***DATE\_FORMAT\_US\_CST*** = "dd MMM yyyy 'at' hh:mm a 'US CST'";

**public** **static** **final** String ***RESP\_REC\_STAT\_CD\_RESP\_RVW\_TS\_DSCLSR\_SW\_DEL\_SW\_DEL\_PRDCT\_RQST\_ID\_HOLD\_RLSE\_DT\_DEL\_PRVCY\_RQST\_ID*** = " resp\_rec\_stat\_cd, resp\_rvw\_ts, dsclsr\_sw, del\_sw, del\_prdct\_rqst\_id, hold\_rlse\_dt, del\_prvcy\_rqst\_id)\r\n";

**public** **static** **final** String ***DEFAULT\_LANG*** = "en-us";

**public** **static** **final** String ***APOSTROPHE*** = "'";

**public** **static** **final** String ***COMMA*** = ",";

**public** **static** **final** String ***GET*** = "GET";

**public** **static** **final** String ***DELETE*** = "DELETE";

**public** **static** **final** String ***POST*** = "POST";

**public** **static** **final** String ***AWAITING\_RESPONSE\_STATUS*** = "A";

**public** **static** **final** String ***AUTHORIZATION*** = "Authorization";

**public** **static** **final** String ***PRODUCT\_PRIVACY\_REQ\_ID*** = "productPrivacyReqId";

**public** **static** **final** String ***RESP\_REC\_STAT\_CD\_RESP\_RVW\_TS\_DSCLSR\_SW\_DEL\_SW\_DEL\_PRDCT\_RQST\_ID\_HOLD\_RLSE\_DT\_DEL\_PRV\_RQST\_ID*** = " resp\_rec\_stat\_cd, resp\_rvw\_ts, dsclsr\_sw, del\_sw, del\_prdct\_rqst\_id, hold\_rlse\_dt, del\_serv\_func\_rqst\_id, del\_prvcy\_rqst\_id)\r\n";

**public** **static** **final** String ***EMAIL\_SITE\_ID*** = "822";

**public** **static** **final** String ***EMAIL\_CAMPAIGN\_ID*** = "1-1M5BVM7";

**public** **static** **final** String ***SERVICE\_FUNCTION\_PRIVACY\_REQUEST\_ID\_COLUMN*** = "serv\_func\_prvcy\_rqst\_id";

**public** **static** **final** String ***REVIEWED\_TIME*** = "reviewdTime";

**public** **static** **final** String ***NUM\_DAYS\_IN\_QUEUE*** = "noOfDaysInQueue";

**public** **static** **final** String ***OFFSET*** = "offset";

**public** **static** **final** String ***LIMIT*** = "limit";

**public** **static** **final** String ***SORT\_ORDER*** = "sortorder";

**public** **static** **final** String ***COLUMN\_NAME*** = "columnname";

**public** **static** **final** String ***PRIVACY\_REQUEST\_STATUS\_READY\_TO\_PROCESS*** = "AP";

**public** **static** **final** String ***PRIVACY\_REQUEST\_STATUS\_PENDING\_REVIEW*** = "PR";

**public** **static** **final** String ***AUDIT\_HIST\_AUD\_TYPE\_CD\_REQ\_UPDATE*** = "RQST\_UPDT";

**public** **static** **final** String ***EMPTY\_REQUEST\_BODY*** = "{}";

**public** **static** **final** String ***UPDATE\_REQUEST\_TYPE*** = "U";

**public** **static** **final** String ***RQST\_DELETE\_ALL*** = "RQST\_DA";

**public** **static** **final** String ***REQUEST\_TYPE\_VIEW*** = "View";

**public** **static** **final** String ***REQUEST\_TYPE\_DELETE\_ALL*** = "Delete All";

**public** **static** **final** String ***PRIVACY\_NOTE*** = "privacyNotice";

**public** **static** **final** String ***PRIVACY\_NOTE\_VALUE*** = "https://www.mastercard.us/en-us/about-mastercard/what-we-do/privacy.html";

**public** **static** **final** String ***GDP\_B2B\_WEB\_MANUALRESPONSE\_REQUEST\_ID*** = "/gdp-b2b-web/manualresponse?requestId=";

**public** **static** **final** String ***INTERNAL*** = "Internal";

**public** **static** **final** String ***EXTERNAL*** = "External";

**public** **static** **final** String ***DISPATCH\_TYPE*** = "dispatchType";

**public** **static** **final** String ***DECLINED\_DATE*** = "declinedTimeStamp";

**public** **static** **final** String ***SPACE*** = " ";

**public** **static** **final** String ***SEARCH\_KEY*** = "searchKey";

**public** **static** **final** String ***FALSE*** = "false";

**public** **static** **final** String ***TRUE*** = "true";

**public** **static** **final** String ***DB\_TIME\_REGEX*** = "yyyy-MM-dd HH:mm:ss";

**public** **static** **final** String ***TIME\_ZONE*** = "America/Chicago";

**public** **static** **final** String ***DOUBLE\_ENCODER\_OF\_SLASH*** = "%252F";

**public** **static** **final** String ***FAILED\_API\_RESPONSE\_IS*** = "Failed API response is: ";

**public** **static** **final** String ***DATA\_ACCESS\_SYSTEM*** = "Data Access System";

**public** **static** **final** String ***GDP\_SUPER\_USER\_NAME*** = "GDP Super User";

**public** **static** **final** String ***GDP\_SUPER\_USER\_ID*** = "gdpsuperuser";

**public** **static** **final** String ***COLON\_SEPARATOR*** = "\\s\*:\\s\*";

**public** **static** **final** String ***COLON*** = ":";

**public** **static** **final** String ***PRVCY\_REQ\_IDENTITY\_PROOF\_ID*** = "privacyRequestIdentityProofId";

**public** **static** **final** String ***REVEIEWER\_USER\_ID*** = "reviewerUserId";

**public** **static** **final** String ***LOG\_APPROVE\_FLAG*** = "Verifying is Approve flag";

**public** **static** **final** String ***APPROVE*** = "approve";

**public** **static** **final** String ***DECL\_REASON\_COMMENT*** = "declineReasonComment";

**public** **static** **final** String ***PUBLISHED\_DOCUMENTS*** = "Published:\n";

**public** **static** **final** String ***INTERNAL\_COMMENT\_TEXT\_IS\_INCORRECT*** = "Internal Comment Text is incorrect";

**public** **static** **final** String ***NEW\_LINE\_SEPARATOR*** = "\n";

**public** **static** **final** String ***DOCUMENT\_DOWNLOAD\_PATH*** = "./src/main/resources/config/TestData/documentDownload/%s";

**public** **static** **final** String ***EXTERNAL\_COMMENTS*** = "Message to Requester on review";

**public** **static** **final** String ***INTERNAL\_COMMENTS*** = "Add Internal Comments on review";

// Document upload constants

**public** **static** **final** String ***UPLOAD\_DESCRIPTION*** = "uploadDescription";

**public** **static** **final** String ***FILE*** = "file";

**public** **static** **final** String ***DOCUMENT\_SCOPE*** = "documentScope";

**public** **static** **final** String ***APPRV\_REASON\_COMMENT*** = "Additional Description: Dear Requester - We have verified your identity and approved. You may go ahead and access the documents.";

**public** **static** **final** String ***APPRV\_REASON\_COMMENT\_2*** = "Kind Regards.";

**public** **static** **final** String ***LABEL\_IDEN\_TYPE*** = "Identification Type: ";

**public** **static** **final** String ***COMMENTS*** = "comments";

**public** **static** **final** String ***ID\_PROOF\_TYPE\_CODE*** = "idProofTypeCode";

**public** **static** **final** String ***OTHER\_ID\_PROOF\_NAME*** = "otherIdProofName";

**public** **static** **final** String ***FIRST\_NAME*** = "firstName";

**public** **static** **final** String ***LAST\_NAME*** = "lastName";

**public** **static** **final** String ***LAST\_FOUR\_DIGITS\_ID\_PROOF*** = "lastFourDigitsOfIdProof";

**public** **static** **final** String ***ID\_PROOF\_DOCUMENTS*** = "idProofDocuments";

**public** **static** **final** String ***PRIVACY\_REQUEST\_DOCUMENT\_UPLOAD\_ID\_TEXT*** = "privacyRequestDocumentUploadId";

**public** **static** **final** String ***COLUMN\_NAME\_TXT*** = "columnName";

**public** **static** **final** String ***SORT\_ORDER\_TXT*** = "sortOrder";

**public** **static** **final** String ***AKER\_PROXY\_REQUEST\_TIMEOUT*** = "request-timeout";

**public** **static** **final** String ***REQUEST\_TYPE*** = "requestType";

**public** **static** **final** String ***FIRST\_NAME\_COLUMN*** = "first\_nam";

**public** **static** **final** String ***FIRSTNAME\_XSL\_KEY*** = "body.salutation.firstName";

**public** **static** **final** String ***REQUESTID\_XSL\_KEY*** = "body.requestId";

**public** **static** **final** String ***REQUESTURL\_XSL\_KEY*** = "body.requestURL";

**public** **static** **final** String ***MOCK\_EMAILID\_XSL\_KEY*** = "mockEmailId";

**public** **static** **final** String ***SEARCH\_REQUEST\_ID*** = "<REQUESTID>";

**public** **static** **final** String ***DELETE\_REQUEST\_ID*** = "DeleteReqId";

**public** **static** **final** String ***VIEW\_REQUEST\_ID*** = "ViewReqId";

**public** **static** **final** String ***EVENT\_TYPE\_RQST\_SBMT*** = "RQST\_SBMT";

**public** **static** **final** String ***EVENT\_TYPE\_RQST\_DELAY*** = "RQST\_DELAY";

**public** **static** **final** String ***EVENT\_TYPE\_EMAIL\_VER*** = "EMAIL\_VER";

**public** **static** **final** String ***EVENT\_TYPE\_EMAIL\_RMNDR*** = "EMAIL\_RMNDR";

**public** **static** **final** String ***DATEOFREQUEST\_XSL\_KEY*** = "body.dateOfRequest";

**public** **static** **final** String ***POI\_PURGE\_AUD\_DESC*** = "POI Documents purge Action completed for Privacyrequest: %s.";

**public** **static** **final** String ***REQUEST\_PURGE\_AUD\_DESC*** = "Privacy request purging";

**public** **static** **final** String ***PHONE\_NUMBERS*** = "phoneNumbers";

**public** **static** **final** **int** ***NO\_CONTENT\_STATUS\_CODE*** = 204;

**public** **static** **final** **int** ***TIMED\_OUT\_STATUS\_CODE*** = 408;

**public** **static** **final** String ***PRIVACY\_REQUEST\_APPROVALS\_BATCH*** = "PrivacyRequestApprovalsBatch";

**public** **static** **final** **int** ***SUCCESS\_STATUS\_CODE*** = 200;

**public** **static** **final** String ***ISMANUAL*** = "isManual";

**public** **static** **final** String ***VERIFIED*** = "V";

**public** **static** **final** String ***CURING\_ATTEMPTS*** = "curingAttemptsCount";

**public** **static** **final** String ***IN\_CURING\_DAYS*** = "daysInCuring";

**public** **static** **final** String ***UPLOADFILEPATH*** = "./src/main/resources/config/TestData/";

**public** **static** **final** String ***SEARCH\_KEYS\_CARD\_DETAILS\_FIRST\_CARD\_NUMBER*** = "searchKeys.cardDetails[0].cardNumber";

**public** **static** **final** String ***CARD\_DETAILS\_FIRST\_CARD\_NUMBER*** = "cardDetails[0].cardNumber";

**public** **static** **final** String ***EXPORT\_CARD\_NUMBERS\_LIST*** = "products[0].serviceFunctions[0].records[2].recordData.cardDetails.findAll{it}.cardNumber";

**public** **static** **final** String ***REQUEST\_ID\_REPLACER*** = "<REQUEST\_ID>";

**public** **static** **final** String ***AUD\_CODE\_REPLACER*** = "<AUD\_CODE>";

**public** **static** **final** String ***POA*** = "POA";

**public** **static** **final** String ***RWS*** = "RWS";

**public** **static** **final** String ***PAS\_WD*** = "password";

**public** **static** **final** String ***PRODUCT\_VERIFICATION\_STATUS\_CODE*** = "prdct\_ver\_stat\_cd";

**public** **static** **final** String ***AGRMT\_TYPE*** = "agrmt\_type";

**public** **static** **final** String ***ADDR\_TYPE*** = "addr\_type";

**public** **static** **final** String ***DOC\_TYPE*** = "doc\_type";

**public** **static** **final** String ***EXPRT\_ACTN*** = "EXPRT\_ACTN";

**public** **static** **final** String ***FILE\_REPO\_ID*** = "fileRepoId";

**public** **static** **final** String ***USER\_ID*** = "userId";

**public** **static** **final** String ***DATE*** = "date";

**public** **static** **final** String ***BULK\_UPLOAD\_FILE\_PATH*** = "./src/main/resources/config/TestData/%s";

**public** **static** **final** String ***TILDE\_SEPERATOR*** = "~";

**public** **static** **final** String ***BULK\_REQUEST\_UPLOAD\_ID*** = "bulkRequestUploadId";

**public** **static** **final** String ***BULK\_REQUEST\_DEL\_DRAFT*** = "BULK\_RQST\_DEL\_DRAFT";

**private** Constants() {

}

}

**package** com.mastercard.testing.gdp.api.helper.constants;

**public** **final** **class** ContextConstants {

**public** **static** **final** String ***RESPONSE*** = "RESPONSE";

**public** **static** **final** String ***REQUEST\_SPECIFICATION*** = "REQUEST\_SPECIFICATION";

**public** **static** **final** String ***REQUEST\_PAYLOAD*** = "REQUEST\_PAYLOAD";

**public** **static** **final** String ***RESPONSE\_OBJECT*** = "RESPONSE\_OBJECT";

**private** ContextConstants() {

}

}

**package** com.mastercard.testing.gdp.api.model;

**public** **enum** Result {

***PASSED***,

***FAILED***,

***NOT\_EXECUTED***

}

**package** com.mastercard.testing.gdp.api.model;

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** Scenario {

**private** String title;

**private** List<Step> steps;

**private** Result result;

**private** Throwable exception;

**public** Scenario(String title, Result result) {

**this**.title = title;

**this**.result = result;

**this**.steps = **new** LinkedList<>();

}

**public** String getTitle() {

**return** title;

}

**public** List<Step> getSteps() {

**return** steps;

}

**public** Result getResult() {

**return** result;

}

**public** **void** setResult(Result result) {

**this**.result = result;

}

**public** Throwable getException() {

**return** exception;

}

**public** **void** setException(Throwable exception) {

**this**.exception = exception;

}

}

**package** com.mastercard.testing.gdp.api.model;

**public** **class** Step {

**private** String description;

**private** Result result;

**public** Step(String description, Result result) {

**this**.description = description;

**this**.result = result;

}

**public** String getDescription() {

**return** description;

}

**public** Result getResult() {

**return** result;

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** org.jbehave.core.model.ExamplesTable;

**import** org.jbehave.core.model.GivenStories;

**import** org.jbehave.core.model.Lifecycle;

**import** org.jbehave.core.model.Meta;

**import** org.jbehave.core.model.Narrative;

**import** org.jbehave.core.model.OutcomesTable;

**import** org.jbehave.core.model.Scenario;

**import** org.jbehave.core.model.Story;

**import** org.jbehave.core.model.StoryDuration;

**import** org.jbehave.core.reporters.StoryReporter;

**import** org.jbehave.core.steps.StepCollector;

**import** java.util.List;

**import** java.util.Map;

**abstract** **class** BaseReporter **implements** StoryReporter {

@Override

**public** **void** beforeStory(Story story, **boolean** givenStory) {}

@Override

**public** **void** storyNotAllowed(Story story, String filter) {}

@Override

**public** **void** storyCancelled(Story story, StoryDuration storyDuration) {}

@Override

**public** **void** afterStory(**boolean** givenOrRestartingStory) {}

@Override

**public** **void** narrative(Narrative narrative) {}

@Override

**public** **void** lifecyle(Lifecycle lifecycle) {}

@Override

**public** **void** scenarioNotAllowed(Scenario scenario, String filter) {}

@Override

**public** **void** beforeScenario(Scenario var1) {}

@Override

**public** **void** beforeScenario(String scenarioTitle) {}

@Override

**public** **void** scenarioMeta(Meta meta) {}

@Override

**public** **void** afterScenario() {}

@Override

**public** **void** beforeGivenStories() {}

@Override

**public** **void** givenStories(GivenStories givenStories) {}

@Override

**public** **void** givenStories(List<String> storyPaths) {}

@Override

**public** **void** afterGivenStories() {}

@Override

**public** **void** beforeExamples(List<String> steps, ExamplesTable table) {}

@Override

**public** **void** example(Map<String, String> tableRow) {}

@Override

**public** **void** afterExamples() {}

@Override

**public** **void** beforeStep(String step) {}

@Override

**public** **void** successful(String step) {}

@Override

**public** **void** ignorable(String step) {}

@Override

**public** **void** comment(String step) {}

@Override

**public** **void** pending(String step) {}

@Override

**public** **void** notPerformed(String step) {}

@Override

**public** **void** failed(String step, Throwable cause) {}

@Override

**public** **void** failedOutcomes(String step, OutcomesTable table) {}

@Override

**public** **void** restarted(String step, Throwable cause) {}

@Override

**public** **void** restartedStory(Story story, Throwable cause) {}

@Override

**public** **void** dryRun() {}

@Override

**public** **void** pendingMethods(List<String> methods) {}

@Override

**public** **void** beforeStorySteps(StepCollector.Stage stage) {}

@Override

**public** **void** afterStorySteps(StepCollector.Stage stage) {}

@Override

**public** **void** beforeScenarioSteps(StepCollector.Stage stage) {}

@Override

**public** **void** afterScenarioSteps(StepCollector.Stage stage) {}

@Override

**public** **void** example(Map<String, String> map, **int** i) {}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** org.jbehave.core.reporters.Format;

**import** java.util.Comparator;

/\*\*

\* Created by Yahor\_Alve on 6/17/2015.

\*/

**public** **class** ComparatorFormatByName **implements** Comparator<Format> {

@Override

**public** **int** compare(Format o1, Format o2) {

String name1 = o1.toString();

String name2 = o2.toString();

**return** name1.compareToIgnoreCase(name2);

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** com.mastercard.testing.gdp.api.model.Result;

**import** org.apache.log4j.Logger;

**import** org.springframework.stereotype.Component;

**import** com.mastercard.testing.gdp.api.model.\*;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** java.nio.file.Paths;

**import** java.nio.file.StandardOpenOption;

@Component

**public** **class** FailedStoriesReporter **extends** BaseReporter {

**public** **static** **final** String ***REPORT\_FILE\_PATH*** = "./target/FailedStories.txt";

**public** **static** **final** String ***PATHS\_BREAKER*** = ",";

**private** **static** **final** Logger ***LOGGER*** = Logger.*getLogger*(FailedStoriesReporter.**class**);

**private** ThreadLocal<Story> currentStory = **new** ThreadLocal<>();

**public** FailedStoriesReporter() {

File file = **new** File(***REPORT\_FILE\_PATH***);

**try** {

Files.*deleteIfExists*(file.toPath());

} **catch** (IOException ignored) {

***LOGGER***.debug(String.*format*("Could not delete file '%s'", ***REPORT\_FILE\_PATH***));

}

}

@Override

**public** **void** beforeStory(org.jbehave.core.model.Story story, **boolean** givenStory) {

**if** (!givenStory) {

**this**.currentStory.set(**new** Story(story));

}

}

@Override

**public** **void** failed(String step, Throwable cause) {

currentStory.get().setResult(Result.***FAILED***);

}

@Override

**public** **void** afterStory(**boolean** givenOrRestartingStory) {

**if** (currentStory.get().getResult() == Result.***FAILED*** && !givenOrRestartingStory) {

saveFailedStoryName(currentStory.get().getStory().getName());

currentStory.remove();

}

}

**private** **void** saveFailedStoryName(String name) {

**if** (Paths.*get*(***REPORT\_FILE\_PATH***).toFile().exists()) {

**try** {

Files.*write*(Paths.*get*(***REPORT\_FILE\_PATH***), ("," + name).getBytes(), StandardOpenOption.***APPEND***, StandardOpenOption.***SYNC***);

} **catch** (IOException e) {

***LOGGER***.error(e.getCause());

}

}

**else** {

**try** {

Files.*write*(Paths.*get*(***REPORT\_FILE\_PATH***), name.getBytes());

} **catch** (IOException e) {

***LOGGER***.error(e.getCause());

}

}

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.RallyStoryReporter;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.alm.ALMService;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.alm.rally.RallyReportScenario;

**public** **class** GdpRallyStoryReporter **extends** RallyStoryReporter {

**public** GdpRallyStoryReporter(ALMService almService) {

**super**(almService);

**this**.initialReporterSetup();

}

@Override

**public** **synchronized** **void** updateTestCase(RallyReportScenario scenario) {

**super**.updateTestCase(scenario);

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** java.lang.reflect.InvocationTargetException;

**import** java.lang.reflect.UndeclaredThrowableException;

**public** **class** ReporterUtils {

**private** **static** **final** Integer ***DEFAULT\_CAUSE\_DEEP*** = 1;

**private** **static** **final** Integer ***COUNT\_ELEMENT\_IN\_STACK\_TRACE*** = 40;

**private** ReporterUtils() {

}

**public** **static** String formatErrorMessage(String messageText, Throwable throwable) {

StringBuilder message = **new** StringBuilder().append(messageText);

**if** (throwable != **null**) {

message.append(throwable.getMessage());

message.append("\n");

message.append("ERROR : " + throwable);

message.append("\n");

message.append("First " + ***DEFAULT\_CAUSE\_DEEP*** + " causes of ERROR with their short StackTrace : \n");

*appendCause*(message, throwable);

message.append("\n\n");

message.append("All causes of error in descending order without short StackTrace : \n");

*appendAllCauses*(message, throwable);

message.append("\n\n");

message.append("Main StackTrace: ");

*appendMainStackTrace*(message, throwable.getStackTrace());

} **else** {

message.append(throwable);

}

**return** message.toString();

}

**private** **static** Throwable getPreviousThrowable(Throwable th) {

**if** (th **instanceof** InvocationTargetException) {

**return** ((InvocationTargetException) th).getTargetException();

}

**if** (th **instanceof** UndeclaredThrowableException) {

**return** ((UndeclaredThrowableException) th).getUndeclaredThrowable();

}

**return** th.getCause();

}

**private** **static** **void** appendStackTrace(StringBuilder message, StackTraceElement[] stackTrace) {

**if** (stackTrace != **null**) {

**int** count = stackTrace.length < ***COUNT\_ELEMENT\_IN\_STACK\_TRACE*** ? stackTrace.length

: ***COUNT\_ELEMENT\_IN\_STACK\_TRACE***;

**for** (**int** i = 0; i < count; i++) {

message.append("\n\t");

message.append(stackTrace[i].toString());

}

} **else** {

message.append("<stack trace is null>");

}

}

**private** **static** **void** appendMainStackTrace(StringBuilder message, StackTraceElement[] stackTrace) {

**if** (stackTrace != **null**) {

**for** (StackTraceElement stackTraceElement : stackTrace) {

message.append("\n\t");

message.append(stackTraceElement.toString());

}

} **else** {

message.append("<stack trace is null>");

}

}

**private** **static** **void** appendAllCauses(StringBuilder message, Throwable root) {

Throwable prevCause = root;

message.append("\n" + "ERROR : " + prevCause + " happened due to ");

**while** (*getPreviousThrowable*(prevCause) != **null**) {

prevCause = *getPreviousThrowable*(prevCause);

message.append("\n Cause by: ");

message.append("\n" + prevCause);

message.append("\n...");

}

}

**private** **static** **void** appendCause(StringBuilder message, Throwable root) {

Throwable prevCause = root;

**for** (**int** causeCounter = 0; causeCounter < ***DEFAULT\_CAUSE\_DEEP***; causeCounter++) {

prevCause = *getPreviousThrowable*(prevCause);

**if** (prevCause == **null**) {

**break**;

}

message.append("\n Cause by: ");

message.append(prevCause.getMessage());

message.append("\n" + prevCause);

message.append("\n");

message.append("Short StackTrace : ");

*appendStackTrace*(message, prevCause.getStackTrace());

message.append("\n...");

}

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** java.io.File;

**import** java.util.List;

**import** java.util.Properties;

**import** java.util.concurrent.atomic.AtomicBoolean;

**import** org.jbehave.core.model.StoryMaps;

**import** org.jbehave.core.reporters.FreemarkerViewGenerator;

**import** com.epam.reportportal.jbehave.JBehaveUtils;

**public** **class** RPFreemarkerViewGenerator **extends** FreemarkerViewGenerator {

**private** **final** AtomicBoolean finished;

**public** RPFreemarkerViewGenerator() {

finished = **new** AtomicBoolean(**false**);

}

@Override

**public** **void** generateMapsView(File outputDirectory, StoryMaps storyMaps, Properties viewResources) {

**super**.generateMapsView(outputDirectory, storyMaps, viewResources);

**if** (finished.compareAndSet(**false**, **true**)) {

JBehaveUtils.*finishLaunch*();

}

}

@Override

**public** **void** generateReportsView(File outputDirectory, List<String> formats, Properties viewResources) {

**super**.generateReportsView(outputDirectory, formats, viewResources);

**if** (finished.compareAndSet(**false**, **true**)) {

JBehaveUtils.*finishLaunch*();

}

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** com.mastercard.testing.gdp.api.aspect.StepArgumentProvider;

**import** org.jbehave.core.model.OutcomesTable;

**import** org.jbehave.core.reporters.ConcurrentStoryReporter;

**import** org.jbehave.core.reporters.StoryReporter;

**public** **class** StepArgumentReporter **extends** ConcurrentStoryReporter {

**private** StepArgumentProvider stepArgumentProvider;

**public** StepArgumentReporter(StoryReporter crossReferencing, StoryReporter delegate, **boolean** multiThreading,

StepArgumentProvider stepArgumentProvider) {

**super**(crossReferencing, delegate, multiThreading);

**this**.stepArgumentProvider = stepArgumentProvider;

}

**private** String getStepInfo(String step) {

Object[] arguments = stepArgumentProvider.getLastStepArguments();

**if** (arguments == **null** || arguments.length == 0) {

**return** step;

}

StringBuilder builder = **new** StringBuilder(step).append(" values = [");

**for** (**int** i = 0; i < arguments.length; i++) {

builder.append(arguments[i]).append(i < arguments.length - 1 ? ", " : "]");

}

**return** builder.toString();

}

@Override

**public** **void** successful(String step) {

**super**.successful(getStepInfo(step));

}

@Override

**public** **void** ignorable(String step) {

**super**.ignorable(getStepInfo(step));

}

@Override

**public** **void** pending(String step) {

**super**.pending(getStepInfo(step));

}

@Override

**public** **void** notPerformed(String step) {

**super**.notPerformed(getStepInfo(step));

}

@Override

**public** **void** failed(String step, Throwable cause) {

**super**.failed(getStepInfo(step), cause);

}

@Override

**public** **void** failedOutcomes(String step, OutcomesTable table) {

**super**.failedOutcomes(getStepInfo(step), table);

}

@Override

**public** **void** restarted(String step, Throwable cause) {

**super**.restarted(getStepInfo(step), cause);

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** com.mastercard.testing.gdp.api.aspect.StepArgumentProvider;

**import** org.jbehave.core.reporters.DelegatingStoryReporter;

**import** org.jbehave.core.reporters.NullStoryReporter;

**import** org.jbehave.core.reporters.StoryReporter;

**import** org.jbehave.core.reporters.StoryReporterBuilder;

**import** java.util.Map;

**import** java.util.TreeMap;

**public** **class** StepArgumentReporterBuilder **extends** StoryReporterBuilder {

**private** StepArgumentProvider stepArgumentProvider;

**public** StepArgumentReporterBuilder(StepArgumentProvider stepArgumentProvider) {

**this**.stepArgumentProvider = stepArgumentProvider;

}

@Override

**public** StoryReporter build(String storyPath) {

Map<org.jbehave.core.reporters.Format, StoryReporter> delegates = **new** TreeMap<>(**new** ComparatorFormatByName());

**for** (org.jbehave.core.reporters.Format format : formats()) {

delegates.put(format, reporterFor(storyPath, format));

}

DelegatingStoryReporter delegate = **new** DelegatingStoryReporter(delegates.values());

**return** **new** StepArgumentReporter(**new** NullStoryReporter(), delegate, multiThreading(), stepArgumentProvider);

}

}

**package** com.mastercard.testing.gdp.api.steps;

**import** com.mastercard.testing.gdp.api.aspect.JsonTestData;

**import** com.mastercard.testing.gdp.api.domain.AuditDataTextPurge;

**import** com.mastercard.testing.gdp.api.domain.ErrorMessage;

**import** com.mastercard.testing.gdp.api.domain.ErrorResponse;

**import** com.mastercard.testing.gdp.api.domain.entities.AuditHistoryEntity;

**import** com.mastercard.testing.gdp.api.domain.entities.PrivacyRequestEntity;

**import** com.mastercard.testing.gdp.api.domain.entities.handlers.AuditHistoryEntityHandler;

**import** com.mastercard.testing.gdp.api.domain.entities.handlers.PrivacyRequestHandler;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

**import** com.mastercard.testing.gdp.api.helper.ResponseConverter;

**import** com.mastercard.testing.gdp.api.helper.constants.Constants;

**import** com.mastercard.testing.gdp.api.helper.constants.LinkConstants;

**import** com.mastercard.testing.gdp.api.helper.constants.QueryConstants;

**import** com.mastercard.testing.gdp.restassured.domain.ConsumerRequest;

**import** com.mastercard.testing.gdp.restassured.domain.User;

**import** com.mastercard.testing.gdp.restassured.mock.TestConsumerAPI;

**import** com.mastercard.testing.gdp.restassured.utils.StepUtils;

**import** io.restassured.response.Response;

**import** io.restassured.specification.RequestSpecification;

**import** org.apache.commons.lang3.StringUtils;

**import** org.apache.commons.lang3.math.NumberUtils;

**import** org.apache.http.HttpStatus;

**import** org.assertj.core.api.Assertions;

**import** org.jbehave.core.annotations.\*;

**import** org.jbehave.core.model.ExamplesTable;

**import** org.json.JSONObject;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.stereotype.Component;

**import** java.io.IOException;

**import** java.util.Arrays;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.Constants.\*;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.ContextConstants.***REQUEST\_PAYLOAD***;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.ContextConstants.***REQUEST\_SPECIFICATION***;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.ContextConstants.***RESPONSE***;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.QueryConstants.***REMOVE\_ERRONEOUS\_ENTRIES\_FROM\_INTEGRATION\_TABLE***;

**import** **static** org.assertj.core.api.Assertions.*assertThat*;

@Component

**public** **class** CommonSteps **extends** GDPTestSteps {

**private** **static** **final** String ***USER\_NAME*** = "userName";

**private** String resourcePath = **null**;

@Autowired

**private** JsonTestData jsonTestData;

@Value("${env:local}")

**private** String environment;

@Value("${app.auth.password}")

**private** String b2cPwd;

@Autowired

**private** TestConsumerAPI testConsumerAPI;

@Given("comment $comment")

**public** **void** comment(@Named("$comment") **final** String comment) {

*LOG*.info("comment :" + comment);

}

@Given("I set the json file $path for this story into context")

@Alias("the json file $path is set for this story into context")

**public** **void** setJsonPathToContext(@Named("$path") String path) {

String jsonTestdataPath = **null**;

**if** (path != **null** && !path.isEmpty()) {

**if** (**new** ClassPathResource("/config/" + environment + path).exists()) {

jsonTestdataPath = "/config/" + environment + path;

} **else** **if** (**new** ClassPathResource("/config/testdatafiles" + path).exists()) {

jsonTestdataPath = "/config/testdatafiles" + path;

}

*LOG*.info("Json test data file: " + jsonTestdataPath);

jsonTestData.init(jsonTestdataPath);

}

}

@Given("expected status code must be $statusCode")

@When("expected status code must be $statusCode")

@Then("expected status code must be $statusCode")

**public** **void** validateStatusCode(@Named("$statusCode") String expectedStatusCode) {

commonFunctions.validateStatusCode(Integer.*parseInt*(expectedStatusCode));

}

@Then("expected status code from authenticated rest call must be $statusCode")

**public** **void** validateStatusCodeAuthenticatedRestCall(@Named("$statusCode") String expectedStatusCode) {

commonFunctions.validateStatusCodeAuthenticatedCall(Integer.*parseInt*(expectedStatusCode));

}

@When("execute put call with request payload $requestBody")

**public** **void** executePutAPI(@Named("$requestBody") String requestBody) {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

contextVariableTestData.setContextVariable(***REQUEST\_PAYLOAD***, requestBody);

commonFunctions.executePutRequestWithRequestBodyAndWithRequestSpecification(requestSpec,requestBody);

}

@Then("attribute $path has the value $attributeValue in the response")

**public** **void** compareValueFromResponse(@Named("$path") String path, @Named("$attributeValue") String attributeValue) {

Assertions.*assertThat*(commonFunctions.getResponseFromContextProvider().path(path).toString()).as("the value in response does not match")

.isEqualTo(attributeValue);

}

@Then("verify that below attributes are sent in the response $attributes")

**public** **void** compareValueFromResponse(@Named("$attributes") ExamplesTable attributes) {

attributes.getRows().forEach(value -> Assertions

.*assertThat*(commonFunctions.getResponseFromContextProvider().path(value.get("attribute")).toString())

.as("the value in response does not match").isNotNull());

}

@Given("GDP data access url is set as base url")

**public** **void** setGdpDataAccessUrl() {

commonFunctions.setBaseURL(LinkConstants.***GDP\_DATA\_ACCESS\_BASE\_URL***);

}

@Given("GDP app api is set as base url")

**public** **void** setGdpAppApiUrl() {

commonFunctions.setBaseURL(LinkConstants.***GDP\_AOB\_APP\_BASE\_URL***);

}

@Then("validate that expected status code should be $statusCode")

**public** **void** verifyStatusCodeFromHttpResponse(@Named("statusCode") **int** statusCode) **throws** IOException {

Assertions.*assertThat*(commonFunctions.getStatusCodeFromHttpResponse())

.as(Constants.***FAILED\_API\_RESPONSE\_IS*** + commonFunctions.getHttpResponse()).isEqualTo(statusCode);

}

@Then("Error Message should be returned back with $errorCode, $errorMessage, $statusCode, $field")

**public** **void** verifyError(@Named("errorMessage") String errorMessage, @Named("errorCode") String errorCode,

@Named("statusCode") String statusCode, @Named("field") String field) {

List<ErrorMessage> errMessage;

**if** (testContextProvider.get().get(***RESPONSE***) **instanceof** ResponseConverter) {

errMessage = Arrays.*asList*(commonFunctions.getActualPojoClassesFromContext(ErrorMessage.**class**));

} **else** {

errMessage = Arrays.*asList*(commonFunctions.getActualPojoClasses(ErrorMessage.**class**));

}

*assertThat*(errMessage.get(NumberUtils.***INTEGER\_ZERO***).getMessage()).isEqualTo(errorMessage);

*assertThat*(errMessage.get(NumberUtils.***INTEGER\_ZERO***).getCode()).isEqualTo(errorCode);

*assertThat*(errMessage.get(NumberUtils.***INTEGER\_ZERO***).getStatus()).isEqualTo(statusCode);

**if** (field.isEmpty()) {

*assertThat*(errMessage.get(NumberUtils.***INTEGER\_ZERO***).getField()).isNull();

} **else** {

*assertThat*(errMessage.get(NumberUtils.***INTEGER\_ZERO***).getField()).isEqualTo(field);

}

}

@Then("validate the JSON error response $jsonErrorResponse for $message")

@Alias("validate JSON error response $jsonErrorResponse for the $message")

**public** **void** validateErrorMessageInCommonSteps(@Named("jsonErrorResponse") String jsonErrorResponse,

@Named("message") String message) {

*LOG*.info("Validating the actual json response for " + message);

List<ErrorMessage> actualErrorMessage = Arrays.*asList*(commonFunctions.getActualPojoClasses(ErrorMessage.**class**));

List<ErrorMessage> expectedErrorMessage = Arrays

.*asList*(commonFunctions.parseResponse(jsonErrorResponse, ErrorMessage.**class**));

Assertions.*assertThat*(actualErrorMessage).as("error response is incorrect").containsAll(expectedErrorMessage);

}

@Then("validate the API response $jsonResponse for invalid scenarios")

**public** **void** validateErrorResponse(@Named("jsonResponse") String expectedJson) {

ErrorResponse actualErrorMessage = commonFunctions.getActualPojoClasses(ErrorResponse.**class**);

ErrorResponse expectedErrorMessage = commonFunctions.parseResponse(expectedJson, ErrorResponse.**class**);

Assertions.*assertThat*(actualErrorMessage).as("error response is incorrect").isEqualTo(expectedErrorMessage);

}

@Given("Test start time is captured")

**public** **void** setStartTime() {

commonFunctions.setStartTime();

}

@Given("Test end time is captured")

@When("Test end time is captured")

**public** **void** setEndTime() {

commonFunctions.setEndTime();

}

@Given("the mock scheduler API is available for job $scheduledJob")

**public** **void** givenAPIExists(@Named("scheduledJob") String scheduledJob) {

StringBuilder builder = **new** StringBuilder();

resourcePath = builder.append(LinkConstants.***GDP\_AOB\_APP\_API***)

.append(String.*format*(LinkConstants.***BATCH\_JOB***, scheduledJob)).toString();

commonFunctions.setBaseURL(LinkConstants.***GDP\_AOB\_APP\_BASE\_URL***);

*LOG*.info("Mock scheduler API is invoked with end-point :" + resourcePath);

}

@Given("the API is available for job $scheduledJob")

**public** **void** givenBatchAPIExists(@Named("scheduledJob") String scheduledJob) {

StringBuilder builder = **new** StringBuilder();

resourcePath = builder.append(LinkConstants.***GDP\_AOB\_APP\_API***)

.append(String.*format*(LinkConstants.***BATCH\_JOB\_FILE***, scheduledJob)).toString();

commonFunctions.setBaseURL(LinkConstants.***GDP\_AOB\_APP\_BASE\_URL***);

*LOG*.info("Batch API is invoked with end-point :" + resourcePath);

}

@When("Mock scheduler API is called")

**public** **void** invokeSchedulerAPI() {

cleanUpJunkDataBeforeAPICall();

commonFunctions.executeGetRequest(resourcePath, **null**);

**if** (isPrivacyRequestApprovalsBatchSuccess()) {

String request = contextVariablesSteps.getContextVariableData("requestID");

String statusQuery = String.*format*(QueryConstants.***GET\_PRVCY\_RQST\_STATUS***, request);

verifyIfAPICallIsSuccessful(request, statusQuery);

} **else** {

throwTimeOutException();

}

}

@Given("updated $HideOrEncrypt switch flag for field $fieldName and status as $status")

**public** **void** updateHideSwitchField(@Named("$HideOrEncrypt") String hideOrEncrypt,

@Named("$fieldName") String fieldName, @Named("$status") String status) {

String qryToExecute;

**if** (hideOrEncrypt.contains("hide")) {

qryToExecute = String.*format*(QueryConstants.***UPDATE\_FIELD\_HIDE\_SW***, Boolean.*parseBoolean*(status), fieldName);

} **else** {

qryToExecute = String.*format*(QueryConstants.***UPDATE\_FIELD\_ENCRYPT\_SW***, Boolean.*parseBoolean*(status),

fieldName);

}

dbOperationUtils.executeUpdateStatement(qryToExecute);

}

@Given("Update hide switch flag with status $status for all elements in data elmt table")

**public** **void** updateDataElmtTableWithHideSwitchFlag(@Named("status") String status) {

String qryToExecute = String.*format*(QueryConstants.***UPDATE\_FIELD\_HIDE\_SW\_FOR\_ALL\_FIELDS***,

Boolean.*parseBoolean*(status));

dbOperationUtils.executeUpdateStatement(qryToExecute);

}

@Given("Update Encrypted switch flag with status $status for all elements in data elmt table")

**public** **void** updateDataElmtTableWithEncryptedSwitchFlag(@Named("status") String status) {

String qryToExecute = String.*format*(QueryConstants.***UPDATE\_FIELD\_ENCRYPT\_SW\_FOR\_ALL\_FIELDS***,

Boolean.*parseBoolean*(status));

dbOperationUtils.executeUpdateStatement(qryToExecute);

}

@Given("Update config\_parm\_val\_txt with value $status for activity.data.file.upload.retry.max.limit element in config\_parm table")

**public** **void** updateConfigParamTable(@Named("status") String status) {

String qryToExecute = String.*format*(QueryConstants.***UPDATE\_FILE\_UPLD\_RETRY\_MAX***,

Boolean.*parseBoolean*(status));

dbOperationOracleUtils.executeUpdateStatement(qryToExecute);

}

@Given("Update config\_parm\_val\_txt with value $config\_val for $config\_key element in config\_parm table")

**public** **void** updateConfigParamTableValue(@Named("config\_val") String value,@Named("config\_key") String key) {

String qryToExecute = String.*format*(QueryConstants.***UPDATE\_CONFIG\_PARAM\_VALUE***,

value,key);

dbOperationOracleUtils.executeUpdateStatement(qryToExecute);

}

@Then("privacy request $requestId should be having status $requestStatus")

**public** **void** getRequestStatusFromDb(@Named("requestId") String requestId,

@Named("requestStatus") String requestStatus) {

PrivacyRequestEntity prvcyRqstRecord = (PrivacyRequestEntity) (dbOperationUtils.getCompleteRowData(

String.*format*(QueryConstants.***GET\_PRIVACY\_REQUEST\_FOR\_GIVEN\_REQUEST\_ID***, requestId),

PrivacyRequestHandler.**class**)).get(0);

String prvcyRqstStatus = dbOperationUtils

.getSingleDbValue(String.*format*(QueryConstants.***GET\_TYPE\_DESCRIPTION\_FROM\_LKP\_TABLE***,

"prvcy\_rqst\_stat\_lkp", prvcyRqstRecord.getRequestStatusCode()));

*assertThat*(prvcyRqstStatus).as("Request expected to be " + requestStatus + "but is " + prvcyRqstStatus)

.isEqualToIgnoringCase(requestStatus);

}

@Then("expected status code for batch jobs is $statusCode for requestId $requestId")

**public** **void** validateStatusCodeForDispatchApi(@Named("$statusCode") String expectedStatusCode,

@Named("requestId") String requestId) {

**if** (isJobTimedOut()) {

String query = String.*format*(QueryConstants.***GET\_CHILD\_RECORD\_COUNT***, requestId);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query))).as("Child Records are not created")

.isGreaterThan(NumberUtils.***INTEGER\_ZERO***);

} **else** {

*assertThat*(commonFunctions.getActualStatusCodeFromContextProvider())

.as(Constants.***FAILED\_API\_RESPONSE\_IS*** + commonFunctions.getResponseFromContextProvider().getBody())

.isEqualTo(Integer.*parseInt*(expectedStatusCode));

}

}

@Given("serviceFunctionLitigationHold table has $noOfRecords entry with serv\_func\_cd $serviceFunctionCode")

@Then("serviceFunctionLitigationHold table has $noOfRecords entry with serv\_func\_cd $serviceFunctionCode")

**public** **void** verifyServiceFunctionLitigationHold(@Named("noOfRecords") String noOfRecords,

@Named("serviceFunctionCode") String serviceFunctionCode) {

String query = String.*format*(QueryConstants.***GET\_COUNT\_OF\_SERVICE\_FUNCTION\_LITIGATION\_HOLDS***,

serviceFunctionCode);

*assertThat*(dbOperationUtils.getSingleDbValue(query))

.as("serviceFunctionLitigationHold table doesn't have " + noOfRecords + " records")

.isEqualTo(noOfRecords);

}

@Then("verify serviceFunctionLitigationHold table entry is created by user $userName for $serviceFunctionCode")

**public** **void** verifyServiceFunctionLitigationHoldCreatedUser(@Named(***USER\_NAME***) String userName,

@Named("serviceFunctionCode") String serviceFunctionCode) {

String query = String.*format*(QueryConstants.***GET\_CREATED\_USER\_FOR\_LATEST\_LH\_RECORD***, serviceFunctionCode);

*assertThat*(dbOperationUtils.getSingleDbValue(query))

.as("serviceFunctionLitigationHold table is not created by user : " + userName).isEqualTo(userName);

}

@Then("verify privacy request $requestID is purged from Data Access database")

**public** **void** verifyRequestNotExists(@Named("requestID") String requestID) {

String query = String.*format*(QueryConstants.***GET\_PRIVACY\_REQUEST\_RECORD\_COUNT***, requestID);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query)))

.as(requestID + " is NOT purged from Data Access DB").isEqualTo(NumberUtils.***INTEGER\_ZERO***);

}

@Then("verify privacy request $requestID is NOT purged from Data Access database")

**public** **void** verifyRequestExists(@Named("requestID") String requestID) {

String query = String.*format*(QueryConstants.***GET\_PRIVACY\_REQUEST\_RECORD\_COUNT***, requestID);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query)))

.as(requestID + " is purged from Data Access DB").isEqualTo(1);

}

@Then("verify record pertaining to service function $serviceFunctionInLH of privacy request $requestID is NOT purged")

**public** **void** verifyServiceFunctionForRequestExists(@Named("$serviceFunctionInLH") String serviceFunctionInLH,

@Named("requestID") String requestID) {

String query = String.*format*(QueryConstants.***GET\_COUNT\_OF\_SERVICE\_FUNCTIONS\_WITH\_PRIVACY\_REQUEST***, requestID,

serviceFunctionInLH);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query)))

.as(serviceFunctionInLH + " of " + requestID + " which is in LH is purged from Data Access DB")

.isEqualTo(1);

}

@Then("verify record pertaining to service function $serviceFunctionNotInLH of privacy request $requestID is purged")

**public** **void** verifyServiceFunctionForRequestNotExists(@Named("serviceFunctionNotInLH") String serviceFunctionNotInLH,

@Named("requestID") String requestID) {

String query = String.*format*(QueryConstants.***GET\_COUNT\_OF\_SERVICE\_FUNCTIONS\_WITH\_PRIVACY\_REQUEST***, requestID,

serviceFunctionNotInLH);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query))).as(

serviceFunctionNotInLH + " of " + requestID + " which is not in LH is NOT purged from Data Access DB")

.isEqualTo(NumberUtils.***INTEGER\_ZERO***);

}

@Then("Verify audit is created for purge with aud\_type\_cd as $auditTypeCode and aud\_data\_txt as $auditDataText")

**public** **void** verifyAuditForPurge(@Named("auditTypeCode") String auditTypeCode,

@Named("auditDataText") String auditDataText) {

String auditHistory = String.*format*(QueryConstants.***GET\_AUDIT\_RECORD\_BY\_AUD\_TYPE\_CD***, auditTypeCode);

List<AuditHistoryEntity> auditRecordsFromDB = dbOperationUtils.getCompleteRowData(auditHistory,

AuditHistoryEntityHandler.**class**);

*assertThat*(auditRecordsFromDB.size()).as("Audit is NOT made into aud\_hist table").isEqualTo(1);

*assertThat*(auditRecordsFromDB.get(0).getAuditHistoryId()).as("aud\_hist\_id is null in aud\_hist table")

.isNotNull();

*assertThat*(auditRecordsFromDB.get(0).getAuditDescription()).as("aud\_desc is incorrect in aud\_hist table")

.isEqualTo(Constants.***REQUEST\_PURGE\_AUD\_DESC***);

*assertThat*(auditRecordsFromDB.get(0).getAuditTimestamp()).as("aud\_ts is null in aud\_hist table").isNotNull();

*assertThat*(auditRecordsFromDB.get(0).getAuditUserId()).as("aud\_user\_id is incorrect in aud\_hist table")

.isEqualTo(Constants.***DATA\_ACCESS\_SYSTEM***);

List<AuditDataTextPurge> actualAuditDataText = Arrays.*asList*(

commonFunctions.parseResponse(auditRecordsFromDB.get(0).getAuditDataText(), AuditDataTextPurge.**class**));

List<AuditDataTextPurge> expectedAuditDataText = Arrays

.*asList*(commonFunctions.parseResponse(auditDataText, AuditDataTextPurge.**class**));

*assertThat*(actualAuditDataText).as(actualAuditDataText + " is NOT as expected")

.isEqualTo(expectedAuditDataText);

}

@Then("Verify privacy requests containing documents data with pattern $pattern is purged from Data Access database")

**public** **void** verifyDocumentsNotExists(@Named("pattern") String pattern) {

String chunkDocumentsCount = String.*format*(QueryConstants.***GET\_CHUNK\_DOCUMENTS\_COUNT***, pattern);

String docUploadCount = String.*format*(QueryConstants.***GET\_DOCUMENTS\_UPLOAD\_COUNT***, pattern);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(chunkDocumentsCount)))

.as(pattern + " Documents chunk is NOT purged from Data Access DB").isEqualTo(NumberUtils.***INTEGER\_ZERO***);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(docUploadCount)))

.as(pattern + " Documents upload is NOT purged from Data Access DB")

.isEqualTo(NumberUtils.***INTEGER\_ZERO***);

}

@Given("update $flag pci flag in ff4j features")

**public** **void** updatePCIFlag(@Named("flag") String flag) {

**if** (flag.contains("enable"))

dbOperationUtils.performUpdate(QueryConstants.***ENABLE\_PCI\_FLAG***);

**else**

dbOperationUtils.performUpdate(QueryConstants.***DISABLE\_PCI\_FLAG***);

}

@Given("update the FF4J flag $flag for attribute $attribute in ff4j features table")

**public** **void** updateFF4JData(@Named("flag") String flag, @Named("attribute") String attribute) {

String query = String.*format*(QueryConstants.***UPDATE\_PCI\_FLAG***, Integer.*valueOf*(flag), attribute);

dbOperationUtils.performUpdate(query);

}

@Then("verify Json with expectedJson $expectedJson")

**public** **void** verifyJson(@Named("expectedJson") String expectedJson) **throws** IOException {

**if** (StringUtils.*isEmpty*(expectedJson))

*assertThat*(commonFunctions.getResponseBody()).isEqualTo(StringUtils.***EMPTY***);

**else**

commonFunctions.compareJSONStrings(commonFunctions.getResponseBody(), expectedJson);

}

**private** **boolean** isPrivacyRequestApprovalsBatchSuccess() {

**return** Constants.***NO\_CONTENT\_STATUS\_CODE*** == commonFunctions.getActualStatusCodeFromContextProvider()

&& resourcePath.contains(Constants.***PRIVACY\_REQUEST\_APPROVALS\_BATCH***);

}

**private** **boolean** isJobTimedOut() {

**return** Constants.***TIMED\_OUT\_STATUS\_CODE*** == commonFunctions.getActualStatusCodeFromContextProvider();

}

@Then("verify mobile verification abandoned requests purged from the Data Access DB")

**public** **void** verifyMobileVerificationRequestsNotExists() {

*assertThat*(Integer.*parseInt*(

dbOperationUtils.getSingleDbValue(QueryConstants.***GET\_MOBILE\_VERIFICATION\_ABANDONED\_REQUESTS\_COUNT***)))

.as("mobile verification abandoned requests is NOT purged from Data Access DB")

.isEqualTo(NumberUtils.***INTEGER\_ZERO***);

}

@Then("verify phone number verification entry for privacy request $requestID is purged from Data Access database")

**public** **void** verifyPhVerNotExists(@Named("requestID") String requestID) {

String query = String.*format*(QueryConstants.***GET\_MOBILE\_VERIFICATION\_RECORDS\_COUNT***, requestID);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query)))

.as(requestID + " having mobile records is NOT purged from Data Access DB")

.isEqualTo(NumberUtils.***INTEGER\_ZERO***);

}

@Then("verify phone number verification entry for privacy request $requestID is NOT purged from Data Access database")

**public** **void** verifyPhVerExists(@Named("requestID") String requestID) {

String query = String.*format*(QueryConstants.***GET\_MOBILE\_VERIFICATION\_RECORDS\_COUNT***, requestID);

*assertThat*(Integer.*parseInt*(dbOperationUtils.getSingleDbValue(query)))

.as(requestID + " having mobile records is purged from Data Access DB")

.isGreaterThan(NumberUtils.***INTEGER\_ZERO***);

}

@Then("verify json response $expectedJsonResponse")

**public** **void** verifyJsonResponse(@Named("expectedJsonResponse") String expectedJsonResponse) {

*assertThat*(commonFunctions.getResponseFromContextProvider().asString())

.as("actual is not equal to expected json").isEqualTo(expectedJsonResponse);

}

@When("the generated $requestID is saved to the context")

**public** **void** saveRequestIDToContext(@Named("requestID") String requestID) {

contextVariablesSteps.saveContextVariable("requestID", requestID);

}

@Given("update the encryption flag for $dataElement to false if enabled")

**public** **void** updateEncryptionFlag(@Named("dataElement") String dataElement) {

**if** (isEncryptionEnabledForGivenDataElement(dataElement)) {

dbOperationUtils.executeUpdateStatement(

String.*format*(QueryConstants.***UPDATE\_ENCRYPTION\_SWITCH\_FOR\_GIVEN\_DATA\_ELEMENT***, dataElement));

}

}

@Given("the B2C user $userName is available in the system")

**public** **void** createUser(@Named(***USER\_NAME***) String userName) {

Map<String, Object> data = **new** HashMap<>();

data.put(***USER\_NAME***, userName);

data.put(Constants.***PAS\_WD***, b2cPwd);

String requestBody = **new** JSONObject(data).toString();

ConsumerRequest request = getRequestDetails(requestBody);

Response response = testConsumerAPI.executeCreateUserORDeleteRequests(request);

*assertThat*(testConsumerAPI.getStatusCode(response)).isEqualTo(String.*valueOf*(HttpStatus.***SC\_CREATED***));

}

@Given("the B2C user $userName is removed from the system")

**public** **void** removeUser(@Named(***USER\_NAME***) String userName) {

Map<String, Object> data = **new** HashMap<>();

data.put(***USER\_NAME***, userName);

String requestBody = **new** JSONObject(data).toString();

ConsumerRequest request = getRequestDetails(requestBody);

Response response = testConsumerAPI.executeDeleteUser(request);

*assertThat*(testConsumerAPI.getStatusCode(response)).isEqualTo(String.*valueOf*(HttpStatus.***SC\_OK***));

}

@Given("the privacy requests for B2C user $userName are deleted from the system")

**public** **void** deleteUserRequests(@Named(***USER\_NAME***) String userName) {

Map<String, Object> data = **new** HashMap<>();

data.put(***USER\_NAME***, userName);

data.put("email", userName + "@mastercard.com");

String requestBody = **new** JSONObject(data).toString();

ConsumerRequest request = getDeleteUserRequestsSpecRequest(requestBody);

Response response = testConsumerAPI.executeCreateUserORDeleteRequests(request);

*assertThat*(testConsumerAPI.getStatusCode(response)).isEqualTo(String.*valueOf*(HttpStatus.***SC\_OK***));

}

**private** ConsumerRequest getDeleteUserRequestsSpecRequest(String requestBody) {

**return** ConsumerRequest.*builder*().specification(testConsumerAPI.getDeleteUserRequestsSpec())

.userLogin(StepUtils.*getUserLogin*(User.*getGdpUserDetails*().getUserName())).requestBody(requestBody)

.build();

}

**private** ConsumerRequest getRequestDetails(String requestBody) {

**return** ConsumerRequest.*builder*().specification(testConsumerAPI.getCreateOrDeleteUserSpec())

.userLogin(StepUtils.*getUserLogin*(User.*getGdpUserDetails*().getUserName())).requestBody(requestBody)

.build();

}

**private** **boolean** isEncryptionEnabledForGivenDataElement(String dataElement) {

**return** NumberUtils.***INTEGER\_ONE***.toString().equals(dbOperationUtils

.getSingleDbValue(String.*format*(QueryConstants.***GET\_ENCRYPTED\_DATA\_ELEMENT\_COUNT***, dataElement)));

}

**private** **void** verifyIfAPICallIsSuccessful(String request, String statusQuery) {

**if** (StringUtils.*isNotEmpty*(request)) {

String requestStatus = dbOperationUtils.getSingleDbValue(statusQuery);

updateServiceFunctionDataIfKeySwitchFalse(requestStatus);

**boolean** flag = requestStatus.equals(***IN\_PROGRESS***) || requestStatus.equals(***COMPLETED***);

*assertThat*(flag)

.as("Unable to create the child records!\n" + " Check the logs from the data-access for the issue")

.isTrue();

}

}

**private** **void** throwTimeOutException() {

**try** {

**if** (isJobTimedOut())

**throw** **new** GDPAPIException("Timed out while trying to execute the batch job");

} **catch** (GDPAPIException e) {

**throw** **new** GDPAPIException("Timed out while trying to execute the batch job", e);

}

}

**private** **void** updateServiceFunctionDataIfKeySwitchFalse(String requestStatus) {

**if** (StringUtils.*equals*(***PRIVACY\_REQUEST\_STATUS\_READY\_TO\_PROCESS***, requestStatus)) {

dbOperationUtils.executeUpdateStatement(QueryConstants.***UPDATE\_KEY\_TRUE\_FOR\_MISSING\_SERVICE\_FUNCTIONS***);

}

}

**private** **void** cleanUpJunkDataBeforeAPICall() {

**if** (resourcePath.contains(Constants.***PRIVACY\_REQUEST\_APPROVALS\_BATCH***)) {

dbOperationUtils.executeDeleteStatement(QueryConstants.***CLEAN\_UP\_SERV\_FUNC\_INTEGRATION***);

dbOperationUtils.executeUpdateStatement(***REMOVE\_ERRONEOUS\_ENTRIES\_FROM\_INTEGRATION\_TABLE***);

}

}

@Then("verify audit is created for requestID as $requestID with aud\_type\_cd as $auditTypeCode")

**public** **void** verifyAuditForPurgeByRequestId(@Named("auditTypeCode") String auditTypeCode,

@Named("requestID") String requestId) {

String auditHistory = StringUtils.*replace*(QueryConstants.***GET\_AUDIT\_RECORD\_BY\_AUD\_TYPE\_CD\_AND\_REQUEST\_ID***

.replace(Constants.***AUD\_CODE\_REPLACER***, auditTypeCode), Constants.***REQUEST\_ID\_REPLACER***, requestId);

String count = dbOperationUtils.getSingleDbValue(auditHistory);

*assertThat*(count).as("Audit is NOT made into aud\_hist table").isEqualTo("1");

}

}

**package** com.mastercard.testing.gdp.api.steps;

**import** com.mastercard.testing.gdp.api.aspect.ContextVariableTestData;

**import** com.mastercard.testing.gdp.api.aspect.JsonTestData;

**import** org.assertj.core.api.Assertions;

**import** org.jbehave.core.annotations.Given;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** java.util.ArrayList;

**import** java.util.List;

@Component

**public** **class** ContextVariablesSteps {

@Autowired

**private** ContextVariableTestData variableTestData;

@Autowired

**private** JsonTestData jsonTestData;

/\*\*

\* Save string value in context

\*

\* **@param** name variable name

\* **@param** value variable value

\*/

@Given("variable '$name' equals '$value'")

**public** **void** saveContextVariable(String name, String value) {

variableTestData.setContextVariable(name, value);

}

/\*\*

\* Remove variable from context

\*

\* **@param** name variable name

\*/

@Given("variable '$name' is removed")

**public** **void** removeContextVariable(String name) {

variableTestData.setContextVariable(name, **null**);

}

/\*\*

\* Push value into array

\*

\* **@param** name list name

\* **@param** value value

\*/

@Given("value '$value' is pushed into list '$name'")

**public** **void** pushValueIntoList(String value, String name) {

Object list = variableTestData.getContextVariable(name);

**if** (list == **null**) {

list = **new** ArrayList<String>();

variableTestData.setContextVariable(name, list);

}

Assertions.*assertThat*(list **instanceof** List)

.overridingErrorMessage(String.*format*("Variable %s (type = %s) is not a List", name, list.getClass()))

.isTrue();

((List<String>) list).add(value);

}

/\*\*

\* Save list of string values in context

\*

\* **@param** name variable name

\* **@param** value variable values

\*/

@Given("list '$name' equals '$value'")

**public** **void** saveContextVariable(String name, List<String> value) {

variableTestData.setContextVariable(name, value);

}

/\*\*

\* Save exists context variable with new name

\*

\* **@param** name exists variable name

\* **@param** alias alias name

\*/

@Given("alias of '$name' is '$alias'")

**public** **void** saveContextVariableAs(String name, String alias) {

variableTestData.setContextVariable(alias, variableTestData.getContextVariable(name));

}

/\*\*

\* Save json object in context variables. Json object is got by jsonPath

\*

\* **@param** jsonPath path in json to object for save, array indexing is not

\* allowed

\* **@param** variableName new context variable name

\*/

@Given("json data '$jsonPath' saved as '$variableName'")

**public** **void** saveJsonDataAsVariable(String jsonPath, String variableName) {

variableTestData.setContextVariable(variableName, jsonTestData.getMap(jsonPath));

}

**public** String getContextVariableData(String key) {

**return** variableTestData.getValue(key);

}

}

**package** com.mastercard.testing.gdp.api.steps;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.ContextConstants.***REQUEST\_PAYLOAD***;

**import** **static** com.mastercard.testing.gdp.api.helper.constants.ContextConstants.***REQUEST\_SPECIFICATION***;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.charset.StandardCharsets;

**import** java.util.\*;

**import** org.assertj.core.api.Assertions;

**import** org.codehaus.plexus.util.StringUtils;

**import** org.jbehave.core.annotations.Alias;

**import** org.jbehave.core.annotations.Named;

**import** org.jbehave.core.annotations.Then;

**import** org.jbehave.core.annotations.When;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.stereotype.Component;

**import** com.mastercard.quality.engineering.mtaf.jbehave.context.TestContextProvider;

**import** com.mastercard.quality.engineering.mtaf.jbehave.steps.AbstractSteps;

**import** com.mastercard.testing.gdp.api.aspect.ContextVariableTestData;

**import** com.mastercard.testing.gdp.api.dbconnection.DBOperationOracleUtils;

**import** com.mastercard.testing.gdp.api.dbconnection.DBOperationUtils;

**import** com.mastercard.testing.gdp.api.domain.ErrorResponse;

**import** com.mastercard.testing.gdp.api.helper.CommonFunctions;

**import** com.mastercard.testing.gdp.api.helper.constants.Constants;

**import** com.mastercard.testing.gdp.api.steps.queries.CommonDBSteps;

**import** io.restassured.specification.RequestSpecification;

@Component

**public** **abstract** **class** GDPTestSteps **extends** AbstractSteps {

@Autowired

@Qualifier("commonFunctions")

**protected** CommonFunctions commonFunctions;

@Autowired

**protected** DBOperationUtils dbOperationUtils;

@Autowired

**protected** TestContextProvider testContextProvider;

@Autowired

**protected** ContextVariablesSteps contextVariablesSteps;

@Autowired

**protected** ContextVariableTestData contextVariableTestData;

**protected** RequestSpecification requestSpec;

**protected** String requestPayload;

**protected** String randomUUID;

@Autowired

**protected** DBOperationOracleUtils dbOperationOracleUtils;

@Autowired

**protected** CommonDBSteps commonDBSteps;

**public** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(GDPTestSteps.**class**);

**protected** **static** Map<String, String> prepareBasicMap() {

Map<String, String> dbToUIColumnMap = **new** HashMap<>();

dbToUIColumnMap.put("privacyRequestId", "prvcy\_rqst\_id");

dbToUIColumnMap.put("requestType", "rqst\_type\_cd");

dbToUIColumnMap.put("requestor", "rqst\_cntxt\_cd");

dbToUIColumnMap.put("status", "rqst\_stat\_cd");

dbToUIColumnMap.put("firstName", "first\_nam");

dbToUIColumnMap.put("lastName", "lst\_nam");

dbToUIColumnMap.put("createdDate", "crte\_ts");

dbToUIColumnMap.put("viewId", "vw\_rqst\_id");

dbToUIColumnMap.put(Constants.***NUM\_DAYS\_IN\_QUEUE***, Constants.***NUM\_DAYS\_IN\_QUEUE***);

**return** dbToUIColumnMap;

}

@When("execute post call with request payload $requestBody")

@Alias("execute POST call with request payload $requestBody")

**public** **void** executePostAPI(@Named("$requestBody") String requestBody) {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

contextVariableTestData.setContextVariable(***REQUEST\_PAYLOAD***, requestBody);

commonFunctions.executePostWithRequestSpecification(requestSpec.body(requestBody));

}

@When("execute post call with payload $requestBody, header key '$key' and random UUID value")

**public** **void** executePostAPIWithRandomCorelationID(@Named("$requestBody") String requestBody,

@Named("$key") String key) {

randomUUID = UUID.*randomUUID*().toString();

executePostAPIWithcorrelationId(requestBody, key, randomUUID);

}

@When("execute post call with payload $requestBody, header key '$key' and value '$value'")

**public** **void** executePostAPIWithcorrelationId(@Named("$requestBody") String requestBody, @Named("$key") String key,

@Named("$value") String value) {

Map<String, String> header = **new** HashMap<>();

header.put(key, value);

randomUUID = value;

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

requestSpec.headers(header);

contextVariableTestData.setContextVariable(***REQUEST\_PAYLOAD***, requestBody);

commonFunctions.executePostWithRequestSpecification(requestSpec.body(requestBody));

}

@When("execute post call with request body $requestBody for user '$user' and password '$password'")

**public** **void** executePostAPIWithAuthentication(@Named("$requestBody") String requestBody, @Named("$user") String user,

@Named("$password") String password) {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

contextVariableTestData.setContextVariable(***REQUEST\_PAYLOAD***, requestBody);

commonFunctions.executePostWithReqSpeWithAuthorization(requestSpec.body(requestBody),

user, password);

}

@When("execute post call with multipart files :'$multipartsFiles' and form payload '$key' and '$requestBody' for user '$user' and password '$password'")

**public** **void** executePostAPIWithAuthenticationForm(@Named("$multipartsFiles") List<String> multipartsFiles,

@Named("$key") String key, @Named("$requestBody") String requestBody, @Named("$user") String user,

@Named("$password") String password) {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

requestSpec.contentType("multipart/form-data");

**for** (String files : multipartsFiles) {

String[] keyAndValue = StringUtils.*split*(files, "=");

File f = **new** File(Constants.***UPLOADFILEPATH*** + keyAndValue[1]);

requestSpec.multiPart(keyAndValue[0], f);

}

requestSpec.multiPart(key, Base64.*getEncoder*().encodeToString(requestBody.getBytes(StandardCharsets.***UTF\_8***)));

contextVariableTestData.setContextVariable(***REQUEST\_PAYLOAD***, requestBody);

commonFunctions.executePostWithReqSpeWithAuthorization(requestSpec, user, password);

}

@Then("attribute $path save to context as key $attributeKey from response")

**public** **void** saveIDIntoContext(@Named("$path") String path, @Named("$attributeKey") String attributeKey) {

contextVariableTestData.setContextVariable(attributeKey, commonFunctions.getResponseFromContextProvider().path(path).toString());

}

@Then("response save to context as $attributeKey")

**public** **void** saveResponseIntoContext(@Named("$attributeKey") String attributeKey) {

contextVariableTestData.setContextVariable(attributeKey, commonFunctions.getResponseFromContextProvider().asString());

}

@Then("validate error response $jsonResponse for invalid scenarios")

**public** **void** validateErrorMessage(@Named("jsonResponse") String expectedJson) {

ErrorResponse actualErrorMessage = commonFunctions.parseResponse(commonFunctions.getResponseFromContextProvider().getBody().asString(),

ErrorResponse.**class**);

ErrorResponse expectedErrorMessage = commonFunctions.parseResponse(expectedJson, ErrorResponse.**class**);

Assertions.*assertThat*(actualErrorMessage).as("error response is incorrect").isEqualTo(expectedErrorMessage);

}

@When("execute GET call")

**public** **void** executeGETAPI() {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

commonFunctions.executeGetWithRequestSpecification(requestSpec);

}

@When("execute authorized GET call for user '$user' and password '$password'")

**public** **void** executeGETAPIWithAuthentication(@Named("$user") String user, @Named("$password") String password) {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

commonFunctions.executeGetWithReqSpeWithAuthorization(requestSpec, user, password);

}

@Then("validate the Json response $responseBody")

**public** **void** validateJsonResponse(@Named("$responseBody") String responseBody) **throws** IOException {

commonFunctions.compareJSONStrings(commonFunctions.getResponseFromContextProvider().asString(), responseBody);

}

@Then("validate the Json array response $responseBody")

**public** **void** validateJsonArrayResponse(@Named("$responseBody") String responseBody) **throws** IOException {

commonFunctions.compareJSONArrays(commonFunctions.getResponseFromContextProvider().asString(), responseBody);

}

@Then("validate the Json array without JSON Object $responseBody")

**public** **void** validateJsonArrayWithValueResponse(@Named("$responseBody") String responseBody) {

commonFunctions.compareJSONArraysWithoutJsonObject(commonFunctions.getResponseFromContextProvider().asString(), responseBody);

}

@When("execute DELETE call")

**public** **void** executeDELETEAPI() {

requestSpec = (RequestSpecification) contextVariableTestData.getContextVariable(***REQUEST\_SPECIFICATION***);

commonFunctions.executeDeleteWithRequestSpecification(requestSpec);

}

@When("the generated Correlation ID is saved to context as $key")

**public** **void** saveCorrelationIDIntoContext(@Named("$key") String key) {

contextVariablesSteps.saveContextVariable(key, randomUUID);

}

}

**package** com.mastercard.testing.gdp.api.reporting;

**import** com.epam.ta.reportportal.log4j.appender.ReportPortalAppender;

**public** **class** WrapRPAppender **extends** ReportPortalAppender {

**public** WrapRPAppender() {

**super**();

}

}

**package** com.mastercard.testing.gdp.api.model;

**import** java.util.ArrayDeque;

**import** java.util.Deque;

**public** **class** Story {

**private** org.jbehave.core.model.Story jbehaveStory;

**private** Deque<Scenario> scenarios;

**private** Result result;

**public** Story(org.jbehave.core.model.Story story) {

**this**.jbehaveStory = story;

**this**.result = Result.***NOT\_EXECUTED***;

**this**.scenarios = **new** ArrayDeque<>();

}

**public** Result getResult() {

**return** result;

}

**public** **void** setResult(Result result) {

**this**.result = result;

}

**public** org.jbehave.core.model.Story getStory() {

**return** jbehaveStory;

}

**public** String getName() {

**return** jbehaveStory.getName().split("[.]")[0];

}

**public** **void** addScenario(Scenario scenario) {

**this**.scenarios.addFirst(scenario);

}

**public** Scenario getLastScenario() {

**return** scenarios.peekFirst();

}

**public** Scenario getLastFailedScenario() {

**for** (Scenario scenario : scenarios) {

**if**(scenario.getResult().equals(Result.***FAILED***)){

**return** scenario;

}

}

**return** **null**;

}

**public** Scenario removeLastScenario() {

Scenario scenario = **null**;

**if**(!scenarios.isEmpty()) {

scenario = scenarios.removeFirst();

}

**return** scenario;

}

}

**package** com.mastercard.testing.gdp.api.helper.constants;

**public** **final** **class** LinkConstants {

// Application base URL constants

**public** **static** **final** String ***GDP\_DATA\_ACCESS\_BASE\_URL*** = "gdp.dataaccess.base.url";

**public** **static** **final** String ***GDP\_FILE\_SERVICE\_BASE\_URL*** = "gdp.fileservice.base.url";

**public** **static** **final** String ***GDP\_CMG\_BASE\_URL*** = "gdp.cmg.base.url";

**public** **static** **final** String ***GDP\_CMG\_NEW\_BASE\_URL*** = "gdp.cmg.new.base.url";

**public** **static** **final** String ***GDP\_AOB\_APP\_BASE\_URL*** = "gdp.aob.app.base.url";

**public** **static** **final** String ***GDP\_AOB\_APP\_BASE\_URLS*** = "gdp.aob.app.base.urls";

**public** **static** **final** String ***GDP\_B2B\_APP\_BASE\_URL*** = "gdp.b2b.app.base.url";

**public** **static** **final** String ***GDP\_B2C\_APP\_BASE\_URL*** = "gdp.b2c.app.base.url";

**public** **static** **final** String ***GDP\_PUBLIC\_BASE\_URL*** = "gdp.public.base.url";

**public** **static** **final** String ***GDP\_DGR\_APP\_BASE\_URL*** = "gdp.dgr.base.url";

**public** **static** **final** String ***DGR\_FILE\_SERVICE\_API*** = "/dgr-file-service-api";

**public** **static** **final** String ***GDP\_FILE\_SERVICE*** = "/dgr-file-service-api/batchJob/startJob/createAttorneyCSVBatch";

// Application resource constants

**public** **static** **final** String ***GDP\_DATA\_ACCESS*** = "/gdp-dataaccess-api";

**public** **static** **final** String ***GDP\_AOB\_APP\_API*** = "/dgr-aob-app-api";

**public** **static** **final** String ***GDP\_B2C\_APP\_API*** = "/gdp-b2c-app-api";

**public** **static** **final** String ***GDP\_B2B\_APP\_API*** = "/dgr-b2b-app-api";

**public** **static** **final** String ***GET\_CMG\_EMAIL\_SERVICE*** = "/gdp-cmg-mock-email-service";

**public** **static** **final** String ***GET\_CMG\_EMAIL\_SERVICE\_NEW*** = "/gdp-cmg-mock-email-service/email/";

// Other end-point related constants

**public** **static** **final** String ***REQUEST\_TYPE\_DELETE*** = "?requestType=D";

**public** **static** **final** String ***REQUEST\_TYPE\_UPDATE*** = "?requestType=U";

**public** **static** **final** String ***REQUEST\_TYPE\_VIEW*** = "?requestType=V";

**public** **static** **final** String ***PRODUCT\_SUMMARY\_MARKED\_DELETE\_RECORDS*** = "/productSummary/%s/markDeleted";

**public** **static** **final** String ***PRODUCT\_SUMMARY\_MARKED\_UPDATE\_RECORDS*** = "/productSummary/%s/markUpdated";

**public** **static** **final** String ***PRODUCT\_SUMMARY\_MARKED\_UPDATE\_RECORDS\_AOB*** = "/productSummary/%s/markUpdatedForAOB";

**public** **static** **final** String ***GET\_PRIVACY\_REQUESTS*** = "/privacyrequests/";

**public** **static** **final** String ***PRIVACY\_REQUEST*** = "/privacyRequest/";

**public** **static** **final** String ***PRIVACY\_RESPONSE*** = "/privacyresponse";

**public** **static** **final** String ***HEALTH\_CHECK*** = "/health/";

**public** **static** **final** String ***APPROVE\_PRIVACY\_REQUEST*** = "/privacy-requests/%s/dispatch";

**public** **static** **final** String ***GET\_SEARCH\_KEY*** = "/searchKeys/";

**public** **static** **final** String ***GET\_CMG\_EMAIL\_DATA*** = "/rest/emaildata/";

**public** **static** **final** String ***GET\_CMG\_ALL\_EMAIL\_DATA*** = "/email/all/";

**public** **static** **final** String ***GET\_PRODUCT\_SUMMARY*** = "/viewProductSummary/";

**public** **static** **final** String ***MANUAL\_RESPONSE\_SUBMIT*** = "/manualResponse/submit";

**public** **static** **final** String ***PRIVACY\_RESPONSE\_COMMENTS*** = "/privacyResponse/comments";

**public** **static** **final** String ***PRIVACY\_RESPONSE\_RECORD\_COMMENTS*** = "/responseRecord/comments";

**public** **static** **final** String ***MANUAL\_RESPONSE\_CONVERT\_CSV\_SUBMIT*** = "/manualResponse/submit/csv";

**public** **static** **final** String ***SUBMIT\_REVIEW\_DATA*** = "/submitReviewData/";

**public** **static** **final** String ***PRVCY\_RQST\_BY\_PAGE*** = "/getPrivacyRequestsByPage?";

**public** **static** **final** String ***GET\_DUPLICATE\_FOUND*** = "duplicates";

**public** **static** **final** String ***GET\_PRODUCT\_PRIVACY\_REQUEST*** = "/productPrivacyRequest/";

**public** **static** **final** String ***SRVC\_FUNC\_PRVCY\_RQST*** = "/serviceFunctionPrivacyRequest/";

**public** **static** **final** String ***PRIVACY\_RESPONSE\_STATUS*** = "/privacy-response-status";

**public** **static** **final** String ***PRODUCT\_SUMMARY*** = "/productsummary";

**public** **static** **final** String ***PUBLISH*** = "/publish";

**public** **static** **final** String ***PUBLISHED\_RECORDS*** = "/published-records";

**public** **static** **final** String ***PRODUCTS*** = "/productResults/";

**public** **static** **final** String ***BFF\_PRODUCTS*** = "/products/";

**public** **static** **final** String ***GET\_ORIGINAL\_REQUEST*** = "/productSummary/originalRequest/";

**public** **static** **final** String ***GET\_TRANSLATED\_ELEM\_TAGS*** = "/dataElementLabel/";

**public** **static** **final** String ***RESPONSE\_RECORD*** = "/responseRecord/";

**public** **static** **final** String ***MAINTAIN\_STATUSES*** = "/maintainStatuses/";

**public** **static** **final** String ***AOB\_VIEW\_DISCLOSED\_PRODUCTS*** = "/viewDisclosedProductSummaryByPrivacyRequestId/";

**public** **static** **final** String ***RESPONSE\_COMMENTS*** = "comments/";

**public** **static** **final** String ***VIEW\_DELETED\_PRODUCTS*** = "/viewDeletedProductSummaryByPrivacyRequestId/";

**public** **static** **final** String ***GET\_REQ\_TYPE\_LOOKUP*** = "/lookUp/prvcy\_rqst\_type\_lkp";

**public** **static** **final** String ***SELECTIVE\_DELETE\_CONFIRMATION*** = "/privacyRequest/deleteRequest";

**public** **static** **final** String ***BUSINESS\_CONTEXT\_LOOKUP*** = "/lookUp/busn\_cntxt\_lkp";

**public** **static** **final** String ***EXPORT\_PRODUCTS*** = "/export/products/";

**public** **static** **final** String ***B2C\_AUTH\_API*** = "/gdp-b2c-app-api/requestDataOwnerDetails/testAPI/";

**public** **static** **final** String ***DATA\_ACCESS\_AUTH\_API*** = "/requestDataOwnerDetails/{pathParam}/";

**public** **static** **final** String ***B2C\_AUTHENTICATION\_URL*** = "http://ech-10-157-134-194.mastercard.int:25003/gdp-b2c-app-api/requestDataOwnerDetails/testAPI/";

**public** **static** **final** String ***PROD\_TRANSLATED\_ELEM\_TAGS*** = "/labelTranslation/";

**public** **static** **final** String ***EXPORT*** = "export/";

**public** **static** **final** String ***LOOK\_UP\_API*** = "/lookUp/";

**public** **static** **final** String ***EXPORT\_THRESHOLD*** = "/exportThreshold/";

**public** **static** **final** String ***FILTER\_BY*** = "?filterBy=";

**public** **static** **final** String ***OFFSET*** = "offset=%s";

**public** **static** **final** String ***LIMIT*** = "limit=%s";

**public** **static** **final** String ***PRIVACY\_LAW*** = "/privacy-law";

**public** **static** **final** String ***STATE\_CODE*** = "stateCode";

**public** **static** **final** String ***ACTIVITY\_DATA***="/activity-data";

**public** **static** **final** String ***FILES***="/files";

**public** **static** **final** String ***FILEDETAILS***="/file-details";

**public** **static** **final** String ***DOWNLOADS***="/downloads";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_DETAILS*** = "/privacyRequest/requestDetails/";

**public** **static** **final** String ***VIEW\_DISCLOSED\_PRODUCT\_SUMMARY*** = "/productSummary/disclosed/";

**public** **static** **final** String ***CONSUMER\_NOTIFICATION\_REQUEST*** = "/notification/consumer/en-us";

**public** **static** **final** String ***ATTORNEY\_NOTIFICATION\_REQUEST*** = "/notification/attorney/en-us";

**public** **static** **final** String ***REQUEST*** = "/request?";

**public** **static** **final** String ***PRODUCTS\_RESPONSE*** = "/response/products?";

**public** **static** **final** String ***RECORDS\_RESPONSE*** = "/response/records?";

**public** **static** **final** String ***PRODUCT\_IDS*** = "productIds=%s";

**public** **static** **final** String ***RECORDIDS*** = "recordIds=%s";

**public** **static** **final** String ***DISPATCH*** = "/privacyRequest/%s/dispatch";

**public** **static** **final** String ***GDP\_DELETE\_PRODUCT\_SUMMARY*** = "/deleteProductSummary/B2C/";

**public** **static** **final** String ***GDP\_AGENT\_API*** = "/agent-requests/";

**public** **static** **final** String ***GDP\_VIEW\_SUMMARY\_DELETE*** = "/viewProductSummaryByDeletePrivacyRequestId/";

**public** **static** **final** String ***PUT\_APPROVE\_REQUEST*** = "/attorneyReview/privacyRequest/%s/review";

**public** **static** **final** String ***PUT\_APPROVE\_REQUEST\_DA*** = "/attorneyReview/privacyRequest/review";

**public** **static** **final** String ***PUT\_PRIVACY\_REQUEST\_REJECT*** = "/privacyRequest/reject/";

**public** **static** **final** String ***SAVE\_UPDATE\_REQUEST*** = "/privacyRequest/updateRequest/";

**public** **static** **final** String ***GET\_PRODUCT\_SUMMARY\_UPDATE\_REQUEST*** = "/productSummary/updateRequest/";

**public** **static** **final** String ***VIEW\_DISCLOSED\_PRODUCT\_SUMMARY\_WITH\_LEGAL\_NOTE*** = "/productSummaryWithLegalNote/disclosed/";

**public** **static** **final** String ***LOCALE\_US\_ENGLISH*** = "en-us";

**public** **static** **final** String ***GET\_REJECT\_REASONS*** = "/lookUp/rej\_rsn\_lkp";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_REJECTION\_SUMMARY*** = "/privacyRequest/rejectionsummary/";

**public** **static** **final** String ***VIEW\_DELETE\_DISCLOSED\_PRODUCT\_SUMMARY*** = "/disclosedProductSummary/";

**public** **static** **final** String ***B2BPRODUCTS*** = "products/";

**public** **static** **final** String ***GET\_PRIVACY\_REQUESTS\_B2B*** = "/privacyRequests?";

**public** **static** **final** String ***BATCH\_JOB\_FILE*** = "/start/batchJob/fileservice/%s";

**public** **static** **final** String ***GET\_DELETE\_REQUEST\_REVIEW*** = "/productSummary/deleteRequest/%s/review";

**public** **static** **final** String ***PUT\_APPROVE\_REQUEST\_VIEW\_UPDATE*** = "/approve";

**public** **static** **final** String ***BUSINESS\_NOTIFICATION\_REQUEST*** = "/notification/business/en-us";

**public** **static** **final** String ***VIEW\_UPDATE\_DISCLOSED\_PRODUCT\_SUMMARY*** = "/productSummaryWithUpdatedRecords/";

**public** **static** **final** String ***GET\_UPDATE\_REQUESTER\_COMMENTS*** = "/privacyRequest/%s/requestComments";

**public** **static** **final** String ***REQUEST\_TYPE*** = "?type=%s";

**public** **static** **final** String ***SELECTIVE\_UPDATE\_SAVE\_REQUEST*** = "/privacyRequest/updateRequest/en-us";

**public** **static** **final** String ***GET\_DELETE\_ALL\_REVIEW\_ORIGINAL\_REQUEST*** = "/privacyRequest/requestReviewDetails/";

**public** **static** **final** String ***BATCH\_JOB*** = "/start/batchJob/%s";

**public** **static** **final** String ***BATCH\_JOB\_NEW*** = "/batchJob/startJob/%s";

**public** **static** **final** String ***BATCH\_JOB\_DELETE*** ="/activity-data/chunks/";

**public** **static** **final** String ***APPROVE\_DA\_PRIVACY\_REQUEST*** = "/privacyRequest";

**public** **static** **final** String ***GET\_PRODUCT\_SUMMARY\_UPDATE*** = "/disclosedProductSummaryByPrivacyRequestIdForUpdateRequest/";

**public** **static** **final** String ***GET\_PRODUCT\_SUMMARY\_OPEN\_UPDATE*** = "/productSummary/update/request/";

**public** **static** **final** String ***PRIVACY\_REQUEST\_MANUAL\_RESPONSE*** = "/privacyRequestForManualResponse/";

**public** **static** **final** String ***ID\_AND\_V\_UPLOAD\_API*** = "/upload/identityDocuments";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_ACTIONS*** = "/privacyRequest/actions/%s";

**public** **static** **final** String ***LATEST\_MESSAGE\_TO\_REQUESTOR*** = "/privacyRequest/actions/%s/latest/comment?commentType=E";

**public** **static** **final** String ***ACTIONS*** = "actions";

**public** **static** **final** String ***JSONPARSERESPONSE*** = "JsonParseException: %s";

**public** **static** **final** String ***JSONMAPPINGRESPONSE*** = "JsonParseException: %s";

**public** **static** **final** String ***EXCEPTIONWHILEPARSINGEXPIREDATE*** = "Exception occurred while parsing expiryDate : %s";

**public** **static** **final** String ***EXPIREDATEISNOTINSPECIFIEDFORMAT*** = "Submitted expiryDate is not in the specified format";

**public** **static** **final** String ***IOEXCEPTION*** = "IOException: %s";

**public** **static** **final** String ***FETCH\_IMAGE\_ID\_AND\_V*** = "/identity-documents/";

**public** **static** **final** String ***UPDATE\_RESPONSE\_DATA*** = "updateResponseData";

**public** **static** **final** String ***DELETE\_REASON\_LOOKUP*** = "/lookUp/id\_proof\_dcln\_rsn\_lkp";

**public** **static** **final** String ***IDV\_APPROVE*** = "/identityVerification/approve";

**public** **static** **final** String ***IDV\_DECLINE*** = "/identityVerification/decline";

**public** **static** **final** String ***DOCUMENTS\_PUBLISH*** = "/documents/publish";

**public** **static** **final** String ***DOCUMENTS\_UPLOAD*** = "/documents/upload";

**public** **static** **final** String ***TRANSACTION\_FILES\_UPLOAD*** ="/activity-data/files";

**public** **static** **final** String ***DOCUMENT\_DOWNLOAD*** = "/document/download/";

**public** **static** **final** String ***DOWNLOAD\_DOCUMENT*** = "/download/document/";

**public** **static** **final** String ***SECURE\_DOWNLOAD\_PRODUCT\_VERIFICATION***= "/secure-download/product-verification";

**public** **static** **final** String ***DOCUMENTS*** = "documents/";

**public** **static** **final** String ***AUTO\_COMPLETE*** = "autoComplete/";

**public** **static** **final** String ***BFF*** = "BFF";

**public** **static** **final** String ***DATAACCESS*** = "dataAccess";

**public** **static** **final** String ***IDV\_IMAGE\_VIEW*** = "view";

**public** **static** **final** String ***IDV\_IMAGE\_REVIEW*** = "review";

**public** **static** **final** String ***GET\_IDENTITY\_VERIFICATION\_STATUS*** = "/identityVerification/status/";

**public** **static** **final** String ***UPLOAD\_COMPLETED*** = "upload/complete";

**public** **static** **final** String ***PRIVACYREQUESTID*** = "privacyRequestId";

**public** **static** **final** String ***AMPERSAND*** = "&";

**public** **static** **final** String ***QUESTION\_MARK*** = "?";

**public** **static** **final** String ***FORWARD\_SLASH*** = "/";

**public** **static** **final** String ***DOCUMENTS\_DELETE*** = "/documents/delete";

**public** **static** **final** String ***DOCUMENTS\_SCOPE*** = "/documents/scope";

**public** **static** **final** String ***IDVERIFICATION\_NOTIFICATION*** = "idverification/count";

**public** **static** **final** String ***FILTER*** = "/filter";

**public** **static** **final** String ***COLUMN\_NAME*** = "columnName";

**public** **static** **final** String ***SORT\_ORDER*** = "sortOrder";

**public** **static** **final** String ***FLAGS*** = "flags/";

**public** **static** **final** String ***VIEW\_DISCLOSED\_PRODUCT\_SUMMARY\_WITH\_LEGAL\_NOTE\_DATA\_ACCESS*** = "/privacy-requests/%s/products/disclosed/legal-note-summary";

**public** **static** **final** String ***VIEW\_DISCLOSED\_PRODUCT\_SUMMARY\_DATA\_ACCESS*** = "/privacy-requests/%s/products/disclosed/summary";

**public** **static** **final** String ***GET\_IDENTITY\_VERIFICATION\_STATUS\_DATA\_ACCESS*** = "/privacy-requests/%s/id-verification-status";

**public** **static** **final** String ***PRVCY\_REQ\_SERV\_FUNCTION\_WITH\_SEARCH\_KEY*** = "/privacy-request/service-function/%s/elmt-tags";

**public** **static** **final** String ***GET\_RECORD\_LEVEL\_COMMENTS*** = "/privacy-request/product/records/validate-comments";

**public** **static** **final** String ***PRIVACY\_REQUESTS*** = "/privacy-requests/";

**public** **static** **final** String ***REQUEST\_COMMENTS*** = "/request-comments";

**public** **static** **final** String ***AUTO\_COMPLETE\_NEW*** = "/auto-complete";

**public** **static** **final** String ***REVIEW*** = "/review";

**public** **static** **final** String ***DATA\_ACCESS\_API*** = "dataAccess";

**public** **static** **final** String ***AOB\_PORTAL\_CODE*** = "AOB";

**public** **static** **final** String ***B2C\_PORTAL\_CODE*** = "B2C";

**public** **static** **final** String ***B2B\_PORTAL\_CODE*** = "B2B";

**public** **static** **final** String ***PRIVACYREQUEST*** = "/privacy-request/";

**public** **static** **final** String ***IDENTITY\_UPLOAD*** = "poi";

**public** **static** **final** String ***DOCUMENT\_COMMENTS*** = "/document/%s/comments";

**public** **static** **final** String ***DOCUMENT\_DETAILS*** = "/document/";

**public** **static** **final** String ***DOCUMENTS\_LIST*** = "/privacy-requests/%s/documents";

**public** **static** **final** String ***GET\_DOCUMENT\_LIST\_FOR\_OPEN\_DELETE*** = "/privacy-request/%s/open-delete/documents";

**public** **static** **final** String ***MANUAL\_RESPONSE\_CONVERT\_EXCEL\_SUBMIT*** = "/manualResponse/submit/excel";

**public** **static** **final** String ***DECOUPLE\_SERVICE\_FUNCTION*** = "/privacy-request/service-function/";

**public** **static** **final** String ***VALIDATE*** = "/validate";

**public** **static** **final** String ***NR*** = "NR";

**public** **static** **final** String ***DOCUMENT\_GRANULAR\_REVIEW*** = "/document/review";

**public** **static** **final** String ***RECORDS\_REVIEW\_SUMMARY*** = "/product/%s/reviewed/record-summary";

**public** **static** **final** String ***DOCUMENTS\_ACTION\_COMPLETE*** = "/documents-action/complete";

**public** **static** **final** String ***GRANULAR\_REVIEW\_ELIGIBILITY*** = "/privacy-requests/%s/review-complete-eligibility";

**public** **static** **final** String ***STATUS*** = "status";

**public** **static** **final** String ***GET\_PROVINCES*** = "/%s/provinces";

**public** **static** **final** String ***GET\_GLOBAL\_DATA*** = "%s/global/data";

**public** **static** **final** String ***REQUEST\_SELECTION\_TYPE*** = "requestSelectionType";

**public** **static** **final** String ***MARK\_DELETED*** = "markDeleted";

**public** **static** **final** String ***SORT\_BYCOLUMN\_NAME*** = "sortByColumnName";

**public** **static** **final** String ***BFF\_PRODUCT*** = "/product/";

**public** **static** **final** String ***NEEDS\_REVIEW*** = "needs-review";

**public** **static** **final** String ***GET\_DELETE\_UPDATE\_GRANULAR\_REVIEW*** = "/productSummary/%sRequest/%s/review";

**public** **static** **final** String ***IDENTITY\_DOCUMENT\_CHUNKS*** = "/identity-document-chunks/";

**public** **static** **final** String ***NEW\_REQUEST\_THRESHOLD*** = "/privacyRequest/new-request-threshold";

**public** **static** **final** String ***COUNTRIES\_API*** = "/countries";

**public** **static** **final** String ***GDP\_FILTER\_REGION*** = "/region/";

**public** **static** **final** String ***GDP\_FILTER\_REGION\_LOOKUP*** = "/lookUp/regn\_lkp";

**public** **static** **final** String ***MERGEDATA*** = "MergeData/";

**public** **static** **final** String ***SELECT\_ATTRIBUTE*** = "select";

**public** **static** **final** String ***XSL\_VALUE\_NODE*** = "xsl:value-of";

**public** **static** **final** String ***UPLOADEMAILFILEPATH*** = "src/main/resources/config/emailTemplates/";

**public** **static** **final** String ***XSL\_FILE\_EXTENSION*** = ".xsl";

**public** **static** **final** String ***COMA*** = ",";

**public** **static** **final** String ***GDP\_EMAIL\_SERVICE\_BASE\_URL*** = "gdp.email.service.base.url";

**public** **static** **final** String ***EMAIL\_SERVICE*** = "/gdp-email-service/mail";

**public** **static** **final** String ***GET\_BODY\_REQUESTURL\_VIEW\_TYPE*** = "https://www.mastercard.%s/secure/my-data/gdp-b2c-web/%s/#/dashboard?locale=%s&target=%252Fviewrequest%253FrequestId%253D%s";

**public** **static** **final** String ***GET\_BODY\_REQUESTURL\_DELETE\_TYPE*** = "https://www.mastercard.%s/secure/my-data/gdp-b2c-web/%s/#/dashboard?locale=%s&target=%252FdeleteRequest%252Fview%252F%s%252F%s%253FrequestType%253D%s";

**public** **static** **final** String ***EMAIL\_PROPERTIES\_FILEPATH*** = "src/main/resources/config/emailTemplates/emailTemplate.properties";

**public** **static** **final** String ***DATA*** = "/data";

**public** **static** **final** String ***REQUESTURL\_DA*** = "%252FdeleteRequest%252Fview%252F<REQUESTID>%252F<REQUESTID>%253FrequestType%253DDA";

**public** **static** **final** String ***REQUESTURL\_VIEW*** = "%252Fviewrequest%253FrequestId%253D<REQUESTID>";

**public** **static** **final** String ***REQUESTURL\_DELETE*** = "%252FdeleteRequest%252Fview%252F<REQUESTID>%252F<VWREQUESTID>%253FrequestType%253DD";

**public** **static** **final** String ***REQUESTURL\_UPDATE*** = "%252Fupdaterequest%252Fview%252F<REQUESTID>%252F<VWREQUESTID>";

**public** **static** **final** String ***REQUESTURL\_VIEW\_DECLINED*** = "%252Fdeclined%252F<REQUESTID>";

**public** **static** **final** String ***DASHBOARD\_COLUMN*** = "/dashboard/%s/preferred-columns";

**public** **static** **final** String ***PREFFERED\_DASHBOARD\_COLUMN*** = "/dashboard/user-preferred-columns";

**public** **static** **final** String ***GET\_DASHBOARD\_COLUMNS*** = "/dashboard/%s/columns";

**public** **static** **final** String ***GET\_PREFFERED\_DASHBOARD\_COLUMNS*** = "/dashboard-columns";

**public** **static** **final** String ***REQUESTURL\_POIAPPR*** = "%252Fviewrequest%253FrequestId%253D84759401%2526docSummary%253Dtrue";

**public** **static** **final** String ***VERIFY\_CARD*** = "/card/verify";

**public** **static** **final** String ***VERIFY\_CARD\_DA*** = "/verifycard";

**public** **static** **final** String ***EDIT*** = "edit/";

**public** **static** **final** String ***BANNER\_NOTIFICATION*** = "/notifications";

**public** **static** **final** String ***PORTAL\_TYPE*** = "?portalType=%s";

**public** **static** **final** String ***REVIEW\_REQUEST\_COUNT*** = "review-requests/count";

**public** **static** **final** String ***COUNTRY\_CODE*** = "countryCode";

**public** **static** **final** String ***CARD\_VERIFY*** = "/verifycard";

**public** **static** **final** String ***OVERRIDE\_VERIFICATION\_SWITCH*** = "/override-verification";

**public** **static** **final** String ***GDP\_AGENT\_API\_CONSTANT*** = "/gdp-agent-api";

**public** **static** **final** String ***CONSUMER\_CONSENT*** = "/privacy-request/consumer-confirmation";

**public** **static** **final** String ***OVERRIDE\_VERIFICATION*** = "/privacy-requests/customer-id-verifications/override";

**public** **static** **final** String ***AGENT\_PRIVACY\_REQUEST*** = "/privacyData";

**public** **static** **final** String ***VIEW\_DISCLOSED\_PRODUCT\_SUMMARY\_LEGALNOTE*** = "/productSummaryWithLegalNote/disclosed/";

**public** **static** **final** String ***AGENT\_LOOK\_UP\_API*** = "/lookup/";

**public** **static** **final** String ***ASYNC*** = "async";

**public** **static** **final** String ***FILES\_STATUS*** = "/files/status";

**public** **static** **final** String ***ACTIVITY\_DATA\_FILES*** = "/activity-data/%s/files";

**public** **static** **final** String ***FILE\_RESPONSE\_RECORD*** = "/fileResponseRecord";

**public** **static** **final** String ***ACTION\_HISTORY*** = "/actionHistory";

**public** **static** **final** String ***DRAFT*** = "DR";

**public** **static** **final** String ***BULK\_REQUEST\_GROUPP\_ASSIGNMENT*** = "BULK\_RQST\_GRP\_ASSGN";

**public** **static** **final** String ***GDP\_B2C\_DGR\_APP\_BASE\_URL*** = "dgr.reportingService.base.url";

**public** **static** **final** String ***DGR\_REPORTING\_APP\_API*** = "/dgr-reporting-service-api";

**public** **static** **final** String ***MY\_DATA\_METRICS*** = "/my-data-metrics";

**private** LinkConstants() {

}

}

**package** com.mastercard.testing.gdp.api.helper.constants;

**public** **final** **class** QueryConstants {

**public** **static** **final** String ***GET\_DELETE\_PRIVACY\_REQUEST\_ID\_BY\_RECORD\_ID*** = "select del\_prvcy\_rqst\_id from gdp\_owner.resp\_rec where resp\_rec\_id = %d;";

**public** **static** **final** String ***GET\_DELETE\_PRODUCT\_REQUEST\_ID\_BY\_RECORD\_ID*** = "select del\_prdct\_rqst\_id from resp\_rec where resp\_rec\_id = %d;";

**public** **static** **final** String ***GET\_DELETE\_SERVICE\_FUNCTION\_REQUEST\_ID\_BY\_RECORD\_ID*** = "select del\_serv\_func\_rqst\_id from resp\_rec where resp\_rec\_id = %d;";

**public** **static** **final** String ***UPDATE\_RESP\_REC\_DEL\_STATUS\_B2C*** = "update resp\_rec set del\_prvcy\_rqst\_id=null where resp\_rec\_nam like '%s'";

**public** **static** **final** String ***GET\_AUDIT\_DESCRIPTION\_BY\_RECORD\_ID*** = "select aud\_desc from aud\_hist where aud\_elmt\_key\_txt = '%s' and aud\_type\_cd='RQST\_DEL';";

**public** **static** **final** String ***GET\_AUDIT\_DATA\_TEXT\_BY\_RECORD\_ID*** = "select aud\_data\_txt from aud\_hist where aud\_elmt\_key\_txt = '%s' and aud\_type\_cd='RQST\_DEL';";

**public** **static** **final** String ***GET\_DELETE\_REASON\_DESCRIPTION*** = "select type\_desc from del\_rsn\_lkp where type\_cd='%s';";

**public** **static** **final** String ***GET\_USERID\_BY\_RECORD\_ID*** = "select aud\_user\_id from aud\_hist where aud\_elmt\_key\_txt = '%s' and aud\_type\_cd='RQST\_DEL';";

**public** **static** **final** String ***GET\_PORTAL\_TYPE\_CODE\_BY\_DELETE\_REQUEST\_ID*** = "select portal\_type\_cd from prvcy\_rqst where prvcy\_rqst\_id=%s;";

**public** **static** **final** String ***GET\_RECORD\_COUNT\_OF\_SERVICE\_FUNCTION\_PRIVACY\_REQUESTS*** = "select count(\*) from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s' and resp\_rcv\_ts IS null;";

**public** **static** **final** String ***GET\_RESPONSE\_PAYLOAD*** = "select prvcy\_resp\_pyld\_txt from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_COUNT\_OF\_PRIVACY\_REQUESTS*** = "SELECT count(\*) FROM prvcy\_rqst WHERE prvcy\_rqst\_id = %s AND rqst\_stat\_cd = '%s';";

**public** **static** **final** String ***GET\_RECORD\_COUNT\_AUDIT*** = "select count(\*) from aud\_hist where aud\_elmt\_key\_txt = '%d' and aud\_type\_cd = '%s';";

**public** **static** **final** String ***GET\_RECORD\_AUDIT\_DESCRIPTION*** = "select aud\_desc from aud\_hist where aud\_elmt\_key\_txt = '%d' and aud\_type\_cd = '%s';";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_RECORD\_COUNT*** = "select count(\*) from prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_FOR\_GIVEN\_REQUEST\_ID*** = "select \* from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRODUCT\_REVIEW\_TIMESTAMP\_FROM\_PRODUCT\_ACTION\_STATUS\_TABLE\_FOR\_GIVEN\_REQUEST\_ID*** = "select prdct\_rvw\_ts from prdct\_actn\_stat where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_CHILD\_RECORDS*** = "select \* from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_PARENT\_RECORD\_COUNT\_WITH\_IN\_PROGRESS\_STATUS*** = "select count(\*) from prvcy\_rqst where prvcy\_rqst\_id = '%s' and rqst\_stat\_cd = 'I';";

**public** **static** **final** String ***GET\_CHILD\_RECORD\_COUNT\_WITH\_RESPONSE\_RECEIVED*** = "select count(\*) from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = '%s' and serv\_func\_cd = '%s' and resp\_rcv\_ts IS NOT NULL;";

**public** **static** **final** String ***GET\_PRODUCT\_PRIVACY\_REQUESTS\_FOR\_GIVEN\_REQUEST\_ID*** = "select \* from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRODUCT\_CODES\_FOR\_GIVEN\_DOMAIN*** = "select distinct prdct\_cd from serv\_def where serv\_cd in (select distinct serv\_cd from serv\_func\_def where busn\_cntxt\_cd = '%s');";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTION\_PRIVACY\_FOR\_GIVEN\_REQUEST\_ID*** = "select \* from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRODUCT\_PRIVACY\_REQUEST\_IDS\_FOR\_A\_GIVEN\_REQUEST\_ID*** = "select prdct\_prvcy\_rqst\_id from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_COUNT\_OF\_PRODUCT\_PRIVACY\_REQUESTS\_FOR\_A\_GIVEN\_REQUEST\_ID\_PRODUCT*** = "select count(\*) from prdct\_prvcy\_rqst where prdct\_cd = '%s' and prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_COUNT\_OF\_SERVICE\_FUNCTION\_PRIVACY\_REQUESTS\_FOR\_A\_GIVEN\_REQUEST\_ID\_PRODUCT*** = "select count(\*) from serv\_func\_prvcy\_rqst where prdct\_cd = '%s' and prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_RESPONSE\_RECORD\_COUNT*** = "select count(\*) from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_RESPONSE\_RECORD\_FOR\_GIVEN\_REQUEST\_ID*** = "select \* from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s ORDER BY resp\_rec\_id ASC;";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTION\_STATUS\_CODE*** = "select serv\_func\_rqst\_stat\_cd from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_SF\_PRVCY\_RQST\_ID*** = "select serv\_func\_prvcy\_rqst\_id from gdp\_owner.serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s AND serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_PRIVACY\_RQST\_ID*** = "select prdct\_prvcy\_rqst\_id from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s AND serv\_func\_cd = '%s';";

**public** **static** **final** String ***UPDATE\_RESP\_REC\_STATUS*** = "update resp\_rec set resp\_rec\_stat\_cd = '%s' where serv\_func\_prvcy\_rqst\_id = %s AND prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_RESP\_RECORDS\_FOR\_GIVEN\_IDS*** = "select \* from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s AND prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_PRODUCT\_REQUEST\_STATUS*** = "update prdct\_prvcy\_rqst set prdct\_rqst\_stat\_cd = '%s' where prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_PRODUCT\_REQUEST\_PUBLISH\_DETAILS*** = "update prdct\_prvcy\_rqst set pblsh\_ts = '%s', pblsh\_user\_id = '%s', pblsh\_user\_nam = '%s' where prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_DETAILS\_FOR\_REQUEST\_ID*** = "select \* from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_RESPONSE\_RECORD\_STATUS\_CODE*** = "select resp\_rec\_stat\_cd from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_SERV\_FUNC\_PRVCY\_RECS*** = "select \* from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_RESP\_RECS*** = "select \* from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_FILECHUNK\_DATA***="SELECT utl\_raw.cast\_to\_varchar2(DBMS\_LOB.SUBSTR(CHUNK\_DATA\_TXT)) FROM FILE\_CNTN\_CHUNK WHERE FILE\_REPO\_ID=%s;";

**public** **static** **final** String ***GET\_FILEREPO\_ID***="SELECT FILE\_REPO\_ID FROM PRVCY\_RESP\_FILES WHERE ACTVY\_DATA\_UID\_ID=%s";

**public** **static** **final** String ***GET\_RESP\_CMNT\_RECS*** = "select \* from resp\_cmnt where resp\_rec\_id = %d order by crte\_ts desc";

**public** **static** **final** String ***GET\_SERV\_FUNC\_DEF\_RECS*** = "select \* from serv\_func\_def where serv\_func\_cd = '%s'";

**public** **static** **final** String ***GET\_PROD\_DEF\_RECS*** = "select \* from prdct\_def where prdct\_cd = '%s'";

**public** **static** **final** String ***GET\_RECORD\_STATUS*** = "select prdct\_rqst\_stat\_cd from prdct\_prvcy\_rqst where prdct\_cd = '%s' AND prvcy\_rqst\_id=%d";

**public** **static** **final** String ***UPDATE\_RESP\_REC\_STATUS\_PARTIAL*** = "update resp\_rec set resp\_rec\_stat\_cd = '%s' where serv\_func\_prvcy\_rqst\_id = %s AND prdct\_prvcy\_rqst\_id = %s AND resp\_rec\_nam like '%%#1';";

**public** **static** **final** String ***GET\_RECORD\_COUNT\_AUDIT\_STATUSES*** = "select count(\*) from aud\_hist where aud\_type\_cd = '%s';";

**public** **static** **final** String ***GET\_RECORD\_AUDIT\_DESCRIPTION\_STATUSES*** = "select aud\_desc from aud\_hist where aud\_type\_cd = '%s';";

**public** **static** **final** String ***GET\_NON\_DISCLOSED\_RECORD\_COUNT*** = "Select count(\*) from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s AND dsclsr\_sw = %b";

**public** **static** **final** String ***INSERT\_COMPLETED\_DATE\_FOR\_PRIVACY\_REQUEST*** = "update prvcy\_rqst set rqst\_cmplt\_ts = '%s' where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***DELETE\_SERV\_FUNCTION\_PRVCY\_RQST\_BY\_ID*** = "Delete from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_PRDCT\_PRVCY\_RQST\_BY\_ID*** = "Delete from prdct\_prvcy\_rqst where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_PRIVACY\_REQUEST\_ACTION\_BY\_PRIVACY\_REQUESTS*** = "Delete from prvcy\_rqst\_actn where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_OPT\_COUNTRY\_BY\_ID*** = "Delete from opt\_cntry where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_PRVCY\_RQST\_BY\_ID*** = "Delete from prvcy\_rqst where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***RESPONSE\_RECORD\_STATUS\_CODE*** = "select resp\_rec\_stat\_cd from resp\_rec where resp\_rec\_id = %d";

**public** **static** **final** String ***RESPONSE\_RECORD\_TIMESTAMP*** = "select resp\_rvw\_ts from resp\_rec where resp\_rec\_id = %d";

**public** **static** **final** String ***PRODUCT\_PUBLISH\_TIMESTAMP*** = "select pblsh\_ts FROM prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id=(select prdct\_prvcy\_rqst\_id from resp\_rec where resp\_rec\_id = %d)";

**public** **static** **final** String ***GET\_PRDCT\_PRVCY\_ID*** = "select prdct\_prvcy\_rqst\_id from prdct\_prvcy\_rqst where prdct\_cd in \r\n"

+ "(select prdct\_cd from serv\_func\_prvcy\_rqst sfpr, resp\_rec rr where rr.serv\_func\_prvcy\_rqst\_id=sfpr.serv\_func\_prvcy\_rqst\_id and resp\_rec\_id=%d) \r\n"

+ "and prvcy\_rqst\_id=%d";

**public** **static** **final** String ***GET\_SERV\_FUNC\_PRVCY\_RECS\_BY\_PRDCT\_PRVCY\_RQST\_ID*** = "select \* from serv\_func\_prvcy\_rqst where prdct\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_SERV\_FUNC\_INTEGRATION\_REC\_BY\_SERV\_FUNC*** = "select \* from serv\_func\_integration where serv\_func\_cd='%s'";

**public** **static** **final** String ***GET\_RESP\_RECS\_BY\_ID*** = "select \* from resp\_rec where resp\_rec\_id = %d";

**public** **static** **final** String ***UPDATE\_RESP\_REC\_DEL\_STATUS*** = "update resp\_rec set del\_prdct\_rqst\_id=null, del\_serv\_func\_rqst\_id=null, del\_prvcy\_rqst\_id=null where resp\_rec\_nam like '%s' or resp\_rec\_id::text like '%s';";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_STATUS*** = "select rqst\_stat\_cd from prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_SERV\_FUNC\_RQST\_STAT\_CD*** = "select serv\_func\_rqst\_stat\_cd from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_COLUMN\_VALUE\_FROM\_SERV\_FUNC\_PRVCY\_RQST*** = "select %s from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_SERV\_FUNC\_RQST\_VER\_NUM*** = "select ver\_num from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_RESP\_RQST\_STAT\_CD*** = "select resp\_rec\_stat\_cd from resp\_rec where resp\_rec\_id = %s";

**public** **static** **final** String ***GET\_RESP\_REC*** = "select count(\*) from resp\_rec where resp\_rec\_id = %s";

**public** **static** **final** String ***GET\_RESPONSE\_RECORD\_ID*** = "select resp\_rec\_id from resp\_rec where resp\_rec\_nam like '%s';";

**public** **static** **final** String ***GET\_PRIVACY\_RECORD\_STATUS*** = "select rqst\_stat\_cd from prvcy\_rqst where prvcy\_rqst\_id=%d";

**public** **static** **final** String ***GET\_PRODUCT\_PRIVACY\_RECORD\_STATUS*** = "select prdct\_rqst\_stat\_cd from prdct\_prvcy\_rqst where prdct\_cd='%s' and prdct\_prvcy\_rqst\_id=%d";

**public** **static** **final** String ***UPDATE\_SERVICE\_FUNCTION\_STATUS\_CODE*** = "update serv\_func\_prvcy\_rqst set serv\_func\_rqst\_stat\_cd='%s', prvcy\_resp\_pyld\_txt=null, resp\_rcv\_ts=null where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***UPDATE\_PRIVACY\_REQUEST\_STATUS\_CODE*** = "update prvcy\_rqst set rqst\_stat\_cd='%s' where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***UPDATE\_PRODUCT\_REQUEST\_STATUS\_CODE*** = "update prdct\_prvcy\_rqst set prdct\_rqst\_stat\_cd='%s' where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_DELETE\_REQUEST\_RESPONSE\_PAYLOAD*** = "select prvcy\_resp\_pyld\_txt from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_PRODUCT\_PRIVACY\_REQUEST\_ID\_FROM\_REQUEST\_ID*** = "select prdct\_prvcy\_rqst\_id from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %s and prdct\_cd = '%s';";

**public** **static** **final** String ***GET\_RESPONSE\_RECORD\_ID\_FOR\_A\_PRODUCT\_PRIVACY\_REQUEST\_ID*** = "select resp\_rec\_id from resp\_rec where prdct\_prvcy\_rqst\_id = %s and resp\_rec\_nam = '%s';";

**public** **static** **final** String ***EXPORT\_COMMENTS\_BY\_PRIVACY\_REQUEST\_ID*** = "select exprt\_cmnt\_txt from prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_AUDIT\_DESCRIPTION\_BY\_PRIVACY\_REQUEST\_ID*** = "select aud\_desc from aud\_hist where aud\_elmt\_key\_txt = '%s' and aud\_type\_cd='PRDCT\_EXPT'";

**public** **static** **final** String ***GET\_USERID\_BY\_PRIVACY\_REQUEST\_ID*** = "select aud\_user\_id from aud\_hist where aud\_elmt\_key\_txt = '%s' and aud\_type\_cd='PRDCT\_EXPT'";

**public** **static** **final** String ***PUBLISH\_COMMENTS\_BY\_PRIVACY\_REQUEST\_ID*** = "select pblsh\_cmnt\_txt from prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_SFC\_FROM\_PRODUCT\_PRIVACY\_REQUEST*** = "select serv\_func\_prvcy\_rqst\_id from serv\_func\_prvcy\_rqst where prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_RECORD\_ID\_FROM\_SFC\_AND\_PRODUCT\_PRIVACY\_REQUEST*** = "select resp\_rec\_id from resp\_rec where serv\_func\_prvcy\_rqst\_id = %s and prdct\_prvcy\_rqst\_id = %s and resp\_rec\_nam = '%s';";

**public** **static** **final** String ***UPDATE\_EXPORT\_COMMENTS*** = "update prdct\_prvcy\_rqst set exprt\_cmnt\_txt = '%s' where prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_PRODUCT\_PRIVACY\_REQUEST\_STATUS*** = "update prdct\_prvcy\_rqst set prdct\_ver\_stat\_cd = '%s' where prvcy\_rqst\_id = %s and prdct\_cd = '%s';";

**public** **static** **final** String ***GET\_LABEL\_TRNSLT\_REC\_BY\_LOCALE*** = "select \* from label\_trnslt where loc\_cd='%s' AND label\_cd IN (%s)";

**public** **static** **final** String ***GET\_DATA\_ELEMS\_FOR\_SERV\_FUNC*** = "select elmt\_tag\_txt from serv\_func\_data\_elmt where serv\_func\_cd='%s'";

**public** **static** **final** String ***GET\_SERV\_FUNC\_LEGAL\_NOTE*** = "select legal\_note\_cd from serv\_func\_def where serv\_func\_cd = '%s'";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTION\_STATUS\_CODE\_BY\_RECORD\_ID*** = "select serv\_func\_rqst\_stat\_cd from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id=(select del\_serv\_func\_rqst\_id from resp\_rec where resp\_rec\_id= %d);";

**public** **static** **final** String ***GET\_PRODUCT\_REQUEST\_STATUS\_CODE\_BY\_RECORD\_ID*** = "select prdct\_rqst\_stat\_cd from prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id=(select del\_prdct\_rqst\_id from resp\_rec where resp\_rec\_id= %d);";

**public** **static** **final** String ***GET\_PRODUCT\_VER\_TIMESTAMP*** = "select prdct\_ver\_ts FROM prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id=%s;";

**public** **static** **final** String ***GET\_RECORD\_FOR\_USER\_NOT\_LOGGED\_IN*** = "select prvcy\_rqst\_id from prvcy\_rqst where rqst\_email\_addr !='%s' limit 1";

**public** **static** **final** String ***SET\_FIRST\_LAST\_NAMES\_FOR\_REQUEST*** = "UPDATE prvcy\_rqst set first\_nam='%s', lst\_nam='%s' where prvcy\_rqst\_id=%s";

**public** **static** **final** String ***GET\_NON\_DISCLOSED\_RECORD\_COUNT\_PRODUCT\_PRIVACY\_REQUEST\_ID*** = "Select count(\*) from resp\_rec where prdct\_prvcy\_rqst\_id = %s AND dsclsr\_sw = %b";

**public** **static** **final** String ***GET\_UNDELETED\_DISCLOSED\_RECORD\_COUNT\_PRODUCT\_PRIVACY\_REQUEST\_ID*** = "Select count(\*) from resp\_rec where prdct\_prvcy\_rqst\_id = %s AND dsclsr\_sw = %b AND del\_serv\_func\_rqst\_id is null";

**public** **static** **final** String ***GET\_PRODUCT\_PUBLISH\_TIMESTAMP*** = "select pblsh\_ts from prdct\_prvcy\_rqst where prdct\_cd = '%s' AND prvcy\_rqst\_id = %d";

**public** **static** **final** String ***GET\_AUDIT\_RECORDS*** = "select \* from aud\_hist where aud\_type\_cd='%s' and aud\_elmt\_key\_txt=%s and aud\_desc ='%s'";

**public** **static** **final** String ***GET\_LABEL\_TRANSLATION\_RECORD*** = "Select \* from label\_trnslt where loc\_cd='%s' and label\_cd='%s'";

**public** **static** **final** String ***GET\_RESP\_RECORDS\_FOR\_PRODUCT\_PRIVACY\_REQUESTS*** = "select \* from resp\_rec where prdct\_prvcy\_rqst\_id in (select prdct\_prvcy\_rqst\_id from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %d);";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_DETAILS\_FOR\_REQUEST\_ID\_WITH\_PRODUCT\_NAME*** = "select \* from prdct\_prvcy\_rqst ppr join prdct\_def pd on pd.prdct\_cd = ppr.prdct\_cd where ppr.prvcy\_rqst\_id = %d;";

**public** **static** **final** String ***GET\_RESP\_REC\_NAMES*** = "select resp\_rec\_nam FROM resp\_rec where serv\_func\_prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_PRDCTS\_FOR\_REQUEST*** = "select \* From prdct\_prvcy\_rqst where prdct\_cd in (%s) and prvcy\_rqst\_id=%s";

**public** **static** **final** String ***GET\_PORTAL\_TYPE*** = "select portal\_type\_cd FROM prvcy\_rqst where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***DELETE\_RESPONSE\_COMMENTS*** = "Delete from resp\_cmnt where resp\_rec\_id in (select resp\_rec\_id from resp\_rec where serv\_func\_prvcy\_rqst\_id in (%s));";

**public** **static** **final** String ***DELETE\_RESPONSE\_RECORDS*** = "Delete from resp\_rec where serv\_func\_prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_VIEW\_REQUEST*** = "Delete from prvcy\_rqst where vw\_rqst\_id in (%s);";

**public** **static** **final** String ***UPDATE\_SERV\_FUNC\_RQST\_STAT\_CD\_BY\_SERV\_FUNC\_RQST\_ID*** = "update serv\_func\_prvcy\_rqst set serv\_func\_rqst\_stat\_cd = 'D' where serv\_func\_prvcy\_rqst\_id in (select del\_serv\_func\_rqst\_id from resp\_rec where resp\_rec\_id in(%s));";

**public** **static** **final** String ***GET\_SERV\_FUNC\_RQST\_STAT\_CD\_BY\_SERV\_FUNC\_RQST\_ID*** = "select serv\_func\_rqst\_stat\_cd from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_SERV\_FUNC\_RQST\_STAT\_CD\_BY\_PRIVACY\_RQST\_ID\_AND\_SERV\_FUNC\_CODE*** = "select serv\_func\_rqst\_stat\_cd from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = '%s' and serv\_func\_cd = '%s'";

**public** **static** **final** String ***GET\_TIME\_STAMP\_BY\_SERV\_FUNC\_RQST\_ID*** = "select rqst\_cmplt\_ts from resp\_rec where del\_serv\_func\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_TIME\_STAMP\_PRODUCT\_PRIVACY\_REQUEST\_TABLE*** = "select rqst\_cmplt\_ts from prdct\_prvcy\_rqst where prvcy\_rqst\_id='%s' and prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_PRODUCT\_STATUS\_PRODUCT\_PRIVACY\_REQUEST\_TABLE*** = "select prdct\_rqst\_stat\_cd from prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id='%s' and prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTION\_VERIFICATION\_QUESTIONS*** = "select \* from serv\_func\_ver\_qstn where serv\_func\_cd in \n"

+ "(select serv\_func\_cd from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id in \n"

+ "(select serv\_func\_prvcy\_rqst\_id from resp\_rec where prdct\_prvcy\_rqst\_id = %d))";

**public** **static** **final** String ***GET\_RESPONSE\_RECORDS\_FOR\_SERVICE\_FUNCTION\_VERIFICATION\_QUESTIONS*** = "select resp\_rec\_nam, resp\_data\_txt from resp\_rec where serv\_func\_prvcy\_rqst\_id in \n"

+ "(select serv\_func\_prvcy\_rqst\_id from serv\_func\_prvcy\_rqst where serv\_func\_cd in \n"

+ "(select serv\_func\_cd from serv\_func\_ver\_qstn where serv\_func\_cd in \n"

+ "(select serv\_func\_cd from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id in \n"

+ "(select serv\_func\_prvcy\_rqst\_id from resp\_rec where prdct\_prvcy\_rqst\_id = %d))))";

**public** **static** **final** String ***UPDATE\_REQUEST\_TYPE\_CODE\_BY\_PRIVACY\_REQUEST\_ID*** = "update prvcy\_rqst set rqst\_type\_cd='U' where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***UPDATE\_PORTAL\_TYPE\_CODE\_BY\_PRIVACY\_REQUEST\_ID*** = "update prvcy\_rqst set portal\_type\_cd = 'B2B' where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_REQUEST\_STATUS\_CODE\_BY\_PRIVACY\_REQUEST\_ID*** = "select rqst\_stat\_cd from prvcy\_rqst where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***UPDATE\_LABEL\_TRNSLT\_TABLE*** = "UPDATE label\_trnslt set label\_cd='%s' where label\_cd='%s'";

**public** **static** **final** String ***GET\_RESPONSE\_RECORDS\_FOR\_SERVICE\_FUNCTION\_PRIVACY\_REQUEST*** = "select resp\_data\_txt from resp\_rec where serv\_func\_prvcy\_rqst\_id = \n"

+ "(select serv\_func\_prvcy\_rqst\_id from serv\_func\_prvcy\_rqst where serv\_func\_cd = '%s')";

**public** **static** **final** String ***GET\_COUNT\_OF\_RESPONSE\_RECORDS\_BY\_DELETE\_PRIVACY\_REQUEST*** = "select count(\*) from resp\_rec where del\_prvcy\_rqst\_id = %d;";

**public** **static** **final** String ***GET\_RESPONSE\_RECORDS\_BY\_PRIVACY\_REQUEST\_ID\_AND\_SERV\_FUNC\_CODE*** = "select \* from gdp\_owner.resp\_rec where serv\_func\_prvcy\_rqst\_id in (select serv\_func\_prvcy\_rqst\_id from gdp\_owner.serv\_func\_prvcy\_rqst where serv\_func\_cd = '%s' and prvcy\_rqst\_id = %s);";

**public** **static** **final** String ***GET\_COUNT\_OF\_RESPONSE\_RECORDS\_BY\_ACTIVITY\_UID*** = "select count(\*) from gdp\_owner.resp\_rec where actvy\_uid = '%s' and serv\_func\_prvcy\_rqst\_id in (select serv\_func\_prvcy\_rqst\_id from gdp\_owner.serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s');";

**public** **static** **final** String ***CLEAN\_UP\_SERV\_FUNC\_INTEGRATION*** = "DELETE from serv\_func\_integration where serv\_func\_cd not in (select distinct serv\_func\_cd from serv\_func\_data\_elmt)";

**public** **static** **final** String ***GET\_RESPONSE\_IDS\_FOR\_PRIVACY\_REQUEST*** = "select resp\_rec\_id from resp\_rec where resp\_rec\_nam like '%s' and prdct\_prvcy\_rqst\_id in (select prdct\_prvcy\_rqst\_id from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %s);";

**public** **static** **final** String ***GET\_CONTEXT\_CODE\_BY\_REQUEST\_ID*** = "select rqst\_cntxt\_cd from prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_REJECT\_REASON\_CODE*** = "select rej\_rsn\_cd from prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_REJECT\_COMMENT\_TEXT*** = "select rej\_cmnt\_txt from prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_REVIEW\_ATTORNEY\_FIRST\_NAME*** = "select rvw\_attorney\_first\_nam from prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_REVIEW\_ATTORNEY\_LAST\_NAME*** = "select rvw\_attorney\_lst\_nam from prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_REVIEW\_TIMESTAMP*** = "select rvw\_ts from prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***UPDATE\_PORTAL\_TYPE\_CODE\_AND\_PRIVACY\_REQUEST\_ID*** = "update prvcy\_rqst set portal\_type\_cd = '%s' where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***DELETE\_EMAIL\_VER\_BY\_PRIVACY\_REQUEST\_ID*** = "delete from email\_ver where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_CARD\_VER\_BY\_PRIVACY\_REQUEST\_ID*** = "delete from card\_ver where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***GET\_RESPONE\_RECORD\_COMMENT\_COUNT*** = "select count(\*) from resp\_cmnt where resp\_rec\_id = %d AND crte\_ts > '%s';";

**public** **static** **final** String ***UPDATE\_TIMESTAMP\_AND\_STATUS\_CODE\_BY\_PRIVACY\_REQUEST\_ID*** = "update prvcy\_rqst set rqst\_stat\_cd='C', rvw\_ts=(SELECT CURRENT\_TIMESTAMP::timestamp without time zone), rvw\_attorney\_first\_nam='Sally', rvw\_attorney\_lst\_nam='Doe' where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_RECORD\_ACTION\_STATUS\_BY\_ID*** = "select \* from rec\_actn\_stat where resp\_rec\_id = %s;";

**public** **static** **final** String ***GET\_PRODUCT\_ACTION\_STATUS\_ID*** = "select prdct\_actn\_stat\_id from rec\_actn\_stat where resp\_rec\_id = %s;";

**public** **static** **final** String ***GET\_PRODUCT\_ACTION\_STATUS\_CODE*** = "select prdct\_rqst\_stat\_cd from prdct\_actn\_stat where prdct\_actn\_stat\_id = %s;";

**public** **static** **final** String ***GET\_PRDCT\_ACTION\_RQST\_STATUS*** = "select prdct\_rqst\_stat\_cd from prdct\_actn\_stat where prdct\_prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_RESP\_REC\_ACTION\_STATUS*** = "select rec\_stat\_cd from rec\_actn\_stat where resp\_rec\_id = %s;";

**public** **static** **final** String ***GET\_TIMESTAMP\_FROM\_RESP\_REC\_ACTION\_STATUS*** = "select rvw\_ts from rec\_actn\_stat where resp\_rec\_id = %s;";

**public** **static** **final** String ***UPDATE\_TIMESTAMP\_AND\_RECORD\_STATUS\_CODE\_AND\_REJECTION\_REASON\_CODE\_BY\_RECORD\_ID*** = "update rec\_actn\_stat set rvw\_ts='%s', rec\_stat\_cd='%s', rej\_rsn\_cd='%s' , rvw\_user\_nam='%s' where resp\_rec\_id= %s;";

**public** **static** **final** String ***UPDATE\_TIMESTAMP\_AND\_RECORD\_STATUS\_CODE\_BY\_RECORD\_ID*** = "update rec\_actn\_stat set rvw\_ts='%s', rec\_stat\_cd='%s', rvw\_user\_nam='%s' where resp\_rec\_id= %s;";

**public** **static** **final** String ***UPDATE\_TIMESTAMP\_AND\_PRODUCT\_REQUEST\_STATUS\_CODE\_BY\_RECORD\_ID*** = "update prdct\_actn\_stat set rvw\_ts='%s', prdct\_rqst\_stat\_cd='%s' where prdct\_actn\_stat\_id=(select prdct\_actn\_stat\_id from rec\_actn\_stat where resp\_rec\_id='%s');";

**public** **static** **final** String ***DELETE\_RECORD\_ACTN\_STAT*** = "Delete from rec\_actn\_stat where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_PRDCT\_ACTN\_STAT*** = "Delete from prdct\_actn\_stat where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_STATUS*** = "select rqst\_stat\_cd from prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_AUDIT\_RECORD\_BY\_ID*** = "select \* from aud\_hist where aud\_elmt\_key\_txt=%s order by aud\_hist\_id desc;";

**public** **static** **final** String ***GET\_RESP\_RECORDS\_FOR\_RECORD\_ACTION\_STATUS*** = "select \* from resp\_rec where resp\_rec\_id in (select resp\_rec\_id from rec\_actn\_stat where prvcy\_rqst\_id = %d);";

**public** **static** **final** String ***GET\_RESPONSE\_ID\_BY\_NAME*** = "select resp\_rec\_id from resp\_rec where resp\_rec\_nam = '%s' order by resp\_rec\_id desc;";

**public** **static** **final** String ***UPDATE\_PRIVACY\_REQUEST\_USER\_DETAILS*** = "update prvcy\_rqst set rqst\_email\_addr = '%s', crte\_user\_id = '%s', lst\_updt\_user\_id = '%s' where prvcy\_rqst\_id = %d;";

**public** **static** **final** String ***GET\_RESPONSE\_RECORDS\_BY\_RESP\_REC\_IDS*** = "SELECT \* from resp\_rec where resp\_rec\_id in (%s)";

**public** **static** **final** String ***GET\_REQUEST\_TYPE\_PRIVACY\_REQUEST\_TABLE*** = "select rqst\_type\_cd from prvcy\_rqst where prvcy\_rqst\_id = '%s'";

**public** **static** **final** String ***UPDATE\_PRIVACY\_REQUEST\_DETAILS\_FOR\_B2C*** = "update prvcy\_rqst set rqst\_email\_addr = '%s',crte\_user\_id = '%s' where crte\_user\_id = '%s' AND lst\_updt\_user\_id = '%s' AND first\_nam = '%s' AND lst\_nam = '%s';";

**public** **static** **final** String ***UPDATE\_PRIVACY\_REQUEST\_DETAILS\_FOR\_B2C\_UPDATED*** = "update prvcy\_rqst set rqst\_email\_addr = '%s', crte\_user\_id = '%s' where crte\_user\_id = '%s';";

**public** **static** **final** String ***UPDATE\_DATA\_ELEMENT\_MASK\_ENBL\_SW\_ENCRYPT\_SW*** = "update data\_elmt set encrypt\_sw = '%s', mask\_enbl\_sw = '%s' where elmt\_tag\_txt = '%s';";

**public** **static** **final** String ***REMOVE\_ERRONEOUS\_ENTRIES\_FROM\_INTEGRATION\_TABLE*** = "delete from serv\_func\_integration where serv\_func\_cd in (select s.serv\_func\_cd from serv\_func\_integration s group by s.serv\_func\_cd, s.gdp\_func\_cd having count(\*) > 1);";

**public** **static** **final** String ***GET\_SERVVICE\_FUNCTION\_INTERGRATION\_EMAIL*** = "select serv\_integration\_email\_addr from serv\_func\_integration where serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_AUDIT\_RECORD\_BY\_ID\_AND\_AUD\_TYPE\_CD*** = "select \* from gdp\_owner.aud\_hist where aud\_elmt\_key\_txt = '%s' and aud\_type\_cd = '%s' order by aud\_hist\_id desc;";

**public** **static** **final** String ***GET\_PRODUCT\_REQUEST\_STATUS\_CODE\_FROM\_REQUEST\_ID*** = "select prdct\_rqst\_stat\_cd from prdct\_prvcy\_rqst where prvcy\_rqst\_id = %s and prdct\_cd = '%s';";

**public** **static** **final** String ***UPDATE\_ALL\_AP\_PRIVACY\_REQUEST\_STATUS\_CODES*** = "update prvcy\_rqst set rqst\_stat\_cd='%s' where rqst\_stat\_cd = '%s'";

**public** **static** **final** String ***DELETE\_AUDIT\_RECORD\_BY\_ID*** = "delete from aud\_hist where aud\_elmt\_key\_txt in (%s);";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_FOR\_A\_CNTXT\_AND\_PRTL\_TYPE*** = "select \* from prvcy\_rqst where rqst\_type\_cd='%s' and rqst\_cntxt\_cd='%s' and portal\_type\_cd='%s' order by prvcy\_rqst\_id desc limit 1";

**public** **static** **final** String ***GET\_DELETE\_SERVICE\_FUNCTION\_REQUEST\_ID*** = "select del\_serv\_func\_rqst\_id from resp\_rec where resp\_rec\_nam = '%s';";

**public** **static** **final** String ***UPDATE\_PRODUCT\_PRIVACY\_REQUEST\_STATUS\_DELETE\_ALL*** = "update prdct\_prvcy\_rqst set pblsh\_ts = (SELECT CURRENT\_TIMESTAMP::timestamp without time zone), \n"

+ "pblsh\_user\_id = 'gdpsuperuser', pblsh\_user\_nam = 'GDP Super User', prdct\_rqst\_stat\_cd = '%s' where prvcy\_rqst\_id = %d AND prdct\_cd = '%s';";

**public** **static** **final** String ***GET\_COUNT\_OF\_SERVICE\_FUNCTION\_PRIVACY\_REQUESTS\_DELETE\_ALL*** = "select count(\*) from serv\_func\_prvcy\_rqst where serv\_func\_cd = '%s' and prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_FIELD\_HIDE\_SW*** = "update gdp\_owner.data\_elmt SET hide\_sw=%s where elmt\_tag\_txt='%s'";

**public** **static** **final** String ***UPDATE\_FIELD\_ENCRYPT\_SW*** = "update gdp\_owner.data\_elmt SET encrypt\_sw=%s where elmt\_tag\_txt='%s'";

**public** **static** **final** String ***GET\_RESPONSE\_RECEIVED\_TIMESTAMP*** = "Select resp\_rcv\_ts from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id IN(Select del\_serv\_func\_rqst\_id from resp\_rec where resp\_rec\_id='%s');";

**public** **static** **final** String ***GET\_REVIEWED\_TIMESTAMP\_RECORD\_ACTION\_TABLE*** = "Select rvw\_ts from rec\_actn\_stat where resp\_rec\_id='%s';";

**public** **static** **final** String ***GET\_PRODUCT\_PRIVACY\_REQUEST\_TIME\_STAMP*** = "select rqst\_cmplt\_ts from prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***DELETE\_AUDIT\_BY\_AUDIT\_TYPE\_CODE*** = "delete from aud\_hist where aud\_type\_cd = '%s';";

**public** **static** **final** String ***GET\_RESP\_DATA\_TXT*** = "select resp\_data\_txt from resp\_rec where del\_serv\_func\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_PRVCY\_FILES\_ID*** = "SELECT PRVCY\_RESP\_FILE\_ID FROM ACTVY\_REC WHERE ACTVY\_DATA\_UID\_ID = '%s';";

**public** **static** **final** String ***GET\_RESP\_DATA\_FULL\_TXT*** = "select resp\_data\_full\_txt from resp\_rec where del\_serv\_func\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_PRVCY\_RESP\_PYLD\_TXT*** = "select prvcy\_resp\_pyld\_txt from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = (select serv\_func\_prvcy\_rqst\_id from resp\_rec where del\_serv\_func\_rqst\_id = '%s');";

**public** **static** **final** String ***GET\_AUDIT\_RECORDS\_WITH\_AUDIT\_TYPE\_CODE*** = "select \* from aud\_hist where aud\_type\_cd = '%s'";

**public** **static** **final** String ***GET\_QSTN\_ANSWR\_TXT*** = "select qstn\_answr\_txt from cust\_id\_ver where prdct\_prvcy\_rqst\_id = '%s'";

**public** **static** **final** String ***GET\_RESP\_RCV\_TS*** = "select resp\_rcv\_ts from serv\_func\_prvcy\_rqst where serv\_func\_prvcy\_rqst\_id = '%s'";

**public** **static** **final** String ***UPDATE\_FIELD\_HIDE\_SW\_FOR\_ALL\_FIELDS*** = "update gdp\_owner.data\_elmt SET hide\_sw=%s where 1=1";

**public** **static** **final** String ***TRUNCATE\_SERVICE\_FUNCTION\_PRIVACY\_REQUEST*** = "truncate table gdp\_owner.serv\_func\_prvcy\_rqst cascade;";

**public** **static** **final** String ***TRUNCATE\_SERVICE\_FUNCTION\_INTEGRATION*** = "truncate table gdp\_owner.serv\_func\_integration cascade;";

**public** **static** **final** String ***TRUNCATE\_PRIVACY\_REQUEST*** = "truncate table gdp\_owner.prvcy\_rqst cascade;";

**public** **static** **final** String ***TRUNCATE\_AGENT\_PRIVACY\_REQUEST*** = "truncate table gdp\_owner.agent\_prvcy\_rqst cascade;";

**public** **static** **final** String ***TRUNCATE\_PRODUCT\_DEF*** = "truncate table gdp\_owner.prdct\_def cascade;";

**public** **static** **final** String ***TRUNCATE\_PRIVACY\_REQUEST\_ACTION*** = "truncate table gdp\_owner.prvcy\_rqst\_actn cascade;";

**public** **static** **final** String ***PRIVACY\_REQUEST\_COUNT*** = "select count(\*) from gdp\_owner.prvcy\_rqst;";

**public** **static** **final** String ***SERVICE\_FUNCTION\_PRIVACY\_COUNT*** = "select count(\*) from gdp\_owner.serv\_func\_prvcy\_rqst;";

**public** **static** **final** String ***REMOVE\_DUPLICATES\_FROM\_LABEL\_TRANSLT*** = "delete from gdp\_owner.label\_trnslt where label\_trnslt\_id not in (select min(label\_trnslt\_id) from gdp\_owner.label\_trnslt group by label\_cd, loc\_cd) ";

**public** **static** **final** String ***CHUNK\_DOCUMENT\_COUNT*** = "Select count(\*) FROM gdp\_owner.DOC\_CNTN\_CHUNK where doc\_upld\_id in (Select doc\_upld\_id FROM gdp\_owner.DOC\_UPLD where doc\_file\_nam = '%s' and doc\_upld\_id in (Select doc\_upld\_id FROM gdp\_owner.ID\_PROOF\_DOC\_UPLD where prvcy\_rqst\_id\_proof\_id in ( Select prvcy\_rqst\_id\_proof\_id FROM gdp\_owner.PRVCY\_RQST\_ID\_PROOF where prvcy\_rqst\_id = '%s')))";

**public** **static** **final** String ***DOC\_UPLOAD*** = "Select \* FROM gdp\_owner.DOC\_UPLD where doc\_file\_nam = '%s' and doc\_upld\_id in (Select doc\_upld\_id FROM gdp\_owner.ID\_PROOF\_DOC\_UPLD where prvcy\_rqst\_id\_proof\_id in ( Select prvcy\_rqst\_id\_proof\_id FROM gdp\_owner.PRVCY\_RQST\_ID\_PROOF where prvcy\_rqst\_id = '%s'))";

**public** **static** **final** String ***GET\_LATEST\_MESSAGE\_TO\_REQUESTER*** = "select extrl\_cmnt\_txt from gdp\_owner.prvcy\_rqst\_actn where extrl\_cmnt\_txt IS NOT NULL and prvcy\_rqst\_id='%s' and prvcy\_rqst\_actn\_cd='C' order by crte\_ts desc limit 1";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_ACTIONS\_FOR\_GIVEN\_REQUEST\_ID*** = "select \* from prvcy\_rqst\_actn where prvcy\_rqst\_id = %s order by prvcy\_rqst\_actn\_id desc; ";

**public** **static** **final** String ***GET\_DECLINE\_REASON\_LOOKUP*** = "select \* from gdp\_owner.id\_proof\_dcln\_rsn\_lkp";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_IDENTITY\_PRROF\_ID*** = "SELECT %s FROM %s WHERE prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_ALL\_RECORD\_WITH\_REQUEST\_ID*** = "SELECT \* FROM %s WHERE prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_AUDIT\_HISTORY\_ELEMENT\_KEY*** = "SELECT \* FROM %s WHERE aud\_elmt\_key\_txt='%s' and aud\_type\_cd ='%s'";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_DOCUMENT\_UPLOADS\_FOR\_GIVEN\_REQUEST\_ID*** = "select \* from prvcy\_rqst\_doc\_upld where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***REMOVE\_PRIVACY\_REQUEST\_ACTIONS\_FOR\_A\_REQUEST\_ID*** = "delete from gdp\_owner.prvcy\_rqst\_actn where prvcy\_rqst\_id = %d;";

**public** **static** **final** String ***GET\_AUTO\_COMPLETE\_FLAG\_FOR\_PRVCY\_RQST*** = "select auto\_cmplt\_sw from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***UPDATE\_DOCUMENT\_UPLOAD\_COMPLETE\_SWITCH*** = "update gdp\_owner.prvcy\_rqst set doc\_upld\_cmplt\_sw = %b where prvcy\_rqst\_id = '%s'";

**public** **static** **final** String ***DELETE\_PRIVACY\_REQUEST\_ACTN\_FOR\_DELETE\_AND\_UPDATE*** = "delete from gdp\_owner.prvcy\_rqst\_actn where prvcy\_rqst\_id in (select prvcy\_rqst\_id from gdp\_owner.prvcy\_rqst where vw\_rqst\_id in (%s));";

**public** **static** **final** String ***GET\_DOCUMENT\_UPLOAD\_COMPLETE\_FLAG\_FOR\_PRVCY\_RQST*** = "select doc\_upld\_cmplt\_sw from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***DECLINE\_POI\_DOCUMENT*** = "update gdp\_owner.prvcy\_rqst\_id\_proof set appr\_sw=%b, rvw\_user\_id='gdpsuperuser', rvw\_user\_nam = 'GDP Super User', rvw\_ts = (SELECT CURRENT\_TIMESTAMP::timestamp without time zone), dcln\_rsn\_cd = '%s', dcln\_msg\_txt = '%s' where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***APPROVE\_POI\_DOCUMENT*** = "update gdp\_owner.prvcy\_rqst\_id\_proof set appr\_sw=%b, rvw\_user\_id='gdpsuperuser', rvw\_user\_nam = 'GDP Super User', rvw\_ts = (SELECT CURRENT\_TIMESTAMP::timestamp without time zone) where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_DOCUMENT\_APPROVAL\_STATUS\_PRIVACY\_REQUEST*** = "update gdp\_owner.prvcy\_rqst set id\_proof\_ver\_sw = %b where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***FILTER\_CRITERIA*** = "select p.prvcy\_rqst\_id, p.rqst\_type\_cd, p.rqst\_stat\_cd, p.first\_nam, p.lst\_nam, p.rqst\_cntxt\_cd, p.crte\_ts, p.vw\_rqst\_id, p.days\_in\_queue\_num, d.cntry\_cd, s.st\_cd, c.regn\_cd \n"

+ "from gdp\_owner.prvcy\_rqst p \n" + "left outer join gdp\_owner.data\_prvcy\_law\_assgn d \n"

+ "on p.data\_prvcy\_law\_assgn\_id = d.data\_prvcy\_law\_assgn\_id \n" + "left outer join gdp\_owner.st s \n"

+ "on d.st\_id = s.st\_id\n" + "left outer join gdp\_owner.cntry\_phn\_cd c\n"

+ "on d.cntry\_cd = c.cntry\_phn\_cd where %s order by %s %s, lst\_updt\_ts desc limit %s;";

**public** **static** **final** String ***GET\_NON\_PBLSHD\_PRIVACY\_REQUEST\_DOCUMENT\_UPLOAD\_RECORDS*** = "select \* from prvcy\_rqst\_doc\_upld where prvcy\_rqst\_doc\_upld\_id in (%s) and pblsh\_ts is null";

**public** **static** **final** String ***GET\_DOCUMENT\_UPLOAD\_RECORDS*** = "select \* from doc\_upld where doc\_upld\_id in (%s)";

**public** **static** **final** String ***GET\_DOCUMENT\_CNTN\_CHUNK\_RECORDS*** = "select \* from doc\_cntn\_chunk where doc\_upld\_id in (%s)";

**public** **static** **final** String ***SELECT\_FILE\_NAMES\_FROM\_DOC\_UPLD*** = "Select doc\_file\_nam from doc\_upld where doc\_upld\_id in (%s)";

**public** **static** **final** String ***GET\_COUNT\_OF\_ID\_VERIFICATION\_REQUESTS*** = "SELECT count(\*) FROM prvcy\_rqst WHERE rqst\_stat\_cd = '%s';";

**public** **static** **final** String ***ORDER\_BY\_PROOF\_ID*** = "order by prvcy\_rqst\_id\_proof\_id desc";

**public** **static** **final** String ***ORDER\_BY\_AUDT\_TYPE\_CD*** = "order by aud\_type\_cd asc";

**public** **static** **final** String ***ORDER\_BY\_AUDT\_TS*** = "order by aud\_ts desc";

**public** **static** **final** String ***GET\_VERIFIED\_SWITCH\_POI*** = "SELECT ID\_PROOF\_VER\_SW FROM %s WHERE prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_ATTEMPTS\_COUNT*** = "select id\_proof\_atmpt\_num from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_COUNT*** = "select count(\*) from gdp\_owner.id\_proof\_doc\_upld where prvcy\_rqst\_id\_proof\_id='%s';";

**public** **static** **final** String ***GET\_VIEW\_REQUEST\_ID\_FOR\_A\_REQUEST\_ID*** = "select vw\_rqst\_id from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_DELETE\_PRVCY\_RQSTID\_FROM\_PRVCY\_RQST\_DOC\_UPLD*** = "select del\_prvcy\_rqst\_id from gdp\_owner.prvcy\_rqst\_doc\_upld where prvcy\_rqst\_doc\_upld\_id='%s'";

**public** **static** **final** String ***GET\_DATA\_FROM\_DOC\_ACTN\_STATUS\_BY\_DELETE\_PRVCY\_RQST\_ID\_PRVCY\_RQST\_DOC\_UPLD\_ID*** = "select \* from gdp\_owner.doc\_actn\_stat where prvcy\_rqst\_doc\_upld\_id=%d and prvcy\_rqst\_id=%d;";

**public** **static** **final** String ***GET\_TYPE\_DESCRIPTION\_FROM\_LKP\_TABLE*** = "select type\_nam from %s where type\_cd='%s';";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_DOCUMENT\_UPLOAD\_RECORDS\_BY\_PRVCY\_RQST\_DOC\_UPLD\_ID*** = "select \* from prvcy\_rqst\_doc\_upld where prvcy\_rqst\_doc\_upld\_id in (%s)";

**public** **static** **final** String ***UPDATE\_DOC\_ACTN\_STATUS\_CODE\_FOR\_DOCUMENT\_ID*** = "update gdp\_owner.doc\_actn\_stat set doc\_actn\_stat\_cd='%s' where prvcy\_rqst\_id=%s and prvcy\_rqst\_doc\_upld\_id=%s;";

**public** **static** **final** String ***UPDATE\_DOC\_ACTN\_STAT\_SW*** = "Update prvcy\_rqst set doc\_actn\_cmplt\_sw=%s where prvcy\_rqst\_id=%s";

**public** **static** **final** String ***MAKE\_DEL\_RQST\_WITH\_DOCS\_ONLY\_AS\_RVW\_COMPLET*** = "update gdp\_owner.prvcy\_rqst set rqst\_stat\_cd='AP', rvw\_ts=(SELECT CURRENT\_TIMESTAMP::timestamp without time zone) where prvcy\_Rqst\_id=%s";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_DOCUMENT\_UPLOAD\_FOR\_REQUEST\_ID\_AND\_DOCUMENT\_NAME*** = "select prvcy\_rqst\_doc\_upld\_id from gdp\_owner.prvcy\_rqst\_doc\_upld where del\_prvcy\_rqst\_id = %s and doc\_upld\_id in (select doc\_upld\_id from gdp\_owner.doc\_upld where doc\_file\_nam = '%s');";

**public** **static** **final** String ***GET\_DOCUMENT\_COMMENT\_COUNT*** = "select count(\*) from gdp\_owner.doc\_cmnt where prvcy\_rqst\_doc\_upld\_id = %s AND crte\_ts > '%s';";

**public** **static** **final** String ***GET\_DOCUMENT\_CMNT\_RECS*** = "select \* from doc\_cmnt where prvcy\_rqst\_doc\_upld\_id = %s order by crte\_ts desc";

**public** **static** **final** String ***GET\_DOCUMENT\_ACTION\_STATUS\_BY\_ID*** = "select \* from doc\_actn\_stat where prvcy\_rqst\_doc\_upld\_id = %s;";

**public** **static** **final** String ***GET\_DOC\_ACTN\_STAT\_RECORDS\_BY\_PRVCY\_RQST\_ID*** = "select \* from doc\_actn\_stat where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_CNTNT\_CHUNK\_RECS*** = "select \* from doc\_cntn\_chunk where doc\_upld\_id in (select doc\_upld\_id from prvcy\_rqst\_doc\_upld where prvcy\_rqst\_doc\_upld\_id in \r\n"

+ " (select prvcy\_rqst\_doc\_upld\_id from doc\_actn\_stat where prvcy\_rqst\_id=%s and doc\_actn\_stat\_cd in ('D', 'ND')))";

**public** **static** **final** String ***GET\_DOCUMENT\_UPLD\_ID\_FILE\_NAM\_FOR\_PRVCY\_RQST*** = "select du.doc\_upld\_id, du.doc\_file\_nam from doc\_upld du, prvcy\_rqst\_doc\_upld prdu where prdu.doc\_upld\_id = du.doc\_upld\_id and prdu.prvcy\_rqst\_id=%s";

**public** **static** **final** String ***GET\_DOCUMENT\_UPLD\_RECS\_WITH\_D\_ND\_STATUS*** = "select \* from doc\_upld where doc\_upld\_id in ( \r\n"

+ " select doc\_upld\_id from prvcy\_rqst\_doc\_upld \r\n"

+ " where prvcy\_rqst\_doc\_upld\_id in (select prvcy\_rqst\_doc\_upld\_id from doc\_actn\_stat \r\n"

+ " where prvcy\_rqst\_id=%s and doc\_actn\_stat\_cd in ('D', 'ND')))";

**public** **static** **final** String ***GET\_DOCUMENT\_UPLOAD\_COMPLETE\_SWITCH*** = "SELECT doc\_actn\_cmplt\_sw from prvcy\_rqst WHERE prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_ACTN\_FOR\_PRVCY\_RQST\_DOCUMENT\_STATUS*** = "select \* from prvcy\_rqst\_actn where prvcy\_rqst\_id = '%s' and intrl\_cmnt\_txt='%s'";

**public** **static** **final** String ***UPDATE\_PUBLISH\_TIMESTAMP\_FOR\_POI\_REQUIRED*** = "update gdp\_owner.prvcy\_rqst\_doc\_upld set pblsh\_ts = (SELECT CURRENT\_TIMESTAMP::timestamp without time zone) where id\_proof\_reqd\_sw = true and pblsh\_ts is null;";

**public** **static** **final** String ***GET\_DAYS\_IN\_QUEUE\_BY\_REQUESTID*** = "select days\_in\_queue\_num from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_REVIEW\_TIMESTAMP\_PRIVACY\_REQUEST\_ID\_PROOF*** = "select rvw\_ts from gdp\_owner.prvcy\_rqst\_id\_proof where prvcy\_rqst\_id = %s order by crte\_ts asc;";

**public** **static** **final** String ***GET\_POI\_UPLOAD\_DATE*** = "select crte\_ts from gdp\_owner.prvcy\_rqst\_id\_proof where prvcy\_rqst\_id = %s order by crte\_ts asc;";

**public** **static** **final** String ***GET\_PUBLISHED\_DATE\_FOR\_POI\_REQUIRED\_DOCUMENT*** = "select pblsh\_ts from gdp\_owner.prvcy\_rqst\_doc\_upld where prvcy\_rqst\_id = %s AND id\_proof\_reqd\_sw = true order by pblsh\_ts asc;";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_CREATED\_DATE*** = "select crte\_ts from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_COMPLETE\_DATE*** = "select rqst\_cmplt\_ts from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_FROM\_PRVCY\_RQST\_DOC\_UPLD\_BY\_REQ\_ID*** = "SELECT \* from gdp\_owner.prvcy\_rqst\_doc\_upld WHERE del\_prvcy\_rqst\_id=%s";

**public** **static** **final** String ***INSERT\_INTO\_EMAIL\_NOTIFY\_TABLE\_FOR\_PENDING*** = "insert into gdp\_owner.email\_notif\_log (msg\_body\_txt, sent\_stat\_cd,msg\_subj\_txt,to\_email\_addr,from\_email\_addr,crte\_ts) values('%s', 'P','','','',(SELECT CURRENT\_TIMESTAMP::timestamp without time zone))";

**public** **static** **final** String ***GET\_CHILD\_RECORD\_COUNT*** = "select count(\*) from gdp\_owner.serv\_func\_prvcy\_rqst where prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_CARD\_VERIFICATION\_DETAILS*** = "select \* from gdp\_owner.card\_ver where prvcy\_rqst\_id = %s and card\_num ='%s';";

**public** **static** **final** String ***GET\_CARD\_VERIFICATION\_DETAILS\_ONLY\_WITH\_CARD*** = "select \* from gdp\_owner.card\_ver where card\_num ='%s';";

**public** **static** **final** String ***GET\_COUNT\_OF\_CARD\_VERIFICATION\_ENTRIES*** = "SELECT count(\*) FROM gdp\_owner.card\_ver WHERE prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***FIRST\_FILTER\_ALL*** = "SELECT \* FROM gdp\_owner.prvcy\_rqst p, gdp\_owner.data\_prvcy\_law\_assign d WHERE ((d.cntry\_cd IN(select type\_cd from gdp\_owner.cntry\_phn\_cd\_lkp where regn\_cd = (select pref\_regn from gdp\_owner.user\_pref where user\_id = '%s')))AND (d.data\_prvcy\_law\_assign\_id = p.data\_prvcy\_law\_assgn\_id)) order by p.prvcy\_rqst\_id desc limit %s;";

**public** **static** **final** String ***FIRST\_FILTER\_PENDING\_REQUESTS*** = "SELECT \* FROM gdp\_owner.prvcy\_rqst p, gdp\_owner.data\_prvcy\_law\_assign d WHERE ((d.cntry\_cd IN(select type\_cd from gdp\_owner.cntry\_phn\_cd\_lkp where regn\_cd = (select pref\_regn from gdp\_owner.user\_pref where user\_id = '%s')))AND (d.data\_prvcy\_law\_assign\_id = p.data\_prvcy\_law\_assgn\_id)) AND p.rqst\_stat\_cd NOT IN ('DL', 'C') order by p.prvcy\_rqst\_id desc limit %s;";

**public** **static** **final** String ***FIRST\_FILTER\_ALL\_DEFAULT*** = "select \* from gdp\_owner.prvcy\_rqst order by prvcy\_rqst\_id desc limit %s;";

**public** **static** **final** String ***FIRST\_FILTER\_PENDING\_REQUESTS\_DEFAULT*** = "select \* from gdp\_owner.prvcy\_rqst where rqst\_stat\_cd not in ('C','DL') order by prvcy\_rqst\_id desc limit %s;";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_WITH\_NULL\_DAYS\_REMAINING*** = "select count(\*) from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s and days\_in\_queue\_num ISNULL;";

**public** **static** **final** String ***GET\_FULLFILLMENT\_DAYS\_BY\_REQUESTID*** = "select dplc.data\_prvcy\_law\_flfl\_days\_num from gdp\_owner.prvcy\_rqst pd join gdp\_owner.data\_prvcy\_law\_assgn dpla on (dpla.data\_prvcy\_law\_assgn\_id = pd.data\_prvcy\_law\_assgn\_id) join gdp\_owner.data\_prvcy\_law\_config dplc on (dpla.data\_prvcy\_law\_cd = dplc.data\_prvcy\_law\_cd and dplc.data\_acs\_key\_cd = 'RF' and dplc.rqst\_type\_cd = '%s') where pd.prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_ON\_BEHALF\_FIRST\_NAME\_FOR\_PRVCY\_RQST*** = "select rqstr\_first\_nam from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_ON\_BEHALF\_LAST\_NAME\_FOR\_PRVCY\_RQST*** = "select rqstr\_lst\_nam from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_REQUEST\_SEND\_TIMESTAMP*** = "select rqst\_send\_ts from gdp\_owner.serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_DOC\_CHUNK\_IDS*** = "SELECT doc.id\_proof\_doc\_upld\_id,chunk.doc\_cntn\_chunk\_id,doc1.doc\_file\_nam from gdp\_owner.doc\_cntn\_chunk chunk inner join gdp\_owner.DOC\_UPLD doc1 on chunk.doc\_upld\_id=doc1.doc\_upld\_id inner join gdp\_owner.id\_proof\_doc\_upld doc "

+ "on chunk.doc\_upld\_id=doc.doc\_upld\_id where doc.prvcy\_rqst\_id\_proof\_id in (SELECT prvcy\_rqst\_id\_proof\_id FROM gdp\_owner.prvcy\_rqst\_id\_proof where prvcy\_rqst\_id='%s') and doc1.doc\_file\_nam ='%s' group by doc.id\_proof\_doc\_upld\_id,chunk.doc\_cntn\_chunk\_id,doc1.doc\_file\_nam; ";

**public** **static** **final** String ***DELETE\_AGENT\_PRVCY\_RQST\_BY\_ID*** = "Delete from agent\_prvcy\_rqst where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***GET\_PRODCT\_PRVCY\_RQST\_ENTRY\_COUNT\_TEST\_USER*** = "Select count(\*) from gdp\_owner.prdct\_prvcy\_rqst WHERE prvcy\_rqst\_id='%s' AND prdct\_cd in (Select prdct\_cd from gdp\_owner.serv\_def WHERE serv\_cd in (Select serv\_cd from gdp\_owner.serv\_func\_def WHERE busn\_cntxt\_cd='%s' AND tst\_serv\_func\_sw='%s') );";

**public** **static** **final** String ***GET\_SERV\_FUN\_PRVCY\_RQST\_ENTRY\_COUNT\_TEST\_USER*** = "Select count(\*) from gdp\_owner.serv\_func\_prvcy\_rqst WHERE prvcy\_rqst\_id='%s' AND serv\_func\_cd in (Select serv\_func\_cd from gdp\_owner.serv\_func\_def WHERE busn\_cntxt\_cd='%s' AND tst\_serv\_func\_sw='%s');";

**public** **static** **final** String ***GET\_PRODCT\_PRVCY\_RQST\_ENTRY\_COUNT*** = "Select count(\*) from gdp\_owner.prdct\_prvcy\_rqst WHERE prvcy\_rqst\_id='%s' AND prdct\_cd in (Select prdct\_cd from gdp\_owner.serv\_def WHERE serv\_cd in (Select serv\_cd from gdp\_owner.serv\_func\_def WHERE busn\_cntxt\_cd='%s' AND decommission\_sw='%s') );";

**public** **static** **final** String ***GET\_SERV\_FUN\_PRVCY\_RQST\_ENTRY\_COUNT*** = "Select count(\*) from gdp\_owner.serv\_func\_prvcy\_rqst WHERE prvcy\_rqst\_id='%s' AND serv\_func\_cd in (Select serv\_func\_cd from gdp\_owner.serv\_func\_def WHERE busn\_cntxt\_cd='%s' AND decommission\_sw='%s');";

**public** **static** **final** String ***GET\_STORED\_PROCEDURE\_SERVICE\_FUNCTION\_DECOMMISSION*** = "select gdp\_owner.DecommissionServiceFunction('%s', '%s', %s);";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTION\_DECOMMISSION\_AUDIT\_ENTRY*** = "select \* from gdp\_owner.serv\_func\_decmsn\_aud where serv\_func\_cd='%s' order by serv\_func\_decmsn\_aud\_id desc;";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTION\_DECOMMISSION\_FLAG*** = "select decommission\_sw from gdp\_owner.serv\_func\_def where serv\_func\_cd='%s';";

**public** **static** **final** String ***GET\_PURGE\_TIME\_FROM\_PRVCY\_RQST\_ID\_PROOF*** = "select purge\_ts from gdp\_owner. prvcy\_rqst\_id\_proof where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_PRIVACY\_REQUEST\_DOC\_UPLOAD\_ID*** = "SELECT %s FROM %s WHERE prvcy\_rqst\_id\_proof\_id='%s'";

**public** **static** **final** String ***GET\_ATTEMPTS\_COUNT\_ID\_PROOF\_DOC\_UPLOAD*** = "select count(\*) from %s where %s = '%s';";

**public** **static** **final** String ***GET\_ATTEMPTS\_COUNT\_DOC\_UPLOAD*** = "select count(\*) from %s where %s = '%s';";

**public** **static** **final** String ***GET\_PURGE\_TS\_REQUEST\_ID*** = "select %s from %s where prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_MAX\_SERV\_FUNC\_LITIGATION\_HOLD\_ID*** = "select max(serv\_func\_ltgtn\_hold\_id) from gdp\_owner.serv\_func\_ltgtn\_hold where serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_COUNT\_OF\_SERVICE\_FUNCTION\_LITIGATION\_HOLDS*** = "select count(\*) from serv\_func\_ltgtn\_hold where serv\_func\_cd = '%s'";

**public** **static** **final** String ***GET\_CREATED\_USER\_FOR\_LATEST\_LH\_RECORD*** = "select crte\_user\_id from serv\_func\_ltgtn\_hold where serv\_func\_cd = '%s' order by serv\_func\_ltgtn\_hold\_id asc;";

**public** **static** **final** String ***GET\_COUNT\_OF\_SERVICE\_FUNCTIONS\_WITH\_PRIVACY\_REQUEST*** = "select count(\*) from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_CHUNK\_DOCUMENTS\_COUNT*** = "select count(\*) from gdp\_owner.doc\_cntn\_chunk where doc\_cntn\_chunk\_id::text like '%s';";

**public** **static** **final** String ***GET\_DOCUMENTS\_UPLOAD\_COUNT*** = "select count(\*) from gdp\_owner.doc\_upld where doc\_upld\_id::text like '%s';";

**public** **static** **final** String ***GET\_AUDIT\_RECORD\_BY\_AUD\_TYPE\_CD*** = "select \* from gdp\_owner.aud\_hist where aud\_type\_cd = '%s' order by aud\_hist\_id desc;";

**public** **static** **final** String ***DELETE\_SERV\_FUNC\_LITIGATION\_ENTRY*** = "DELETE FROM gdp\_owner.serv\_func\_ltgtn\_hold WHERE serv\_func\_cd = '%s';";

**public** **static** **final** String ***GET\_SEARCH\_KEY\_TXT*** = "select %s from gdp\_owner.%s where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_SERV\_FUNC\_PAYLD\_TXT*** = "select %s from gdp\_owner.%s where prvcy\_rqst\_id='%s' and serv\_func\_cd='%s';";

**public** **static** **final** String ***GET\_RESP\_DATA\_TXT\_RESP\_REC*** = "select %s from gdp\_owner.%s where resp\_rec\_nam='%s';";

**public** **static** **final** String ***UPDATE\_FIELD\_ENCRYPT\_SW\_FOR\_ALL\_FIELDS*** = "update gdp\_owner.data\_elmt SET encrypt\_sw=%s where 1=1";

**public** **static** **final** String ***UPDATE\_FILE\_UPLD\_RETRY\_MAX*** ="update config\_parm set config\_parm\_val\_txt= %s where config\_parm\_key\_txt='activity.data.file.upload.retry.max.limit'";

**public** **static** **final** String ***UPDATE\_CONFIG\_PARAM\_VALUE*** ="update config\_parm set config\_parm\_val\_txt= %s where config\_parm\_key\_txt='%s'";

**public** **static** **final** String ***DISABLE\_PCI\_FLAG*** = "update gdp\_owner.ff4j\_features set enable=0 where feat\_uid='pci.data.conversion';";

**public** **static** **final** String ***ENABLE\_PCI\_FLAG*** = "update gdp\_owner.ff4j\_features set enable=1 where feat\_uid='pci.data.conversion';";

**public** **static** **final** String ***UPDATE\_PCI\_FLAG*** = "update gdp\_owner.ff4j\_features set enable=%d where feat\_uid='%s';";

**public** **static** **final** String ***FILTER\_CRITERIA\_ALL*** = "select p.prvcy\_rqst\_id, p.rqst\_type\_cd, p.rqst\_stat\_cd, p.first\_nam, p.lst\_nam, p.rqst\_cntxt\_cd, p.crte\_ts, p.vw\_rqst\_id, p.days\_in\_queue\_num, d.cntry\_cd, s.st\_cd, c.regn\_cd from gdp\_owner.prvcy\_rqst p \n"

+ "left outer join gdp\_owner.data\_prvcy\_law\_assgn d on p.data\_prvcy\_law\_assgn\_id = d.data\_prvcy\_law\_assgn\_id left outer join gdp\_owner.st s on d.st\_id = s.st\_id left outer join gdp\_owner.cntry\_phn\_cd c on d.cntry\_cd = c.cntry\_phn\_cd order by %s %s, lst\_updt\_ts desc limit %s;";

**public** **static** **final** String ***UPDATE\_EMAIL\_VERIFICATION\_SWITCH*** = "update gdp\_owner.email\_ver set email\_ver\_sw = 'true' where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_TIME\_IN\_PRIVACY\_REQUEST\_TABLE*** = "update gdp\_owner.prvcy\_rqst set crte\_ts =\n"

+ " ((SELECT CURRENT\_TIMESTAMP - (1 \* interval '28800 minute'))\n"

+ " ::timestamp without time zone) where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_KEYS\_VALUES\_EMAIL\_TRANSLATION*** = "select label\_key,label\_val from gdp\_owner.email\_label\_trnslt where lang\_cd='%s'and event\_type\_nam='%s' and label\_key = ANY (?)";

**public** **static** **final** String ***GET\_SINGLE\_COLUMN\_VALUE\_FROM\_PRVCY\_RQST*** = "select %s from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id=%s";

**public** **static** **final** String ***UPDATE\_CREATED\_TIMESTAMP\_PRIVACY\_REQUEST\_TABLE*** = "update gdp\_owner.email\_ver set email\_send\_ts =\n"

+ " ((SELECT CURRENT\_TIMESTAMP - (1 \* interval '13 day'))\n"

+ " ::timestamp without time zone) where prvcy\_rqst\_id = %s and email\_ver\_sw = false;";

**public** **static** **final** String ***GET\_DOCUMENT\_APPROVE\_SW*** = "select %s from gdp\_owner.prvcy\_rqst\_id\_proof where prvcy\_rqst\_id=%s";

**public** **static** **final** String ***GET\_CARD\_NUMBER\_CARD\_VERIFY*** = "select card\_num from gdp\_owner.card\_ver where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***GET\_LAST\_UPDATED\_USERID*** = "select lst\_updt\_user\_id from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_LAST\_UPDATED\_TIMESTAMP*** = "select lst\_updt\_ts from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_APPROVED\_TIME\_STAMP\_PRIVACY\_RQST*** = "select rqst\_appr\_ts from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id=%s";

**public** **static** **final** String ***GET\_MOBILE\_VERIFICATION\_ABANDONED\_REQUESTS\_COUNT*** = "select count(\*) from gdp\_owner.phn\_ver where prvcy\_rqst\_id is null and sent\_ts < (SELECT CURRENT\_DATE-2);";

**public** **static** **final** String ***GET\_MOBILE\_VERIFICATION\_RECORDS\_COUNT*** = "select count(\*) from gdp\_owner.phn\_ver where prvcy\_rqst\_id ='%s';";

**public** **static** **final** String ***GET\_DP\_LAW\_USING\_CNTRY\_AND\_STATE*** = "SELECT \* FROM GDP\_OWNER.DATA\_PRVCY\_LAW\_LKP WHERE type\_cd IN (SELECT DISTINCT ON (d.data\_prvcy\_law\_cd) d.data\_prvcy\_law\_cd FROM gdp\_owner.data\_prvcy\_law\_assgn d WHERE d.cntry\_cd = '%s' AND d.st\_id = (SELECT st\_id FROM GDP\_OWNER.ST WHERE st\_cd = '%s'));";

**public** **static** **final** String ***GET\_DP\_LAW\_USING\_CNTRY\_ONLY*** = "SELECT \* FROM GDP\_OWNER.DATA\_PRVCY\_LAW\_LKP WHERE type\_cd IN (SELECT DISTINCT ON (d.data\_prvcy\_law\_cd) d.data\_prvcy\_law\_cd FROM gdp\_owner.data\_prvcy\_law\_assgn d WHERE d.cntry\_cd = '%s' );";

**public** **static** **final** String ***UPDATE\_REGN\_CD\_USER\_PREF*** = "update user\_pref set pref\_regn\_cd = '%s' where user\_id = '%s'";

**public** **static** **final** String ***GET\_DISPATCH\_INTEGRATION\_TYPE*** = "select config\_parm\_val\_txt from gdp\_owner.config\_parm where config\_parm\_key\_txt='servicefunction.integrationtypes.regional.dispatch'";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTIONS\_BY\_REQUESTID*** = "select \* from gdp\_owner.serv\_func\_prvcy\_rqst where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***TRUNCATE\_SERVICE\_FUNCTION\_REGIONS*** = "truncate table gdp\_owner.serv\_func\_rgns;";

**public** **static** **final** String ***GET\_ATTORNEY\_OVERRIDE\_SWITCH*** = "select ovrde\_ver\_sw from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_OVERRIDE\_VERICATION\_STATUS\_COUNT*** = "select count(\*) from gdp\_owner.prvcy\_rqst\_actn where prvcy\_rqst\_id='%s' and prvcy\_rqst\_actn\_cd='%s';";

**public** **static** **final** String ***GET\_ALL\_PRODUCTS\_ELIGIBLE\_FOR\_STATUS\_CHANGE*** = "select \* from gdp\_owner.prdct\_prvcy\_rqst where prvcy\_rqst\_id='%s' and pblsh\_ts notnull and prdct\_rqst\_stat\_cd not in ('CL') and prdct\_ver\_stat\_cd not in ('V');";

**public** **static** **final** String ***GET\_PRODUCT\_VERFICATION\_STATUS*** = "select prdct\_ver\_stat\_cd from gdp\_owner.prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_ENCRYPTED\_DATA\_ELEMENT\_COUNT*** = "select count(\*) from data\_elmt where elmt\_tag\_txt = '%s' and encrypt\_sw = true;";

**public** **static** **final** String ***UPDATE\_KEY\_TRUE\_FOR\_MISSING\_SERVICE\_FUNCTIONS*** = "update serv\_func\_data\_elmt set key\_sw = true where serv\_func\_cd in (select distinct sfd.serv\_func\_cd\n"

+ "from serv\_func\_def sfd inner join serv\_func\_data\_elmt sfde on sfd.serv\_func\_cd = sfde.serv\_func\_cd where key\_sw = false);";

**public** **static** **final** String ***UPDATE\_ENCRYPTION\_SWITCH\_FOR\_GIVEN\_DATA\_ELEMENT*** = "update data\_elmt set encrypt\_sw = false where elmt\_tag\_txt = '%s';";

**public** **static** **final** String ***GET\_MANUAL\_RESPONSE\_TOKEN\_FOR\_A\_REQUEST*** = "select url\_token\_id from serv\_func\_prvcy\_rqst where prvcy\_rqst\_id = %s and serv\_func\_cd = '%s';";

**public** **static** **final** String ***ALERT\_CURRENT\_SCHEMA*** = "SET search\_path = %s";

**public** **static** **final** String ***GET\_FILEREPO\_IDS*** = "SELECT FILE\_REPO\_ID FROM DGR\_PR\_OWNER.FILE\_CNTN\_CHUNK WHERE FILE\_REPO\_ID IN (SELECT FILE\_REPO\_ID FROM DGR\_PR\_OWNER.PRVCY\_RESP\_FILES WHERE ACTVY\_DATA\_UID\_ID='%s')";

**public** **static** **final** String ***CHECK\_FILECONTENT*** = "SELECT count(\*) FROM DGR\_PR\_OWNER.FILE\_CNTN\_CHUNK WHERE FILE\_REPO\_ID='%s'";

**public** **static** **final** String ***GET\_FILEREPO\_DETAILS***="SELECT \* FROM DGR\_PR\_OWNER.FILE\_REPO WHERE FILE\_REPO\_ID='%s'";

**public** **static** **final** String ***DELETE\_FILEREPO\_DETAILS***="DELETE FROM DGR\_PR\_OWNER.FILE\_REPO WHERE FILE\_REPO\_ID='%s'";

// Agent schama query

**public** **static** **final** String ***GET\_INITIATOR\_TYPE\_FROM\_PRVCY\_RQST*** = "select initiator\_type\_cd from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_COUNT\_OF\_AGENT\_TABLE\_USING\_INIT\_TYPE*** = "select count(\*) from gdp\_agent.agent where agncy\_rel\_type\_id = (select lkp\_type\_val\_id from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_cd='%s') and agent\_crte\_user\_id ='%s'";

**public** **static** **final** String ***GET\_AGENT\_RQST\_USING\_AGENT\_PRVCY\_RQST\_ID*** = "select \* from gdp\_agent.agent\_rqst where agent\_rqst\_id in (%s)";

**public** **static** **final** String ***GET\_AGENT\_RQST\_USING\_PRVCY\_RQST\_ID*** = "select \* from gdp\_agent.agent\_rqst where agent\_rqst\_id in (select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id = %s)";

**public** **static** **final** String ***GET\_AGENT\_ID\_FOR\_INITIATOR*** = "select agent\_id from gdp\_agent.agent where agncy\_rel\_type\_id = (select lkp\_type\_val\_id from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_cd='%s') and agent\_crte\_user\_id ='%s'";

**public** **static** **final** String ***GET\_LOOK\_UP\_VALUE*** = "select lkp\_type\_val\_id from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_cd = '%s'";

**public** **static** **final** String ***GET\_CONSENT\_AGREEMENT\_USING\_AGENT\_ID*** = "select \* from gdp\_agent.agent\_consent\_agrmt where agent\_rqst\_id in (%s) order by agent\_consent\_agrmt\_id";

**public** **static** **final** String ***GET\_CONSENT\_AGREEMENT\_USING\_ID*** = "select \* from gdp\_agent.agent\_consent\_agrmt where agent\_rqst\_id in (select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id = %s) order by agent\_consent\_agrmt\_id";

**public** **static** **final** String ***GET\_AUTH\_DOC*** = "select \* from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id in (select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id = '%s') and doc\_type\_id in ("

+ ***GET\_LOOK\_UP\_VALUE*** + ")";

**public** **static** **final** String ***GET\_AUTH\_DOC\_BY\_AGENT\_REQUEST\_ID*** = "select \* from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id in ('%s') and doc\_type\_id in ("

+ ***GET\_LOOK\_UP\_VALUE*** + ")";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_ID*** = "select doc\_upld\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id in (select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id = %s)";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_TABLE*** = "select \* from gdp\_agent.doc\_upld where doc\_upld\_id in ("

+ ***GET\_DOC\_UPLOAD\_ID*** + ") order by doc\_upld\_id";

**public** **static** **final** String ***GET\_DOC\_CHUNK*** = "select count(\*) from gdp\_agent.doc\_cntn\_chunk where doc\_upld\_id ='%s'";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_ID\_USING\_FILENAME*** = "SELECT chunk.doc\_upld\_id FROM gdp\_agent.agent\_auth\_doc doc FULL OUTER JOIN gdp\_agent.doc\_upld chunk ON doc.doc\_upld\_id = chunk.doc\_upld\_id WHERE chunk.doc\_file\_nam= '%s' and doc.agent\_rqst\_id in (select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id = '%s')";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_ID\_USING\_FILENAME\_BY\_AGENT\_REQUEST\_ID*** = "SELECT chunk.doc\_upld\_id FROM gdp\_agent.agent\_auth\_doc doc FULL OUTER JOIN gdp\_agent.doc\_upld chunk ON doc.doc\_upld\_id = chunk.doc\_upld\_id WHERE chunk.doc\_file\_nam= '%s' and doc.agent\_rqst\_id in (%s)";

**public** **static** **final** String ***SELECT\_AGENT\_PRVCY\_RQST\_BY\_ID*** = "Select \* from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***GET\_CARD\_VERIFICATION\_DETAILS\_PRVCY\_RQST*** = "select \* from gdp\_owner.card\_ver where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_EMAIL\_VER\_SW\_TIMESTAMP*** = "UPDATE gdp\_owner.email\_ver SET email\_ver\_sw=false,email\_ver\_ts=null,email\_ver\_stat\_cd='UV' where prvcy\_rqst\_id=%s and email\_ver\_ts IS NOT NULL";

**public** **static** **final** String ***UPDATE\_REQUEST\_WITH\_NECESSARY\_DATA\_FOR\_REMOVAL*** = "UPDATE gdp\_owner.prvcy\_rqst SET crte\_ts = (SELECT CURRENT\_DATE - interval '2 years'), tst\_rqst\_sw = true WHERE prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***DELETE\_CARD\_VERIFICATION\_FOR\_A\_REQUEST*** = "delete from card\_ver where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***DELETE\_OPT\_COUNTRY\_FOR\_A\_REQUEST*** = "delete from opt\_cntry where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***SELECT\_REQUEST\_ID\_BASED\_ON\_NAME\_EMAIL*** = "select prvcy\_rqst\_id from prvcy\_rqst where crte\_user\_id = '%s' and rqst\_email\_addr = '%s';";

**public** **static** **final** String ***UPDATE\_REQUEST\_CREATED\_TIMESTAMP*** = "UPDATE gdp\_owner.prvcy\_rqst SET crte\_ts = (SELECT CURRENT\_DATE - interval '%s days'), tst\_rqst\_sw = true WHERE prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_EMAIL\_VERIFICATION\_COUNT*** = "select email\_ver\_rmndr\_num from email\_ver where prvcy\_rqst\_id = %s and email\_addr = '%s';";

**public** **static** **final** String ***UPDATE\_INITIATOR\_TYPE\_FOR\_REQUEST*** = "UPDATE gdp\_owner.prvcy\_rqst SET initiator\_type\_cd = '%s' WHERE prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_COUNT\_OF\_EMAIL\_NOTIFICATIONS*** = "select count(\*) from email\_notif\_log where corltn\_id = '%s' and msg\_subj\_txt ='%s';";

**public** **static** **final** String ***GET\_SEARCH\_KEYS\_FROM\_PRIVACY\_REQUEST*** = "select srch\_key\_txt from prvcy\_rqst where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_AGENT\_DOCUMENT\_UPLOAD\_STATUS*** = "select lkp\_type\_val\_cd from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_id in(select doc\_stat\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id = %s)";

**public** **static** **final** String ***GET\_AGENT\_RQST\_ACTION\_COUNT*** = "select count(\*) from gdp\_owner.prvcy\_rqst\_actn where prvcy\_rqst\_id= %s";

**public** **static** **final** String ***UPDATE\_IN\_CURING\_DAYS*** = "UPDATE gdp\_agent.agent\_rqst\_cure SET cure\_init\_ts = (SELECT CURRENT\_DATE - interval '%s days') WHERE agent\_rqst\_cure.agent\_rqst\_id = %s;";

**public** **static** **final** String ***UPDATE\_CURING\_ATTEMPT*** = "UPDATE gdp\_agent.agent\_rqst\_cure SET cure\_cmplt\_ts = (SELECT CURRENT\_TIMESTAMP::timestamp without time zone) WHERE agent\_rqst\_cure.agent\_rqst\_id = %s;";

**public** **static** **final** String ***INSERT\_CURING\_ATTEMPT*** = "INSERT INTO gdp\_agent.agent\_rqst\_cure(agent\_rqst\_cure\_id, agent\_rqst\_id, cure\_rsn\_lkp\_val\_id, cure\_cmnt\_txt, cure\_init\_ts, cure\_cmplt\_ts, crte\_ts, lst\_updt\_ts, crte\_user\_id, lst\_updt\_user\_id) values (DEFAULT, '%s', 17, 'Document is in curing',(SELECT CURRENT\_TIMESTAMP::timestamp without time zone),DEFAULT,(SELECT CURRENT\_TIMESTAMP::timestamp without time zone),(SELECT CURRENT\_TIMESTAMP::timestamp without time zone),'gdpuser','gdpuser');";

**public** **static** **final** String ***GET\_AGENT\_AUTH\_DOCUMENT\_STATUS*** = "select lkp\_type\_val\_cd from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_id in (select doc\_stat\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id = %s and agent\_auth\_doc\_id = %s);";

**public** **static** **final** String ***GET\_AGENT\_AUTH\_DOCUMENT\_VERSION*** = "select doc\_ver\_num from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id = %s and agent\_auth\_doc\_id = %s;";

**public** **static** **final** String ***GET\_COUNT\_OF\_AGENT\_DOCUMENT\_HISTORY\_ENTRIES*** = "select count(\*) from gdp\_agent.agent\_authtcn\_doc\_hist where agent\_rqst\_id = %s and doc\_ver\_num = %s;";

**public** **static** **final** String ***GET\_AGENT\_RQST\_ID*** = "select \* from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id in (%s);";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_ID\_BY\_AGENT\_RQST\_ID*** = "select \* from gdp\_owner.agent\_prvcy\_rqst where agent\_rqst\_id IN (%s);";

**public** **static** **final** String ***GET\_AGENT\_ADDR\_ID*** = "select \* from gdp\_agent.agent\_addr where agent\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_AGENT\_AUTH\_DOC*** = "DELETE FROM gdp\_agent.agent\_auth\_doc WHERE agent\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_DOC\_CHUNK\_CNTN*** = "DELETE FROM gdp\_agent.doc\_cntn\_chunk WHERE doc\_upld\_id in (select doc\_upld\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id in (%s));";

**public** **static** **final** String ***DELETE\_DOC\_UPLD*** = "DELETE FROM gdp\_agent.doc\_upld WHERE doc\_upld\_id in (select doc\_upld\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id in (%s));";

**public** **static** **final** String ***DELETE\_CONSENT\_AGRMT*** = "DELETE FROM gdp\_agent.agent\_consent\_agrmt WHERE agent\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_AGENT\_ORG*** = "DELETE FROM gdp\_agent.agent\_org WHERE agent\_id in (select agent\_id from gdp\_agent.agent\_rqst where agent\_rqst\_id in (%s));";

**public** **static** **final** String ***DELETE\_AGENT\_ADDR*** = "DELETE FROM gdp\_agent.agent\_addr WHERE agent\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_ADDR*** = "DELETE FROM gdp\_agent.addr WHERE addr\_id in (select addr\_id from gdp\_agent.agent\_addr where agent\_rqst\_id in (%s) and addr\_id not in(select addr\_id from gdp\_agent.agent\_addr aa where aa.agent\_rqst\_id not in(%s)));";

**public** **static** **final** String ***DELETE\_AGENT\_RQST*** = "DELETE FROM gdp\_agent.agent\_rqst WHERE agent\_rqst\_id in (%s);";

**public** **static** **final** String ***GET\_AGENT\_RQST*** = "select \* from gdp\_agent.agent\_rqst where agent\_rqst\_id in (%s);";

**public** **static** **final** String ***DELETE\_ORG*** = "DELETE FROM gdp\_agent.org WHERE org\_id in (%s);";

**public** **static** **final** String ***GET\_AGENT\_REQUEST\_ID\_FROM\_AUTH\_DOC*** = "select \* from gdp\_agent.agent\_auth\_doc where doc\_upld\_id IN (select doc\_upld\_id from gdp\_agent.agent\_auth\_doc where agent\_auth\_doc\_id IN (%s));";

**public** **static** **final** String ***GET\_RF\_VALUE\_BY\_RQST\_ID*** = "select dpcon.data\_prvcy\_law\_flfl\_days\_num from gdp\_owner.prvcy\_rqst pr inner join gdp\_owner.data\_prvcy\_law\_assgn dpla on pr.data\_prvcy\_law\_assgn\_id=dpla.data\_prvcy\_law\_assgn\_id inner join gdp\_owner.data\_prvcy\_law\_config dpcon on dpla.data\_prvcy\_law\_cd=dpcon.data\_prvcy\_law\_cd and dpcon.rqst\_type\_cd=pr.rqst\_type\_cd and dpcon.data\_acs\_key\_cd='RF' and pr.prvcy\_rqst\_id=%s;";

**public** **static** **final** String ***GET\_AGENT\_AUTH\_DOC\_ID*** = "select agent\_auth\_doc\_id from gdp\_agent.agent\_auth\_doc where doc\_type\_id = %s and agent\_rqst\_id =\n"

+ "(select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id = %s);";

**public** **static** **final** String ***GET\_LOOK\_UP\_TYPE\_ID*** = "select lkp\_type\_val\_id from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_cd = '%s';";

**public** **static** **final** String ***UPDATE\_CREATE\_TIME\_IN\_PRIVACY\_REQUEST\_TABLE*** = "update gdp\_owner.prvcy\_rqst \n"

+ "set crte\_ts =NOW() - INTERVAL '%s DAY' \n" + "where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***FETCH\_KEY\_FROM\_EMAIL\_VER*** = "select %s from gdp\_owner.email\_ver where prvcy\_rqst\_id = %s and email\_addr='%s'";

**public** **static** **final** String ***GET\_AUDIT\_RECORD\_BY\_AUD\_TYPE\_CD\_AND\_REQUEST\_ID*** = "select count(\*) from gdp\_owner.aud\_hist where aud\_type\_cd= '<AUD\_CODE>' and aud\_data\_txt -> 'dataRetentionPurgeStatus'->>0 LIKE '%<REQUEST\_ID>%'";

**public** **static** **final** String ***GET\_AGENT\_RQST\_ID\_BY\_PRVCY\_RQST\_ID*** = "select agent\_rqst\_id from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id=%s";

**public** **static** **final** String ***AGENT\_REQUEST\_EXISTS*** = "select case when count(agent\_rqst\_id)>0 then true else false end from gdp\_agent.agent\_rqst where agent\_rqst\_id=%s";

**public** **static** **final** String ***GET\_EXISTING\_ORG\_IDS*** = "select string\_agg(distinct ao.org\_id::text,',') as org\_ids from gdp\_agent.agent\_rqst ar inner join gdp\_agent.agent\_org ao on ao.org\_id=ar.org\_id and ar.agent\_rqst\_id=%s and ar.org\_id not in (select org\_id from gdp\_agent.agent\_rqst where agent\_rqst\_id!=ar.agent\_rqst\_id and org\_id is not null);";

**public** **static** **final** String ***GET\_EXISTING\_ADDR\_IDS*** = "select string\_agg(distinct aa.addr\_id::text,',') as addr\_ids from gdp\_agent.agent\_rqst ar inner join gdp\_agent.agent\_addr aa on aa.agent\_rqst\_id=ar.agent\_rqst\_id and ar.agent\_rqst\_id=%s and aa.addr\_id not in (select addr\_id from gdp\_agent.agent\_addr where agent\_rqst\_id!=ar.agent\_rqst\_id and addr\_id is not null);";

**public** **static** **final** String ***GET\_EXISTING\_DOC\_UPLD\_IDS*** = "select string\_agg(distinct aad.doc\_upld\_id::text,',') as doc\_upld\_ids from gdp\_agent.agent\_rqst ar inner join gdp\_agent.agent\_auth\_doc aad on aad.agent\_rqst\_id=ar.agent\_rqst\_id and ar.agent\_rqst\_id=%s inner join gdp\_agent.lkp\_type\_val ltv on ltv.lkp\_type\_val\_id=aad.doc\_type\_id and ltv.lkp\_type\_val\_cd='%s' and aad.doc\_upld\_id not in (select doc\_upld\_id from gdp\_agent.agent\_auth\_doc aad where agent\_rqst\_id!=ar.agent\_rqst\_id and doc\_upld\_id is not null);";

**public** **static** **final** String ***GET\_EXISTING\_AGENT\_DOC\_UPLD\_IDS*** = "select string\_agg(distinct aad.agent\_auth\_doc\_id::text,',') as doc\_upld\_ids from gdp\_agent.agent\_rqst ar inner join gdp\_agent.agent\_auth\_doc aad on aad.agent\_rqst\_id=ar.agent\_rqst\_id and ar.agent\_rqst\_id=%s inner join gdp\_agent.lkp\_type\_val ltv on ltv.lkp\_type\_val\_id=aad.doc\_type\_id and ltv.lkp\_type\_val\_cd='%s' and aad.doc\_upld\_id not in (select doc\_upld\_id from gdp\_agent.agent\_auth\_doc aad where agent\_rqst\_id!=ar.agent\_rqst\_id and doc\_upld\_id is not null);";

**public** **static** **final** String ***GET\_EXISTING\_AGENT\_IDS*** = "select string\_agg(distinct ar.agent\_id::text,',') as agent\_ids from gdp\_agent.agent\_rqst ar where ar.agent\_rqst\_id=%s and ar.agent\_id not in(select agent\_id from gdp\_agent.agent\_rqst where ar.agent\_rqst\_id!=agent\_rqst\_id and agent\_id is not null);";

**public** **static** **final** String ***GET\_EXISTING\_DOC\_CHUNK\_IDS*** = "select string\_agg(distinct dcc.doc\_cntn\_chunk\_id::text,',') as doc\_cntn\_chunk\_ids from gdp\_agent.agent\_rqst ar inner join gdp\_agent.agent\_auth\_doc aad on aad.agent\_rqst\_id=ar.agent\_rqst\_id inner join gdp\_agent.doc\_upld du on du.doc\_upld\_id=aad.doc\_upld\_id inner join gdp\_agent.doc\_cntn\_chunk dcc on dcc.doc\_upld\_id=du.doc\_upld\_id and ar.agent\_rqst\_id=%s and aad.doc\_upld\_id not in (select doc\_upld\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id!=ar.agent\_rqst\_id and doc\_upld\_id is not null);";

**public** **static** **final** String ***CHECK\_ORG\_IDS*** = "select case when count(org\_id)>0 then true else false end from gdp\_agent.org where org\_id in(%s);";

**public** **static** **final** String ***CHECK\_ADDR\_IDS*** = "select case when count(addr\_id)>0 then true else false end from gdp\_agent.addr where addr\_id in(%s);";

**public** **static** **final** String ***CHECK\_DOC\_UPLD\_IDS*** = "select case when count(doc\_upld\_id)>0 then true else false end from gdp\_agent.doc\_upld where doc\_upld\_id in(%s);";

**public** **static** **final** String ***CHECK\_AGENT\_IDS*** = "select case when count(agent\_id)>0 then true else false end from gdp\_agent.agent where agent\_id in(%s);";

**public** **static** **final** String ***CHECK\_DOC\_CHUNK\_IDS*** = "select case when count(doc\_cntn\_chunk\_id)>0 then true else false end from gdp\_agent.doc\_cntn\_chunk where doc\_cntn\_chunk\_id in(%s);";

**public** **static** **final** String ***GET\_AUDIT\_RECORD\_BY\_AGENT\_REQUEST*** = "select count(\*) from gdp\_agent.aud\_hist where aud\_data\_txt ->> 'agentRequestId' LIKE '%<REQUEST\_ID>%'";

**public** **static** **final** String ***UPDATE\_AGENT\_RQST\_TO\_TEST\_WITH\_INTERVAL*** = "update gdp\_agent.agent\_rqst set tst\_rqst\_sw=true, crte\_ts= now() - interval '%s' where agent\_rqst\_id=%s;";

**public** **static** **final** String ***GET\_AGENT\_RQST\_IDS\_EXISTS*** = "select case when count(agent\_rqst\_id)>0 then false else true end from gdp\_agent.agent\_rqst where agent\_rqst\_id in(%s);";

**public** **static** **final** String ***GET\_AUD\_COUNT\_FOR\_BATCH*** = "select aud\_data\_txt -> 'deleteRecordCount' from gdp\_agent.aud\_hist order by aud\_ts desc limit 1";

**public** **static** **final** String ***GET\_CUSTOMER\_ID\_VERIFICATION\_COUNT*** = "select atmpt\_cnt from gdp\_owner.cust\_id\_ver where prdct\_prvcy\_rqst\_id='%s';";

**public** **static** **final** String ***GET\_RESET\_VERIFICATION\_AUDIT\_HISTORY\_COUNT*** = "select count(\*) from gdp\_owner.aud\_hist where aud\_elmt\_key\_txt= '%s' and aud\_type\_cd='RESET\_VER'";

**public** **static** **final** String ***GET\_ALL\_DOC\_TYPE\_BY\_AGENT\_RQST\_ID*** = "select ltv.lkp\_type\_val\_cd as doc\_type from gdp\_agent.agent\_auth\_doc aad inner join gdp\_agent.lkp\_type\_val ltv on ltv.lkp\_type\_val\_id=aad.doc\_type\_id and aad.agent\_rqst\_id=%s order by aad.agent\_auth\_doc\_id asc;";

**public** **static** **final** String ***GET\_ALL\_ADDR\_TYPE\_BY\_AGENT\_RQST\_ID*** = "select ltv.lkp\_type\_val\_cd as addr\_type from gdp\_agent.agent\_addr aa inner join gdp\_agent.addr a on a.addr\_id=aa.addr\_id inner join gdp\_agent.lkp\_type\_val ltv on ltv.lkp\_type\_val\_id=a.addr\_type\_id and aa.agent\_rqst\_id=%s order by aa.agent\_addr\_id asc;";

**public** **static** **final** String ***GET\_ALL\_AGRMT\_TYPE\_BY\_AGENT\_RQST\_ID*** = "select ltv.lkp\_type\_val\_cd as agrmt\_type from gdp\_agent.agent\_consent\_agrmt aca inner join gdp\_agent.lkp\_type\_val ltv on ltv.lkp\_type\_val\_id=aca.agrmt\_val\_id and aca.agent\_rqst\_id=%s order by aca.agent\_consent\_agrmt\_id asc;";

**public** **static** **final** String ***GET\_DOC\_OPTIONAL\_CONFIG\_FLAG***="select opt\_sw from gdp\_agent.auth\_doc\_config where data\_prvcy\_law\_cd='%s' and doc\_grp\_cd\_id in (select lkp\_type\_val\_id from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_cd='%s');";

**public** **static** **final** String ***GET\_OPOTIONAL\_DOCS\_AUDIT\_BY\_RQST\_ID***="select aud\_data\_txt->'notUploadedOptionalDocumentGroupCodes' from gdp\_agent.aud\_hist where aud\_type\_val\_id= (select lkp\_type\_val\_id from gdp\_agent.lkp\_type\_val where lkp\_type\_val\_cd='IGNR-DOC') and aud\_data\_txt ->> 'agentRequestId' LIKE '<REQUEST\_ID>'";

**public** **static** **final** String ***GET\_UPLOADED\_DOCS\_TYPE***="select lkp\_type\_val\_cd as lkp\_type from gdp\_agent.lkp\_type\_val ltv where ltv.lkp\_type\_val\_id in (select doc\_type\_id from gdp\_agent.agent\_auth\_doc where agent\_rqst\_id=%s)";

**public** **static** **final** String ***GET\_SERVICE\_FUNCTIONS\_BY\_REGION*** = "select sfd.serv\_func\_cd from gdp\_owner.serv\_func\_def sfd join gdp\_owner.serv\_func\_integration sfi on sfd.serv\_func\_cd = sfi.serv\_func\_cd WHERE sfd.busn\_cntxt\_cd = '%s' AND sfd.decommission\_sw = false AND (sfd.tst\_serv\_func\_sw = false OR sfd.tst\_serv\_func\_sw IS null) and sfi.gdp\_func\_cd = 'D' and sfd.serv\_func\_cd NOT IN (select sfr.serv\_func\_cd from gdp\_owner.serv\_func\_rgns sfr join gdp\_owner.serv\_func\_integration sfi on sfr.serv\_func\_cd = sfi.serv\_func\_cd where sfr.regn\_cd not in ('%s') or (sfr.regn\_cd in ('%s') and sfi.serv\_integration\_type\_cd not in ('%s')))";

**public** **static** **final** String ***GET\_COUNT\_OF\_SERVICE\_FUNCTIONS*** = "select count(\*) from gdp\_owner.serv\_func\_def where busn\_cntxt\_cd = '%s';";

**public** **static** **final** String ***GET\_COUNT\_OF\_HIDDEN\_DATA\_ELEMENTS*** = "select count(\*) from gdp\_owner.data\_elmt where hide\_sw is true;";

**public** **static** **final** String ***UPDATE\_HIDDEN\_DATA\_ELEMENTS*** = "update gdp\_owner.data\_elmt set hide\_sw = false where hide\_sw is true;";

**public** **static** **final** String ***GET\_COUNT\_OF\_EMAIL\_LABEL\_TRANSLATIONS*** = "SELECT COUNT(\*) from gdp\_owner.email\_label\_trnslt;";

**public** **static** **final** String ***UPDATE\_EXPRT\_REQUEST*** = "update gdp\_owner.exprt\_rqst set exprt\_stat\_cd='%s', doc\_upld\_id='%s', exprt\_retry\_cnt =%d where prvcy\_rqst\_id = '%s';";

**public** **static** **final** String ***GET\_EXPORT\_THRESHOLD*** = "select config\_parm\_val\_txt from gdp\_owner.config\_parm where config\_parm\_key\_txt='data.element.threshold.value'";

**public** **static** **final** String ***UPDATE\_EXPORT\_THRESHOLD*** = "update gdp\_owner.config\_parm set config\_parm\_val\_txt='%s' where config\_parm\_key\_txt='data.element.threshold.value'";

**public** **static** **final** String ***GET\_EXPORT\_REQUEST\_RECORD\_COUNT*** = "select count(\*) from gdp\_owner.exprt\_rqst where prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_RECORD\_COUNT*** = "select count(\*) from gdp\_owner.bulk\_rqst\_upld where bulk\_rqst\_upld\_id = %s";

**public** **static** **final** String ***GET\_DOC\_UPLOAD\_CHNK\_RECORD\_COUNT*** = "select count(\*) from gdp\_owner.doc\_cntn\_chunk where doc\_upld\_id = %s";

**public** **static** **final** String ***GET\_EXPRT\_REQUEST\_DISPLAY*** = "select exprt\_disp\_sw from gdp\_owner.exprt\_rqst where exprt\_rqst\_id = %s";

**public** **static** **final** String ***GET\_FILE\_UPLD\_STAT\_ID*** = "select file\_upld\_stat\_id from prvcy\_resp\_file\_upld where file\_repo\_id = %s";

**public** **static** **final** String ***GET\_COLUMN\_VALUE\_FROM\_PRDCT\_PRVCY\_RQST*** = "select %s from prdct\_prvcy\_rqst where prdct\_prvcy\_rqst\_id = %s";

**public** **static** **final** String ***GET\_COLUMN\_VALUE\_FROM\_RESP\_REC*** = "select %s from resp\_rec where resp\_rec\_id = %s";

**public** **static** **final** String ***GET\_RESPONSE\_RECORDS\_DETAILS\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID*** = "SELECT \* FROM DGR\_PR\_OWNER.RESP\_REC\_DTL WHERE RESP\_REC\_DTL\_ID= %s";

**public** **static** **final** String ***GET\_RESP\_REC\_TRAN\_STAT\_ID\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID*** = "SELECT RESP\_REC\_TRAN\_STAT\_ID FROM DGR\_PR\_OWNER.RESP\_REC\_DTL WHERE RESP\_REC\_DTL\_ID= %s";

**public** **static** **final** String ***GET\_RESPONSE\_RECORDS\_DETAILS\_ID\_BY\_PRIVACY\_REQUEST\_ID\_AND\_SERV\_FUNC\_CODE*** = "SELECT RESP\_REC\_DTL\_ID FROM DGR\_PR\_OWNER.RESP\_REC\_DTL WHERE PRVCY\_RQST\_ID= %s AND SERV\_FUNC\_CD = '%s'";

**public** **static** **final** String ***GET\_COUNT\_OF\_ACTIVITY\_DATA\_PART\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID\_AND\_PART\_NUM*** = "SELECT count(\*) FROM DGR\_PR\_OWNER.ACTVY\_DATA\_PART WHERE RESP\_REC\_DTL\_ID = %s AND PART\_NUM = %s";

**public** **static** **final** String ***GET\_COUNT\_OF\_ACTIVITY\_REC\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID\_AND\_PART\_NUM*** = "SELECT count(\*) FROM DGR\_PR\_OWNER.ACTVY\_REC WHERE ACTVY\_DATA\_PART\_ID IN (SELECT ACTVY\_DATA\_PART\_ID FROM DGR\_PR\_OWNER.ACTVY\_DATA\_PART where RESP\_REC\_DTL\_ID = %s AND PART\_NUM = %s)";

**public** **static** **final** String ***GET\_ACTIVITY\_DATA\_UID\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID*** = "SELECT \* FROM DGR\_PR\_OWNER.ACTVY\_DATA\_UID where RESP\_REC\_DTL\_ID = %s";

**public** **static** **final** String ***GET\_COUNT\_OF\_RECORDS\_FOR\_ADR\_STATUS*** = "SELECT count(\*) FROM DGR\_PR\_OWNER.RESP\_REC\_DTL where RESP\_REC\_TRAN\_STAT\_ID = 2";

**public** **static** **final** String ***GET\_FULL\_TXT\_BY\_ACTVY\_REC\_ID*** = "SELECT UTL\_RAW.CAST\_TO\_VARCHAR2(DBMS\_LOB.SUBSTR(ACTVY\_DATA\_FULL\_TXT, 4000,1)) FROM DGR\_PR\_OWNER.ACTVY\_REC WHERE ACTVY\_REC\_ID = %s";

**public** **static** **final** String ***GET\_FULL\_TXT\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID*** = "SELECT \* FROM DGR\_PR\_OWNER.ACTVY\_REC WHERE ACTVY\_DATA\_PART\_ID IN (SELECT ACTVY\_DATA\_PART\_ID FROM DGR\_PR\_OWNER.ACTVY\_DATA\_PART where RESP\_REC\_DTL\_ID = %s)";

**public** **static** **final** String ***GET\_ERR\_TXT\_BY\_RESPONSE\_RECORDS\_DETAILS\_ID*** = "SELECT UTL\_RAW.CAST\_TO\_VARCHAR2(DBMS\_LOB.SUBSTR(ERR\_TXT, 4000,1)) FROM DGR\_PR\_OWNER.PRVCY\_RESP\_ERR WHERE RESP\_REC\_DTL\_ID = %s";

**public** **static** **final** String ***GET\_STATUS\_FROM\_RESPONSE\_RECORDS\_TABLE***="SELECT LKP\_TYPE\_VAL\_NAM FROM DGR\_PR\_OWNER.LKP\_TYPE\_VAL WHERE LKP\_TYPE\_VAL\_ID IN (SELECT RESP\_REC\_TRAN\_STAT\_ID FROM DGR\_PR\_OWNER.RESP\_REC\_DTL WHERE PRVCY\_RQST\_ID='%s' AND SERV\_FUNC\_CD='%s')";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_FILE\_UPLOAD\_DATA*** = "select count(\*) from prvcy\_resp\_file\_upld where prvcy\_resp\_file\_upld\_id=%s";

**public** **static** **final** String ***GET\_FILE\_REPO\_ID*** = "select count(\*) from file\_repo where file\_repo\_id = (select file\_repo\_id from prvcy\_resp\_file\_upld where prvcy\_resp\_file\_upld\_id=%s)";

**public** **static** **final** String ***GET\_PRVCY\_RESP\_FILE\_ID*** = "select count(\*) from file\_cntn\_chunk where file\_repo\_id = (select file\_repo\_id from prvcy\_resp\_file\_upld where prvcy\_resp\_file\_upld\_id=%s)";

**public** **static** **final** String ***GET\_FILE\_REPO\_ID\_FOR\_PRVCY\_RESP\_FILE\_ID*** = "select file\_repo\_id from prvcy\_resp\_file\_upld where prvcy\_resp\_file\_upld\_id='%s'";

**public** **static** **final** String ***GET\_UPLD\_ID*** = "select prvcy\_resp\_file\_upld\_id from prvcy\_resp\_file\_upld where prvcy\_resp\_file\_id=%s";

**public** **static** **final** String ***GET\_BULK\_UPLOAD\_STATUS\_CODE***= "SELECT stat\_cd FROM gdp\_owner.bulk\_rqst\_upld where grp\_rqst\_id=%s and bulk\_rqst\_upld\_id=%s;";

**public** **static** **final** String ***GET\_SUCCESS\_BULK\_ACTVY\_LOG*** ="select %s from gdp\_owner.bulk\_rqst\_actvy\_log where bulk\_rqst\_upld\_id='%s'order by crte\_ts desc";

**public** **static** **final** String ***GET\_FAILED\_BULK\_ACTVY\_LOG*** ="select %s from gdp\_owner.bulk\_rqst\_actvy\_log order by crte\_ts desc";

**public** **static** **final** String ***GET\_BULK\_RQST\_UPLD\_ID*** = "select count(\*) from gdp\_owner.bulk\_rqst\_upld where bulk\_rqst\_upld\_id=%s;";

**public** **static** **final** String ***GET\_BULK\_RQST\_ACTIVITY\_STATUS\_COUNT*** = "select count(\*) from gdp\_owner.bulk\_rqst\_actvy\_log where bulk\_rqst\_upld\_id=%s and bulk\_rqst\_actvy\_cd='%s';";

**public** **static** **final** String ***GET\_BULK\_RQST\_DOC\_UPLD\_ID*** = "select doc\_upld\_id from gdp\_owner.bulk\_rqst\_upld where bulk\_rqst\_upld\_id=%s;";

**public** **static** **final** String ***GET\_BULK\_RQST\_STAT\_CD*** = "select stat\_cd from gdp\_owner.bulk\_rqst\_upld where bulk\_rqst\_upld\_id=%s;";

**public** **static** **final** String ***GET\_PRVCY\_RQST\_FOR\_BULK\_REQUEST*** = "select \* from gdp\_owner.prvcy\_rqst where rqst\_type\_cd = '%s' and prvcy\_rqst\_id in (SELECT prvcy\_rqst\_id FROM gdp\_owner.grp\_prvcy\_rqst where bulk\_rqst\_upld\_id=%s);";

**public** **static** **final** String ***GET\_COUNT\_PRVCY\_RQST\_CREATED\_FOR\_BULK\_REQUEST*** = "select count(\*) from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id in (SELECT prvcy\_rqst\_id FROM gdp\_owner.grp\_prvcy\_rqst where bulk\_rqst\_upld\_id=%s);";

**public** **static** **final** String ***GET\_COUNT\_AGENT\_PRVCY\_RQST\_BY\_ID*** = "Select count(\*) from gdp\_owner.agent\_prvcy\_rqst where prvcy\_rqst\_id in (SELECT prvcy\_rqst\_id FROM gdp\_owner.grp\_prvcy\_rqst where bulk\_rqst\_upld\_id=%s);";

**public** **static** **final** String ***GET\_BULK\_REQUEST\_ACTIVITY\_LOG\_FOR\_GIVEN\_ID*** = "select \* from bulk\_rqst\_actvy\_log where bulk\_rqst\_upld\_id = %s";

**public** **static** **final** String ***GET\_DAYS\_TO\_PROCESS\_VALUES*** = "select \n" +

"p.days\_to\_prcss\_num as days\_to\_prcss\_num\n" +

"from gdp\_owner.PRVCY\_RQST p,gdp\_owner.data\_prvcy\_law\_assgn dpla \n" +

"where dpla.data\_prvcy\_law\_assgn\_id = p.data\_prvcy\_law\_assgn\_id \n" +

"and rqst\_type\_cd IN ('%s') and p.rqst\_stat\_cd in ('C','DL') and p.rqst\_cntxt\_cd in ('B2C') \n" +

"and p.crte\_ts >= '%s' and p.crte\_ts <= '%s' \n" +

"and p.tst\_rqst\_sw = false \n" +

"and p.days\_to\_prcss\_num is not null\n" +

"and dpla.data\_prvcy\_law\_cd in ('%s') order by days\_to\_prcss\_num asc";

**public** **static** **final** String ***GET\_MEDIAN\_VALUE*** = "select median\_days from gdp\_owner.cntn\_mtrc\_rpt where cntn\_mtrc\_id='%s' and data\_prvcy\_law\_cd='%s' and rqst\_type\_cd='%s'";

**public** **static** **final** String ***GET\_STATUS\_OF\_REQUESTID*** = "select rqst\_stat\_cd from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***UPDATE\_EMAIL\_VER\_TS*** = "update gdp\_owner.email\_ver set email\_ver\_ts='%s' where prvcy\_rqst\_id='%s'";

**public** **static** **final** String ***UPDATE\_EMAIL\_VERIFICATION\_STAT\_CD*** = "update gdp\_owner.email\_ver set email\_ver\_stat\_cd = 'V' where prvcy\_rqst\_id = %s;";

**public** **static** **final** String ***GET\_DAYS\_REMAINING\_OF\_REQUESTID*** = "select days\_in\_queue\_num from gdp\_owner.prvcy\_rqst where prvcy\_rqst\_id='%s'";

**private** QueryConstants() {

}

}

**package** com.mastercard.testing.gdp.api.steps;

**import** com.mastercard.testing.gdp.api.aspect.ContextVariableTestData;

**import** com.mastercard.testing.gdp.api.aspect.JsonTestData;

**import** org.assertj.core.api.Assertions;

**import** org.jbehave.core.annotations.Given;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** java.util.ArrayList;

**import** java.util.List;

@Component

**public** **class** ContextVariablesSteps {

@Autowired

**private** ContextVariableTestData variableTestData;

@Autowired

**private** JsonTestData jsonTestData;

/\*\*

\* Save string value in context

\*

\* **@param** name

\* variable name

\* **@param** value

\* variable value

\*/

@Given("variable '$name' equals '$value'")

**public** **void** saveContextVariable(String name, String value) {

variableTestData.setContextVariable(name, value);

}

/\*\*

\* Remove variable from context

\*

\* **@param** name

\* variable name

\*/

@Given("variable '$name' is removed")

**public** **void** removeContextVariable(String name) {

variableTestData.setContextVariable(name, **null**);

}

/\*\*

\* Push value into array

\*

\* **@param** name

\* list name

\* **@param** value

\* value

\*/

@Given("value '$value' is pushed into list '$name'")

**public** **void** pushValueIntoList(String value, String name) {

Object list = variableTestData.getContextVariable(name);

**if** (list == **null**) {

list = **new** ArrayList<String>();

variableTestData.setContextVariable(name, list);

}

Assertions.assertThat(list **instanceof** List)

.overridingErrorMessage(String.format("Variable %s (type = %s) is not a List", name, list.getClass()))

.isTrue();

((List<String>) list).add(value);

}

/\*\*

\* Save list of string values in context

\*

\* **@param** name

\* variable name

\* **@param** value

\* variable values

\*/

@Given("list '$name' equals '$value'")

**public** **void** saveContextVariable(String name, List<String> value) {

variableTestData.setContextVariable(name, value);

}

/\*\*

\* Save exists context variable with new name

\*

\* **@param** name

\* exists variable name

\* **@param** alias

\* alias name

\*/

@Given("alias of '$name' is '$alias'")

**public** **void** saveContextVariableAs(String name, String alias) {

variableTestData.setContextVariable(alias, variableTestData.getContextVariable(name));

}

/\*\*

\* Save json object in context variables. Json object is got by jsonPath

\*

\* **@param** jsonPath

\* path in json to object for save, array indexing is not allowed

\* **@param** variableName

\* new context variable name

\*/

@Given("json data '$jsonPath' saved as '$variableName'")

**public** **void** saveJsonDataAsVariable(String jsonPath, String variableName) {

variableTestData.setContextVariable(variableName, jsonTestData.getMap(jsonPath));

}

**public** String getContextVariableData(String key) {

**return** variableTestData.getValue(key);

}

}

**package** com.mastercard.testing.gdp.ui.framework;

**import** com.mastercard.quality.engineering.common.enumeration.WebExecutionPlatform;

**import** com.mastercard.quality.engineering.eyes.utils.factory.EyesFactory;

**import** com.mastercard.quality.engineering.eyes.utils.factory.EyesProperties;

**import** com.mastercard.quality.engineering.eyes.utils.testresults.VisualTestsResults;

**import** com.mastercard.quality.engineering.mtaf.df.factory.DriverFactory;

**import** com.mastercard.quality.engineering.mtaf.ui.providers.MasterCardWebDelegatingDriverProvider;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.firefox.FirefoxOptions;

**import** org.openqa.selenium.remote.CapabilityType;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.remote.LocalFileDetector;

**import** org.openqa.selenium.remote.RemoteWebDriver;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.context.annotation.Primary;

**import** org.springframework.core.env.Environment;

**import** org.springframework.stereotype.Component;

**import** java.util.HashMap;

**import** **static** com.mastercard.testing.gdp.ui.tests.constants.UIConstants.***DOWNLOAD\_FILEPATH***;

@Component

@Primary

**public** **class** CustomCapabilitiesWebDriverProvider **extends** MasterCardWebDelegatingDriverProvider {

@Autowired

Environment env;

@Autowired

DriverFactory driverFactory;

@Autowired

EyesFactory eyesFactory;

@Autowired

@Qualifier("webEyesProperties")

EyesProperties eyesProperties;

@Autowired

VisualTestsResults visualTestsResults;

**public** CustomCapabilitiesWebDriverProvider(DriverFactory driverFactory, EyesFactory eyesFactory,

@Qualifier("webEyesProperties") EyesProperties eyesProperties, VisualTestsResults visualTestsResults) {

**super**(driverFactory, eyesFactory, eyesProperties, visualTestsResults);

}

**private** **static** **final** String ***ENABLE\_VNC*** = "enableVNC";

**private** **static** **final** String ***DOWNLOAD\_DEFAULT\_DIRECTORY*** = "download.default\_directory";

**private** **static** **final** String ***PREFS*** = "prefs";

**private** **static** **final** String ***DEFAULT\_CONTENT\_SETTINGS*** = "profile.default\_content\_settings.popups";

**private** **static** **final** String ***SHOW\_POP\_UP\_FOR\_EACH\_DOWNLOAD*** = "download.prompt\_for\_download";

**private** WebExecutionPlatform webPlatform;

@Override

**public** **void** initialize(WebExecutionPlatform platform) {

**this**.webPlatform = platform;

**super**.initialize(platform);

}

@Override

**public** **void** initialize() {

DesiredCapabilities customCapabilities = **new** DesiredCapabilities();

customCapabilities.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, **true**);

customCapabilities.setCapability(CapabilityType.ACCEPT\_INSECURE\_CERTS, **true**);

**if** (env.getProperty(***ENABLE\_VNC***, Boolean.**class**, **false**)) {

customCapabilities.setCapability(***ENABLE\_VNC***, **true**);

}

**if** (webPlatform == WebExecutionPlatform.***GRID\_CHROME***) {

customCapabilities.setCapability(ChromeOptions.CAPABILITY, **this**.getDownloadFileChromeOptions());

WebDriver driver = driverFactory.getWebDriver(webPlatform, customCapabilities);

RemoteWebDriver remoteDriver = (RemoteWebDriver) driver;

remoteDriver.setFileDetector(**new** LocalFileDetector());

**this**.delegate.set(remoteDriver);

} **else** **if** (webPlatform.toString().toLowerCase().contains("BROWSERSTACK\_".toLowerCase())) {

**this**.delegate.set((**new** BrowserStackRunner()).get());

} **else** **if** (webPlatform == WebExecutionPlatform.***GRID\_FIREFOX***) {

customCapabilities.setCapability(FirefoxOptions.FIREFOX\_OPTIONS, **this**.getDownloadFileFirefoxOptions());

WebDriver driver = driverFactory.getWebDriver(webPlatform, customCapabilities);

**this**.delegate.set(driver);

} **else** {

customCapabilities.setCapability(ChromeOptions.CAPABILITY, **this**.getDownloadFileChromeOptions());

WebDriver driver = driverFactory.getWebDriver(webPlatform, customCapabilities);

**this**.delegate.set(driver);

}

}

**private** ChromeOptions getDownloadFileChromeOptions() {

HashMap<String, Object> chromePrefs = **new** HashMap<>();

chromePrefs.put(***DEFAULT\_CONTENT\_SETTINGS***, 0);

chromePrefs.put(***SHOW\_POP\_UP\_FOR\_EACH\_DOWNLOAD***, **false**);

chromePrefs.put(***DOWNLOAD\_DEFAULT\_DIRECTORY***, System.getProperty("user.dir") + ***DOWNLOAD\_FILEPATH***);

ChromeOptions options = **new** ChromeOptions();

options.setExperimentalOption(***PREFS***, chromePrefs);

**return** options;

}

**private** FirefoxOptions getDownloadFileFirefoxOptions() {

FirefoxOptions fireFoxoptions = **new** FirefoxOptions();

fireFoxoptions.addPreference(***DEFAULT\_CONTENT\_SETTINGS***, 0);

fireFoxoptions.addPreference(***SHOW\_POP\_UP\_FOR\_EACH\_DOWNLOAD***, **false**);

fireFoxoptions.addPreference(***DOWNLOAD\_DEFAULT\_DIRECTORY***, System.getProperty("user.dir") + ***DOWNLOAD\_FILEPATH***);

**return** fireFoxoptions;

}

}

**package** com.mastercard.testing.gdp.ui.framework;

**import** com.mastercard.quality.engineering.mtaf.ui.providers.MasterCardMobileDelegatingDriverProvider;

**import** com.mastercard.testing.gdp.ui.tests.exception.GDPUIException;

**import** org.jbehave.web.selenium.WebDriverProvider;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.core.env.Environment;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** GDPDriverProviderFactory {

@Autowired

**private** CustomCapabilitiesWebDriverProvider webDriverProvider;

@Autowired

CustomIEDriverProvider customIEDriverProvider;

@Autowired

**private** MasterCardMobileDelegatingDriverProvider mobileDriverProvider;

@Autowired

**private** Environment env;

**public** WebDriverProvider getDriverProvider() {

**if** ((env.getProperty("default.web.execution.platform") != **null**)) {

**return** webDriverProvider;

} **else** **if** (env.getProperty("default.mobile.execution.platform") != **null**) {

**return** mobileDriverProvider;

} **else** **if** ("IE".equalsIgnoreCase(env.getProperty("browser"))) {

**return** customIEDriverProvider;

} **else** {

**throw** **new** GDPUIException(

"Neither default.web.execution.platform nor default.mobile.execution.platform is set to appropriate value.");

}

}

}

**package** com.mastercard.testing.gdp.ui.tests.constants;

**import** java.io.File;

**public** **class** UIConstants {

**public** **static** **final** String ***DISABLED*** = "disabled";

**public** **static** **final** String ***ENABLED*** = "enabled";

**private** **static** **final** String ***DB\_DATE\_FORMAT*** = "yyyy-MM-dd HH:mm:ss.SSSSSS";

**public** **static** **final** String ***TYPE*** = "TYPE";

**public** **static** **final** String ***CREATE\_GROUP\_INFO\_PAGE\_URL\_PATTERN*** = "newrequest/grouprequest";

**public** **static** **final** String ***PERSONAL\_INFO\_PAGE\_URL\_PATTERN*** = "newrequest/personalinfo";

**public** **static** **final** String ***REVIEW\_PAGE\_URL\_PATTERN*** = "dgr-aob-web/index.html#/newrequest/review";

**public** **static** **final** String ***DASHBOARD\_PAGE\_URL\_PATTERN*** = "dgr-aob-web/index.html#/dashboard";

**public** **static** **final** String ***CONFIRMATION\_PAGE\_URL\_PATTERN*** = "dgr-aob-web/index.html#/newrequest/confirmation";

**public** **static** **final** String ***VALUE\_ATTRIBUTE*** = "value";

**public** **static** **final** String ***READ\_ONLY*** = "readonly";

**public** **static** **final** String ***NEEDS\_REVIEW\_STATUS*** = "Needs Review";

**public** **static** **final** String ***REVIEWED\_STATUS*** = "Reviewed";

**public** **static** **final** String ***JS\_EXECUTOR\_TO\_CLICK*** = "arguments[0].click();";

**public** **static** **final** String ***REQUEST\_STATUS\_HEADER*** = "STATUS";

**public** **static** **final** String ***REQUEST\_RECORDS\_HEADER*** = "RECORDS";

**public** **static** **final** String ***REQUEST\_PUBLISHED\_HEADER*** = "PUBLISHED";

**public** **static** **final** String ***VIEW\_ID\_HEADER*** = "ViewID";

**public** **static** **final** String ***COMPLETED\_STATUS*** = "Completed";

**public** **static** **final** String ***UNDERSCORE*** = "\_";

**public** **static** **final** String ***COLON*** = ":";

**public** **static** **final** String ***KEY*** = "key";

**public** **static** **final** String ***ID*** = "id";

**public** **static** **final** String ***PHONE\_NUMBERS*** = "phoneNumbers";

**public** **static** **final** String ***INTERNAL\_CMNTS*** = "#internalCommentsValue";

**public** **static** **final** String ***AOB\_GET\_REQUEST\_STATUS\_BY\_REQ\_ID*** = "//td[@data-title='Request ID']//\*[starts-with(normalize-space(.),'%s')]/parent::td/following-sibling::td[@data-title='Status']/span";

**public** **static** **final** String ***AOB\_GET\_REQUEST\_TYPE\_BY\_REQ\_ID*** = "//td[@data-title='Request ID']//\*[starts-with(normalize-space(.),'%s')]/parent::td/following-sibling::td[@data-title='Type']/span";

**public** **static** **final** String ***AOB\_GET\_REQUEST\_BY\_REQ\_ID*** = "(//td[@data-title='Request ID']//\*[normalize-space(text())='%s']/ancestor::td[@data-title='Request ID']|//td[@data-title='Request ID']//span[contains(@id,'privacyRequestId') and not(descendant::a)][contains(.,'%s')]/ancestor::td[@data-title='Request ID'])";

**public** **static** **final** String ***AOB\_GET\_REQUEST\_ENTIRE\_ROW*** = "//td[@data-title='Request ID']//\*[starts-with(normalize-space(.),'%s')]/parent::td[@data-title='Request ID']//ancestor::tr";

**public** **static** **final** String ***MAX\_LENGTH\_HUNDREAD*** = "100";

**public** **static** **final** String ***MAX\_LENGTH\_FIFTY*** = "50";

**public** **static** **final** String ***COUNTRY\_CODE*** = "countrycode";

**public** **static** **final** String ***MOBILE\_NUM*** = "mobilenum";

**public** **static** **final** String ***ADDRESS*** = "address";

**public** **static** **final** String ***CARD\_NUMBER*** = "cardnumber";

**public** **static** **final** String ***EXPIRATION\_DATE*** = "expirationdate";

**public** **static** **final** String ***EMAIL*** = "email";

**public** **static** **final** String ***SPACE*** = " ";

**public** **static** **final** String ***HYPHEN*** = "-";

**public** **static** **final** String ***LESS\_THAN*** = "<";

**public** **static** **final** String ***GREATER\_THAN*** = ">";

**public** **static** **final** String ***DATE\_TIME\_FORMAT\_DB*** = ***DB\_DATE\_FORMAT***;

**public** **static** **final** String ***PRODUCT\_STATUS\_CHANGE*** = "PRD\_ST\_CHG";

**public** **static** **final** String ***AUD\_DESC*** = "aud\_desc";

**public** **static** **final** String ***PRODUCT\_REQUEST\_STATUS\_CHANGED\_TO*** = "Product Request status changed to";

**public** **static** **final** **int** ***SUCCESS\_STATUS\_CODE*** = 200;

**public** **static** **final** String ***UPLOADFILEPATH*** = "src/main/resources/config/TestData/";

**public** **static** **final** String ***DISABLED\_STATUS*** = "disabled";

**public** **static** **final** String ***ERROR\_0\_DUPLICATE*** = "error\_0\_duplicate";

**public** **static** **final** String ***REQUEST\_ID\_HEADER*** = "REQUEST ID";

**public** **static** **final** String ***DATE\_TIME\_FORMAT\_DB\_B2C*** = ***DB\_DATE\_FORMAT***;

**public** **static** **final** String ***DUMMY\_DASHBOARD\_PAGE\_PATTERN*** = "dummydashboard";

**public** **static** **final** String ***VIEW\_REQUEST\_KEY*** = "viewRequest";

**public** **static** **final** String ***PRODUCT\_SUMMARY\_NOT\_DISPLAYED*** = "The Product Summary header is not displayed";

**public** **static** **final** String ***ENABLED\_STATUS*** = "enabled";

**public** **static** **final** String ***MAX\_LENGTH\_TWO\_FIFTY\_SIX*** = "256";

**public** **static** **final** String ***MAX\_LENGTH\_FIVE\_HUNDRED*** = "500";

**public** **static** **final** String ***ATTRIBUTE\_MAXLENGHT*** = "maxlength";

**public** **static** **final** **int** ***MAINWINDOWINDEX*** = 0;

**public** **static** **final** **int** ***FIRSTWINDOWINDEX*** = 1;

**public** **static** **final** String ***TEXT\_LOCATOR*** = "//\*[text()='";

**public** **static** **final** String ***CARET\_UP*** = "caret-up";

**public** **static** **final** String ***CARET\_DOWN*** = "caret-down";

**public** **static** **final** String ***COMMA*** = ",";

**public** **static** **final** String ***PLACEHOLER\_ATTRIBUTE*** = "placeholder";

**public** **static** **final** Byte ***GMT\_TO\_CST\_TIME\_DIFF*** = -6;

**public** **static** **final** String ***ERROR\_MESSAGE\_DATE\_ONE*** = "Exception occurred while parsing date :";

**public** **static** **final** String ***ERROR\_MESSAGE\_DATE\_TWO*** = "Submitted date is not in the specified format";

**public** **static** **final** String ***RECORD\_SORT\_ERROR\_MESSAGE*** = "Records are not sorted properly by 'records'";

**public** **static** **final** String ***STATUS\_ERROR\_MESSAGE*** = "Status is not as expected for record";

**public** **static** **final** String ***DATE\_SUBMITTED\_LABEL\_ERROR\_MESSAGE*** = "Date Submitted label text is not as expected";

**public** **static** **final** String ***DATE\_SUBMITTED\_VALUE\_ERROR\_MESSAGE*** = "Date Submitted value is blank";

**public** **static** **final** String ***PARTIAL*** = "partial";

**public** **static** **final** String ***ORDERBY*** = "ORDERBY";

**public** **static** **final** String ***DAYS\_REMAINING*** = "Days Remaining";

**public** **static** **final** String ***COUNTRY*** = "Country";

**public** **static** **final** String ***ACTUAL\_STRING*** = "Actual::";

**public** **static** **final** String ***STATUS*** = "Status";

**public** **static** **final** String ***DATE\_SUBMITTED\_DB*** = "Date Submitted";

**public** **static** **final** String ***DATE\_APPROVED\_DB*** = "Date Approved";

**public** **static** **final** String ***FIRST\_NAME\_DB*** = "First Name";

**public** **static** **final** String ***LAST\_NAME\_DB*** = "Last Name";

**public** **static** **final** String ***REQUESTER\_DB*** = "Requester";

**public** **static** **final** String ***TYPE\_DB*** = "Type";

**public** **static** **final** String ***VIEW\_ID*** = "View ID";

**public** **static** **final** String ***LASTNAME*** = "LASTNAME";

**public** **static** **final** String ***REQUESTER*** = "REQUESTER";

**public** **static** **final** String ***FIRSTNAME*** = "FIRSTNAME";

**public** **static** **final** String ***REVIEWED\_STATUS\_CODE*** = "RV";

**public** **static** **final** String ***DATE\_TIME\_FORMAT\_DB\_B2B*** = ***DB\_DATE\_FORMAT***;

**public** **static** **final** String ***DATE\_FORMAT*** = "dd MMM yyyy";

**public** **static** **final** String ***PLACE\_HOLDER*** = ***PLACEHOLER\_ATTRIBUTE***;

**public** **static** **final** String ***REJECT\_DESCRIPTION*** = "This request was declined on ";

**public** **static** **final** String ***BY*** = " by ";

**public** **static** **final** String ***REJECT\_REASON*** = "Decline reason: ";

**public** **static** **final** String ***PRODUCT\_LEGAL\_NOTE\_TXT*** = "Product Legal Note\*\*";

**public** **static** **final** String ***ASC*** = "asc";

**public** **static** **final** String ***DESC*** = "desc";

**public** **static** **final** String ***UP*** = "up";

**public** **static** **final** String ***DOWN*** = "down";

**public** **static** **final** String ***DATE\_SUBMITTED\_HEADER*** = ***DATE\_SUBMITTED\_DB***;

**public** **static** **final** String ***REQUEST\_ID*** = "Request ID";

**public** **static** **final** String ***STATUS\_HEADER*** = ***STATUS***;

**public** **static** **final** String ***CLASS\_ATTRIBUTE*** = "class";

**public** **static** **final** String ***DATE*** = "Date";

**public** **static** **final** String ***REQUEST\_TYPE\_CODE*** = "rqst\_type\_cd";

**public** **static** **final** String ***APPROVE\_LABEL*** = "Approve";

**public** **static** **final** String ***DECLINE\_LABEL*** = "Decline";

**public** **static** **final** String ***APPROVE\_RECORD\_CONSENT\_TEXT*** = "I approve the deletion of this record.";

**public** **static** **final** String ***APPROVE\_DOCUMENT\_CONSENT\_TEXT*** = "I approve the deletion of this document.";

**public** **static** **final** String ***APPROVE\_RECORD\_CONSENT\_TEXT\_UPDATE*** = "I approve the update of this record.";

**public** **static** **final** String ***INTERNAL\_COMMENTS\_LABEL*** = "Internal Comments";

**public** **static** **final** String ***MESSAGE\_TO\_REQUESTER\_LABEL*** = "Message to Requester";

**public** **static** **final** String ***MESSAGE\_TO\_REQUESTER\_VERBIAGE*** = "This message will be viewed by the requester when this record is published or exported.";

**public** **static** **final** String ***MESSAGE\_TO\_REQUESTER\_VERBIAGE\_DOCUMENTS*** = "This message will be viewed by the requester when this document is published.";

**public** **static** **final** String ***ERROR\_MESSAGE\_TO\_REQUESTER\_FIELD*** = "Error - Message to requester is required.";

**public** **static** **final** String ***ERROR\_INTERNAL\_COMMENTS\_FIELD*** = "Error - Internal Comments is required";

**public** **static** **final** String ***ALLOWED\_VALUES\_LABEL*** = "0 of 256 characters";

**public** **static** **final** String ***REASON\_FOR\_DECLINE\_LABEL*** = "Reason for Decline:";

**public** **static** **final** String ***ERROR\_MESSAGE\_APPROVE\_CHECKBOX*** = "Error - You must agree to approve this action";

**public** **static** **final** String ***ERROR\_MESSAGE\_DECLINE\_REASONS\_LIST*** = "Error - Reason for Decline is required";

**public** **static** **final** String ***ERROR\_MESSAGE\_DECLINE\_REASONS\_REQUIRED*** = "Error - Decline reason is required";

**public** **static** **final** String ***DATE\_TIME\_FORMAT*** = "dd MMM yyyy 'at' hh:mm a 'US CST'";

**public** **static** **final** String ***RETURN\_TO\_REVIEW\_REQUEST*** = "< Return to Review Request";

**public** **static** **final** String ***RECORDS*** = "records";

**public** **static** **final** String ***OUTER\_HTML*** = "outerHTML";

**public** **static** **final** String ***MANUAL\_RESPONSE\_PAGE*** = "manualResponse";

**public** **static** **final** String ***RECORD\_DETAILS\_PAGE*** = "recordDetails";

**public** **static** **final** String ***NOT\_DISPLAYED*** = " is NOT displayed";

**public** **static** **final** String ***APPROVE*** = "approve";

**public** **static** **final** String ***READY\_TO\_PROCESS*** = "Ready to Process";

**public** **static** **final** String ***IN\_PROGRESS*** = "In Progress";

**public** **static** **final** String ***CARD\_NUMBER\_MANUAL\_RES*** = "Card Number";

**public** **static** **final** String ***EXPIRATION\_DATE\_MANUAL\_RES*** = "Expiration Date";

**public** **static** **final** String ***PHONE\_NUMBER\_MANUAL\_RES*** = "phoneNumber";

**public** **static** **final** String ***REQUEST\_TYPE*** = "Request type";

**public** **static** **final** String ***VERIFIED\_STATUS*** = "Verified";

**public** **static** **final** String ***DELETE\_RESAON*** = "Requester does not want any of their personal data to be retained by Mastercard.";

**public** **static** **final** String ***CARD\_NUMBER\_MASKING\_FORMAT*** = "............";

**public** **static** **final** **int** ***DELTA\_IN\_MILLISECONDS*** = 120000;

**public** **static** **final** String ***DATE\_TIME\_FORMAT\_UI\_ATTORNEY*** = "dd MMM yyyy 'at' hh:mm a 'US CST by GDP Super User'";

**public** **static** **final** String ***COUNTRY\_WHERE\_LOCATED*** = "Country where located";

**public** **static** **final** String ***DATE\_SUBMITTED*** = "DateSubmitted";

**public** **static** **final** String ***UPDATE\_APPROVE\_RECORD\_CONSENT\_TEXT*** = "I approve the update of this record.";

**public** **static** **final** String ***MM\_DD\_YY\_FORMAT\_UI*** = "MMM-dd-yyyy HH:mm a";

**public** **static** **final** String ***REQUESTID*** = "requestId";

**public** **static** **final** String ***REASON\_FOR\_DO\_NOT\_DELETE\_LABEL*** = "Please help us understand the reason for not deleting:";

**public** **static** **final** String ***DATE\_TIME\_FORMAT\_UI*** = "dd MMM yyyy 'at' hh:mm a 'US CST'";

**public** **static** **final** String ***FORWARD\_SLASH*** = "/";

**public** **static** **final** String ***TEXT\_LOCATOR\_CONTAINS*** = "//\*[contains(text(),'";

**public** **static** **final** String ***RECORD*** = "record";

**public** **static** **final** String ***ACTION\_BY*** = "Action By";

**public** **static** **final** String ***MESSAGE\_TO\_REQUESTER*** = "Message To Requester";

**public** **static** **final** String ***INTERNAL*** = "Internal";

**public** **static** **final** String ***EXTERNAL*** = "External";

**public** **static** **final** String ***ACTION*** = "Action";

**public** **static** **final** String ***APPLICATION\_JSON*** = "application/json";

**public** **static** **final** String ***IMAGE\_ERROR*** = "Unable to retrieve and render the document";

**public** **static** **final** String ***IMAGE\_TYPE*** = "Image";

**public** **static** **final** String ***IMAGE\_STATUS*** = "true";

**public** **static** **final** String ***APPROVED*** = "Approved";

**public** **static** **final** String ***DECLINED*** = "declined";

**public** **static** **final** String ***FIVE\_HUNDRED\_CHAR\_WITHOUT\_SPACES*** = "Thisisafivehundredcharacterslengthcommententeredwithoutanyspaceandthetextareaintowhichthisisgettingaddedshouldhandlethisproperly.Thisisafivehundredcharacterslengthcommententeredwithoutanyspaceandthetextareaintowhichthisisgettingaddedshouldhandlethisproperly.Thisisafivehundredcharacterslengthcommententeredwithoutanyspaceandthetextareaintowhichthisisgettingaddedshouldhandlethisproperly.Thisisafivehundredcharacterslengthcommententeredwithoutanyspaceandthetextareaintowhichthisisgettingaddedshouldhan";

**public** **static** **final** String ***FIVE\_HUNDRED\_CHARS\_WITH\_SPACES*** = "This is 500 character length message with proper spaces in between. This is 500 character length message with proper spaces in between. This is 500 character length message with proper spaces in between. This is 500 character length message with proper spaces in between. This is 500 character length message with proper spaces in between. This is 500 character length message with proper spaces in between. This is 500 character length message with proper spaces in between. This is 500 chara";

**public** **static** **final** String ***FIVE\_THOUSAND\_CHAR\_WITHOUT\_SPACES*** = "5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000CharWithoutSpaces.5000Cq";

**public** **static** **final** String ***FIVE\_THOUSAND\_CHAR\_WITH\_SPACES*** = "This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 charaThis is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is 5000 character length message with proper spaces in between. This is ";

**public** **static** **final** String ***OUTERHTML\_ATTRIBUTE*** = "outerHTML";

// Document Summary constants

**public** **static** **final** String ***UPLOAD\_MODEL\_DESCRIPTION\_DEFAULT\_PLACEHOLDER*** = "Optional";

**public** **static** **final** String ***PHONENUMBER\_TXT*** = "phonenumber";

**public** **static** **final** String ***DOCUMENTS*** = "Documents";

**public** **static** **final** String ***SCOPE*** = "Scope";

**public** **static** **final** String ***DATE\_UPLOADED*** = "Date Uploaded";

**public** **static** **final** String ***PUBLISHED*** = "Published";

**public** **static** **final** String ***POI\_PANEL\_HEADER\_TXT*** = "Verify Proof of Identification";

**public** **static** **final** String ***POI\_PANEL\_TXT*** = "Requester has provided proof of identification. Please review, approve or decline.";

**public** **static** **final** String ***POI\_APPROVED\_TXT*** = "Requester’s proof of identity has been verified and approved. Click to view.";

**public** **static** **final** String ***POI\_DECLINED\_TXT*** = "Requester’s proof of identity has been verified and declined. Click to view.";

**public** **static** **final** String ***REQUEST\_COMPLETED\_HEADER*** = "COMPLETED";

**public** **static** **final** String ***ERROR\_MESSAGE\_LOG\_TEXT*** = "-->Error message is not displayed";

**public** **static** **final** String ***PHONE\_NUMBER\_ERROR\_XPATH*** = "//\*[@formarrayname='phones']//\*[@class='form-group'][%s]";

**public** **static** **final** String ***CARD\_ERROR\_XPATH*** = "//\*[@formarrayname='cards']//\*[@class='form-group'][%s]";

**public** **static** **final** Integer ***INTEGER\_TWO*** = 2;

**public** **static** **final** String ***DOWNLOAD\_FILEPATH*** = File.separator + "src" + File.separator + "main" + File.separator

+ "resources" + File.separator + "config" + File.separator + "TestData" + File.separator + "Downloads";

**public** **static** **final** String ***SCROLL\_BAR\_CLASS*** = "document-publish-border-display";

**public** **static** **final** String ***NO\_SCROLL\_BAR\_CLASS*** = "ordered-list-dash-display";

**public** **static** **final** String ***VALUE*** = "value";

**public** **static** **final** String ***BUSINESS*** = "Business";

**public** **static** **final** String ***PRODUCT\_NAME*** = "prdct\_nam";

**public** **static** **final** String ***REGION*** = "Region";

**public** **static** **final** String ***STATE*** = "State";

**public** **static** **final** String ***DESCENDING*** = "descending";

**public** **static** **final** String ***ORIGINAL\_REQUEST\_SECTION*** = "originalRequestSection";

**public** **static** **final** String ***REGIONS\_COUNTRY\_LIST*** = File.separator + "src" + File.separator + "main" + File.separator

+ "resources" + File.separator + "config" + File.separator + "TestData" + File.separator + "Latest.xls";

**public** **static** **final** String ***FONT\_SIZE\_16PX*** = "16px";

**public** **static** **final** String ***FONT\_FAMILY*** = "\"Mastercard Mark\", -apple-system, system-ui, BlinkMacSystemFont, \"Segoe UI\", Roboto, \"Helvetica Neue\", Arial, sans-serif";

**public** **static** **final** String ***TEXTAREA\_FIRST\_NAME\_XPATH*** = "//textarea[contains(@id,'firstName')]";

**public** **static** **final** String ***TEXTAREA\_LAST\_NAME\_XPATH*** = "//textarea[contains(@id,'lastName')]";

**public** **static** **final** String ***TOOLTIP\_ICON\_XPATH1*** = "(//span[@class='sprite icon-error'])[1]";

**public** **static** **final** String ***TOOLTIP\_TEXT\_XPATH1*** = "(//span[@class='tooltiptext'])[1]";

**public** **static** **final** String ***TOOLTIP\_ICON\_XPATH2*** = "(//span[@class='sprite icon-error'])[2]";

**public** **static** **final** String ***TOOLTIP\_TEXT\_XPATH2*** = "(//span[@class='tooltiptext'])[2]";

**private** UIConstants() {

}

}

**package** com.mastercard.testing.gdp.ui.tests.exception;

**public** **class** GDPUIException **extends** RuntimeException {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** GDPUIException(String message) {

**super**(message);

}

**public** GDPUIException(String message, Exception exp) {

**super**(message, exp);

}

}

**package** com.mastercard.testing.gdp.ui.tests.helper;

**import** com.google.common.base.Function;

**import** com.mastercard.testing.gdp.ui.framework.GDPDriverProviderFactory;

**import** com.mastercard.testing.gdp.ui.tests.constants.UIConstants;

**import** com.mastercard.testing.gdp.ui.tests.exception.GDPUIException;

**import** com.mastercard.testing.gdp.ui.tests.utils.WebDriverUtils;

**import** org.assertj.core.api.Assertions;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.postgresql.Driver;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.stereotype.Component;

**import** java.util.List;

@Component

**public** **class** CustomWaiters {

@Autowired

**private** GDPDriverProviderFactory gdpDriverProviderfactory;

@Autowired

**private** WebDriverUtils webDriverUtils;

**private** **final** Logger logger = LoggerFactory.getLogger(CustomWaiters.**class**);

@Value("${element.finder.wait.timeout\_in\_secs:90}")

**private** **int** elementWaitTime;

**public** **int** getElementWaitTime() {

**return** elementWaitTime;

}

**public** WebDriver getDriver() {

**return** gdpDriverProviderfactory.getDriverProvider().get();

}

**public** **void** waitForFunction(**final** Function function) {

waitForFunction(function, elementWaitTime);

}

**public** **void** waitForFunction(**final** Function function, **int** sec) {

WebDriverWait wait = **new** WebDriverWait(getDriver(), sec);

**try** {

wait.ignoring(Throwable.**class**).until(function);

} **catch** (Exception e) {

logger.debug(e.getMessage());

**throw** **new** GDPUIException(e.getCause().getMessage(), e);

}

}

**public** WebElement getVisibleElement(By locator) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), elementWaitTime);

**return** webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(locator));

}

**public** WebElement getVisibleElement(By locator, **int** timeOutInSeconds) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), timeOutInSeconds);

**return** webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(locator));

}

**public** Boolean waitInvisibility(By by) {

**return** waitInvisibilityByTime(by, elementWaitTime);

}

**public** Boolean waitInvisibilityByTime(By by, **int** timeOutInSeconds) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), timeOutInSeconds);

**return** webDriverWait.until(ExpectedConditions.invisibilityOfElementLocated(by));

}

**public** WebElement waitUntilElementClickable(By locator) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), elementWaitTime);

**return** webDriverWait.until(ExpectedConditions.elementToBeClickable(locator));

}

**public** WebElement waitUntilElementClickable(WebElement element) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), elementWaitTime);

**return** webDriverWait.until(ExpectedConditions.elementToBeClickable(element));

}

**public** WebElement getClickableElement(By by) {

webDriverUtils.waitForAngularToLoad();

WebDriverWait driverWait = **new** WebDriverWait(getDriver(), elementWaitTime);

driverWait.until(ExpectedConditions.visibilityOfElementLocated(by));

**return** driverWait.until(ExpectedConditions.elementToBeClickable(by));

}

**public** Boolean waitUntilElementContainsValue(By by, String value) {

WebDriverWait driverWait = **new** WebDriverWait(getDriver(), elementWaitTime);

**return** driverWait.until(ExpectedConditions.attributeContains(by, UIConstants.***VALUE\_ATTRIBUTE***, value));

}

**public** List<WebElement> getVisibleElements(By locator) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), elementWaitTime);

**return** webDriverWait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(locator));

}

**public** List<WebElement> getVisibleElements(By locator, **int** timeOutInSeconds) {

WebDriverWait webDriverWait = **new** WebDriverWait(getDriver(), timeOutInSeconds);

**return** webDriverWait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(locator));

}

**public** List<WebElement> waitForDropdownValueToAppear(By locator) {

**return** waitForDropdownValueToAppear(locator, elementWaitTime);

}

**public** List<WebElement> waitForDropdownValueToAppear(By locator, **int** timeOutInSeconds) {

WebDriverWait wait = **new** WebDriverWait(getDriver(), timeOutInSeconds);

**return** wait.until(ExpectedConditions.presenceOfNestedElementsLocatedBy(locator, By.tagName("option")));

}

**public** **void** waitForNestedElements(By locator, **int** timeOutInSeconds, **int** numberOfExpectedElements) {

WebDriverWait wait = **new** WebDriverWait(getDriver(), timeOutInSeconds);

wait.until(ExpectedConditions.numberOfElementsToBeMoreThan(locator, numberOfExpectedElements));

}

**public** **void** waitElementSelectionStateToBe(By locator, **boolean** selected) {

WebDriverWait wait = **new** WebDriverWait(getDriver(), elementWaitTime);

wait.until(ExpectedConditions.elementSelectionStateToBe(locator, selected));

}

**public** List<WebElement> getElements(By by) {

**return** getDriver().findElements(by);

}

**public** WebElement getElement(By by) {

WebDriver driver = getDriver();

waitForFunction(function -> {

((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true)", driver.findElement(by));

Assertions.assertThat(driver.findElement(by)).isNotNull();

**return** **true**;

}, elementWaitTime);

**return** driver.findElement(by);

}

**public** **boolean** verifyElementNotPresent(By by) {

**return** getElements(by).isEmpty();

}

**public** **boolean** verifyElementNotVisible(By by) {

**return** getVisibleElements(by, elementWaitTime).isEmpty();

}

**public** **void** waitForAngularToLoad() {

waitForAngularToLoad(elementWaitTime);

}

**public** **void** waitForAngularToLoad(**int** timeOutInSeconds) {

webDriverUtils.waitForAngularToLoad(timeOutInSeconds);

}

**public** **void** waitForAttributeValue(By by, String attribute, String value) {

**new** WebDriverWait(getDriver(), elementWaitTime)

.until(ExpectedConditions.attributeContains(by, attribute, value));

}

**public** WebElement getWebElement(By locator) {

**return** getDriver().findElement(locator);

}

**public** WebElement getPresenceOfElement(By locator) {

WebDriverWait wait = **new** WebDriverWait(getDriver(), elementWaitTime);

**return** wait.until(ExpectedConditions.presenceOfElementLocated(locator));

}

}

**package** com.mastercard.testing.gdp.ui.tests.utils;

**import** java.time.Duration;

**import** java.util.List;

**import** java.util.Set;

**import** java.util.concurrent.TimeUnit;

**import** java.util.stream.Collectors;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.NoSuchElementException;

**import** org.openqa.selenium.StaleElementReferenceException;

**import** org.openqa.selenium.TimeoutException;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.Select;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** com.mastercard.testing.gdp.ui.framework.GDPDriverProviderFactory;

**import** com.mastercard.testing.gdp.ui.tests.constants.UIConstants;

**import** com.mastercard.testing.gdp.ui.tests.helper.CustomWaiters;

@Component

**public** **class** WebDriverUtils {

**protected** **static** Logger *logger* = LoggerFactory.getLogger(WebDriverUtils.**class**);

@Autowired

**private** GDPDriverProviderFactory gdpDriverProviderfactory;

@Autowired

**private** CustomWaiters customWaiters;

@Autowired

**private** FormAppUrls formAppUrls;

**private** WebDriverUtils() {

}

**private** WebDriver getDriver() {

**return** gdpDriverProviderfactory.getDriverProvider().get();

}

**public** JavascriptExecutor getJavascriptExecutor() {

**return** (JavascriptExecutor) getDriver();

}

**public** String getText(WebElement element) {

**return** element.getText();

}

**public** String getAttribute(WebElement element, String attribute) {

**return** element.getAttribute(attribute);

}

**public** String getValueAttributeByJS(WebElement element) {

**return** getJavascriptExecutor().executeScript("return arguments[0].value", element).toString();

}

**public** **void** clearTextUsingJS(WebElement element) {

getJavascriptExecutor().executeScript("arguments[0].value=''", element);

}

**public** **void** clearAndSendTextUsingJavaScriptExecutor(WebElement element, String text) {

getJavascriptExecutor().executeScript("arguments[0].value ='';", element);

getJavascriptExecutor().executeScript("arguments[0].value='" + text + "';", element);

}

**public** **void** click(WebElement element) {

**new** WebDriverWait(getDriver(), customWaiters.getElementWaitTime())

.ignoring(StaleElementReferenceException.**class**).until(ExpectedConditions.elementToBeClickable(element));

element.click();

}

**public** **boolean** isDisplayed(WebElement element) {

**return** element.isDisplayed();

}

**public** **boolean** isEnabled(WebElement element) {

**return** element.isEnabled();

}

**public** **boolean** isSelected(WebElement element) {

**return** element.isSelected();

}

**public** **void** switchToGDPApplications() {

getDriver().manage().timeouts().pageLoadTimeout(60, TimeUnit.SECONDS);

// switch to the latest window.

**for** (String windowHandle : getDriver().getWindowHandles()) {

getDriver().switchTo().window(windowHandle);

}

}

**public** **void** swtichToAppFrame() {

WebDriver driver = getDriver();

WebElement appFrame = driver.findElement(By.id("app-iframe"));

driver.switchTo().frame(appFrame);

}

**public** **void** clickUsingAction(By by) {

WebElement element = getDriver().findElement(by);

clickUsingAction(element);

}

**public** **void** clickUsingAction(WebElement element) {

Actions actions = **new** Actions(getDriver());

actions.moveToElement(element).click().build().perform();

}

**public** **void** mouseOver(WebElement element) {

Actions actions = **new** Actions(getDriver());

actions.moveToElement(element).build().perform();

}

**public** **void** clickWithJavaScriptExecutor(WebElement element) {

getJavascriptExecutor().executeScript("arguments[0].click()", element);

}

**public** **void** sendTextUsingJavaScriptExecutor(WebElement element, String text) {

getJavascriptExecutor().executeScript("arguments[0].value='" + text + "'", element);

}

**public** **void** sendTextUsingActionsClass(WebElement element, String text) {

Actions actions = **new** Actions(getDriver());

actions.moveToElement(element).click().sendKeys(text).build().perform();

}

**public** **void** clear(WebElement element) {

element.clear();

}

**public** **void** sendKeys(WebElement element, String text) {

element.sendKeys(text);

}

**public** **void** selectByVisibleText(WebElement element, String visibleText) {

Select select = **new** Select(element);

select.selectByVisibleText(visibleText);

}

**public** **void** selectByValue(WebElement element, String value) {

Select select = **new** Select(element);

select.selectByValue(value);

}

**public** **void** selectBy(WebElement element, **int** index) {

Select select = **new** Select(element);

select.selectByIndex(index);

}

**public** **void** waitForAngularLoad() {

waitForPageToLoad();

waitForAngularToLoad(customWaiters.getElementWaitTime());

}

**public** **void** waitForAngularToLoad(**int** elementWaitTime) {

String waitForAngularScript = "if (typeof window.getAllAngularTestabilities === \"function\") { var ta = window.getAllAngularTestabilities()[0]; return ta.\_isZoneStable && ta.\_pendingCount == 0;} else { return false;}";

WebDriverWait wait = **new** WebDriverWait(getDriver(), elementWaitTime);

wait.until(driver -> {

Boolean angularIsReadyForTests = (Boolean) getJavascriptExecutor().executeScript(waitForAngularScript);

logger.info("Angular is ready = " + angularIsReadyForTests);

**return** angularIsReadyForTests;

});

}

**public** **void** waitForAngularToFinishAllHttpRequests() {

String angularAllHttpRequestCompletedScript = "return angular.element(document).injector().get('$http').pendingRequests.length === 0;";

WebDriverWait wait = **new** WebDriverWait(getDriver(), 5);

wait.until(d -> {

Boolean requestsIsFinished = (Boolean) ((JavascriptExecutor) d)

.executeScript(angularAllHttpRequestCompletedScript);

logger.info("All http requests is finished = " + requestsIsFinished);

**return** requestsIsFinished;

});

}

**public** **void** forceAngularToRetrieveUserId() {

// This is workaround to go throw Consumer Auth,

// because js make two calls to get user id

// and sometimes response for first request came later than first one

String angularRetrieveCapIdScript = "return angular.element(document.getElementsByName('loginForm')).scope().retrieveCapId();";

getJavascriptExecutor().executeScript(angularRetrieveCapIdScript);

*logger*.info("Force retrieve user id for Consumer Auth");

}

**public** **void** jsClick(WebElement elementToJsClick) {

**new** WebDriverWait(getDriver(), customWaiters.getElementWaitTime())

.ignoring(StaleElementReferenceException.**class**)

.until(ExpectedConditions.elementToBeClickable(elementToJsClick));

getJavascriptExecutor().executeScript(UIConstants.***JS\_EXECUTOR\_TO\_CLICK***, elementToJsClick);

}

**public** **void** jsClickWithoutWait(WebElement elementToJsClick) {

getJavascriptExecutor().executeScript(UIConstants.***JS\_EXECUTOR\_TO\_CLICK***, elementToJsClick);

}

**public** Boolean isDisabled(WebElement element) {

**return** (Boolean) getJavascriptExecutor().executeScript("return arguments[0].disabled", element);

}

**public** **void** navigateBack() {

getDriver().navigate().back();

waitForAngularLoad();

}

**public** **void** clearTextUsingSendKeys(WebElement webElement) {

webElement.click();

webElement.sendKeys(Keys.CONTROL + "a");

webElement.sendKeys(Keys.BACK\_SPACE);

}

**public** **void** replaceTextUsingSendKeys(WebElement webElement, String newValue) {

scrollToTheElement(webElement);

isDisplayed(webElement);

clearTextUsingSendKeys(webElement);

webElement.sendKeys(newValue);

webElement.sendKeys(Keys.TAB);

}

**public** **boolean** isElementAvaliable(By locator, **int** sec) {

WebDriverWait wait = **new** WebDriverWait(getDriver(), sec);

**try** {

wait.until(ExpectedConditions.visibilityOfElementLocated(locator));

*logger*.info("Object is visible on webpage");

} **catch** (TimeoutException e) {

*logger*.info("object located by " + locator.toString() + " is not available on screen", e);

**return** **false**;

}

**return** **true**;

}

**public** **boolean** isElementPresent(By by) {

**try** {

getDriver().findElement(by);

**return** **true**;

} **catch** (NoSuchElementException exception) {

*logger*.info(exception.toString());

**return** **false**;

}

}

**public** List<String> getListOfTextForElements(List<WebElement> elements) {

**return** elements.stream().map(WebElement::getText).collect(Collectors.toList());

}

**public** WebElement getChildElementByXpath(WebElement element, String locator) {

**return** element.findElement(By.xpath(locator));

}

**public** List<WebElement> getChildElementsByXpath(WebElement element, String locator) {

**return** element.findElements(By.xpath(locator));

}

**public** WebElement getChildElementById(WebElement element, String locator) {

**return** element.findElement(By.id(locator));

}

**public** String currentURL() {

**return** getDriver().getCurrentUrl();

}

**public** **void** scrollToTheElement(WebElement element) {

getJavascriptExecutor().executeScript("arguments[0].scrollIntoView(true);", element);

}

**public** **void** scrollToTheTopOfThePage() {

getJavascriptExecutor().executeScript("window.scrolTo(0,-document.body.scrollHeight)");

}

**public** **void** doubliClickUsingAction(WebElement element) {

Actions actions = **new** Actions(getDriver());

actions.moveToElement(element).contextClick(element).build().perform();

}

**public** **void** selectCheckbox(WebElement checkBoxElement) {

**if** (!checkBoxElement.isSelected()) {

jsClick(checkBoxElement);

} **else** {

*logger*.info("Element [" + checkBoxElement + "] is already checked");

}

}

**public** **void** uncheck(WebElement checkboxElement) {

**if** (checkboxElement.isSelected()) {

clickWithJavaScriptExecutor(checkboxElement);

} **else** {

*logger*.info("Element [" + checkboxElement + "] is already unchecked");

}

}

**public** String getTagName(WebElement element) {

**return** element.getTagName();

}

**public** **void** switchToWindowByIndex(**int** windowIndex) {

Object[] windowHandles = getDriver().getWindowHandles().toArray();

**if** (windowIndex >= 0) {

getDriver().switchTo().window((String) windowHandles[windowIndex]);

*logger*.info("Window selected based on Index" + windowIndex);

getDriver().manage().timeouts().pageLoadTimeout(60, TimeUnit.SECONDS);

} **else** {

*logger*.info("Window index should be '>=0'");

}

}

**public** **void** closeWindow() {

getDriver().close();

}

**public** String getBackGroundColor(WebElement element, String color) {

**return** element.getCssValue(color);

}

**public** Boolean verifyPageScroll() {

String execScript = "return document.documentElement.scrollHeight>document.documentElement.clientHeight;";

**return** (Boolean) (getJavascriptExecutor().executeScript(execScript));

}

**public** Boolean verifyPageScrollImage(String fileType) {

String execScript = "return document.querySelector('.idv" + fileType

+ "div').scrollHeight>document.querySelector('.idv" + fileType + "div').clientHeight;";

**return** (Boolean) (getJavascriptExecutor().executeScript(execScript));

}

**public** **int** getOpenWindowSize() {

Set<String> openWindow = getDriver().getWindowHandles();

**return** openWindow.size();

}

**public** List<WebElement> getOptions(WebElement webElement) {

Select dropdown = **new** Select(webElement);

**return** dropdown.getOptions();

}

**public** **void** waitForPageToLoad() {

String pageLoadStatus;

**do** {

pageLoadStatus = (String) getJavascriptExecutor().executeScript("return document.readyState");

*logger*.info("still Loading...");

} **while** (!"Complete".equals(pageLoadStatus));

*logger*.info("Page is loaded");

}

**public** WebElement getFirstSelectedElement(WebElement element) {

Select select = **new** Select(element);

**return** select.getFirstSelectedOption();

}

**public** **void** refreshThePage() {

gdpDriverProviderfactory.getDriverProvider().get().navigate().refresh();

customWaiters.waitForAngularToLoad();

}

**public** **void** navigateForward() {

getDriver().navigate().refresh();

}

**public** **void** openLinkAndMaximizeTheWindow(String brokenVerificationLink) {

getDriver().manage().window().maximize();

getDriver().navigate().to(brokenVerificationLink);

}

**public** **boolean** isClickable(WebElement elem) {

**try** {

WebDriverWait wait = **new** WebDriverWait(customWaiters.getDriver(), customWaiters.getElementWaitTime());

wait.until(ExpectedConditions.elementToBeClickable(elem));

**return** **true**;

} **catch** (Exception e) {

*logger*.error(e.getMessage());

**return** **false**;

}

}

**public** **boolean** isElementNotAvailable(WebElement elem, **int** sec) {

**try** {

**new** WebDriverWait(getDriver(), sec).until(ExpectedConditions.invisibilityOfAllElements(elem));

**return** **true**;

} **catch** (Exception e) {

*logger*.info("object is still available in page ", e);

**return** **false**;

}

**return** **true**;

}

**public** **boolean** isElementAvailable(WebElement elem, **int** sec) {

**try** {

WebDriverWait wait = **new** WebDriverWait(customWaiters.getDriver(), sec);

wait.until(ExpectedConditions.visibilityOf(elem));

} **catch** (TimeoutException e) {

*logger*.info("object is not available in page ", e);

**return** **false**;

}

**return** **true**;

}

**public** **boolean** isElementNotAvailableByLocator(By locator, **int** sec) {

**try** {

WebDriverWait wait = **new** WebDriverWait(customWaiters.getDriver(), sec);

wait.until(ExpectedConditions.invisibilityOfElementLocated(locator));

} **catch** (TimeoutException e) {

*logger*.info("locator is still available in page ", e);

**return** **false**;

}

**return** **true**;

}

**public** Boolean isChecked(WebElement element) {

**return** (Boolean) getJavascriptExecutor().executeAsyncScript("return arguments[0].checked;", element);

}

**public** **void** waitForElementToSelected(WebElement element, Boolean selectionState) {

**new** WebDriverWait(getDriver(), customWaiters.getElementWaitTime())

.until(ExpectedConditions.elementSelectionStateToBe(element, selectionState));

}

**public** String getFirstSelectedDropDownValue(WebElement webElement) {

**return** getFirstSelectedElement(webElement).getText();

}

**public** **boolean** isDropdownSuportsMultiSelect(WebElement webElement) {

Select dropDown = **new** Select(webElement);

**return** dropDown.isMultiple();

}

**public** List<String> getOptionsInDropdown(WebElement webElement) {

Select dropdown = **new** Select(webElement);

dropdown.getOptions().stream().map(ele -> ele.getText()).collect(Collectors.toList());

}

**public** **void** maxmizeBrowser() {

customWaiters.getDriver().manage().window().maximize();

}

**public** String getBeforePseudoContent() {

String script = "return window.getComputedStyle( document.querySelector('input[type=radio]:checked+label.radio-label'),':before').getPropertyValue('color')";

JavascriptExecutor js = (JavascriptExecutor) getDriver();

**return** (String) js.executeScript(script);

}

}

//

// Source code recreated from a .class file by IntelliJ IDEA

// (powered by Fernflower decompiler)

//

**package** java.nio.charset;

**public** **final** **class** StandardCharsets {

**public** **static** **final** Charset ***US\_ASCII*** = Charset.forName("US-ASCII");

**public** **static** **final** Charset ***ISO\_8859\_1*** = Charset.forName("ISO-8859-1");

**public** **static** **final** Charset ***UTF\_8*** = Charset.forName("UTF-8");

**public** **static** **final** Charset ***UTF\_16BE*** = Charset.forName("UTF-16BE");

**public** **static** **final** Charset ***UTF\_16LE*** = Charset.forName("UTF-16LE");

**public** **static** **final** Charset ***UTF\_16*** = Charset.forName("UTF-16");

**private** StandardCharsets() {

**throw** **new** AssertionError("No java.nio.charset.StandardCharsets instances for you!");

}

}

**package** org.jbehave.web.selenium;

**import** org.openqa.selenium.WebDriver;

**public** **interface** WebDriverProvider {

WebDriver get();

**void** initialize();

**boolean** saveScreenshotTo(String path);

**void** end();

}

**package** restAssured;

**import** java.nio.charset.StandardCharsets;

**public** **final** **class** Constants {

**public** **static** **final** String ***COMMA\_SEPARATOR*** = "\\s\*,\\s\*";

**public** **static** **final** String ***SERVICE\_FUNCTION\_CODE*** = "serviceFunctionCode";

**public** **static** **final** String ***REQUEST\_ID*** = "requestId";

**public** **static** **final** String ***ERROR\_CODE*** = "errorCode";

**public** **static** **final** String ***REQUEST\_STATUS\_CODE*** = "requestStatusCode";

**public** **static** **final** String ***RESPONSE\_CODE*** = "responseCode";

**public** **static** **final** String ***ENCRYPTED\_CODE*** = "encryptedKey";

**public** **static** **final** String ***EMAIL\_PARAMETERS*** = "emailParameters";

**public** **static** **final** String ***URL\_SEPARATOR*** = ":";

**public** **static** **final** String ***MERGE\_DATA*** = "mergeData";

**public** **static** **final** String ***URL*** = "url";

**public** **static** **final** String ***PRODUCT\_NAME*** = "productName";

**public** **static** **final** String ***SERVICE\_FUNCTION\_NAME*** = "serviceFunctionName";

**public** **static** **final** String ***APPLICATION\_DATA*** = "applicationData";

**public** **static** **final** String ***MESSAGE\_DATA*** = "messageData";

**public** **static** **final** String ***SITE\_ID*** = "siteID";

**public** **static** **final** String ***CAMPAIGN\_ID*** = "campaignID";

**public** **static** **final** String ***TEMPLATE*** = "template";

**public** **static** **final** String ***SUBJECT*** = "subject";

**public** **static** **final** String ***RECIPIENT\_EMAIL*** = "recipientEmail";

**public** **static** **final** String ***SENDER\_EMAIL*** = "senderEmail";

**public** **static** **final** String ***GDP\_WEB\_MANUALRESPONSE\_REQUEST\_ID*** = "/gdp-web/manualresponse?requestId=";

**public** **static** **final** String ***FUNCTION\_CODE*** = "&functionCode=";

**public** **static** **final** String ***ENCRYPTED\_KEY*** = "&encryptedKey=";

**public** **static** **final** String ***ENCRYPTED\_KEY\_SEPARATOR*** = "-";

**public** **static** **final** String ***EMAIL\_TEXT\_REGEX*** = "^[\\w!#$%&'\*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'\*+/=?`{|}~^-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$";

**public** **static** **final** String ***YYYY\_MM\_DD*** = "yyyy-MM-dd";

**public** **static** **final** **int** ***RADIX*** = 16;

**public** **static** **final** **int** ***SIGNUM*** = 1;

**public** **static** **final** String ***APPLICATION\_JSON*** = "application/json";

**public** **static** **final** String ***HEADER\_NAME\_CONTENT\_TYPE*** = "Content-Type";

**public** **static** **final** String ***CORRELATION\_ID*** = "Correlation-ID";

**public** **static** **final** String ***HEADER\_NAME\_ACCCEPT*** = "Accept";

**public** **static** **final** String ***ZONE\_ID*** = "Canada/Central";

**public** **static** **final** String ***PARAMETER\_SEPARATOR\_AMPERSAND*** = "&";

**public** **static** **final** String ***PARAMETER\_SEPARATOR\_QUESTION\_MARK*** = "?";

**public** **static** **final** String ***HEADER\_SEPARATOR\_EQUAL\_TO*** = "=";

**public** **static** **final** String ***BUSINESS\_REVIEW\_DATA\_BEAN\_HEADER*** = "BusinessReviewDataBean";

**public** **static** **final** String ***CONSUMER\_REVIEW\_DATA\_BEAN\_HEADER*** = "ConsumerReviewDataBean";

**public** **static** **final** String ***INPUT\_TYPE*** = "inputType";

**public** **static** **final** String ***UNIQUE\_KEY\_SEPARATOR*** = "\_";

**public** **static** **final** String ***OK*** = "OK";

**public** **static** **final** **int** ***API\_RESPONSE\_OK*** = 200;

**public** **static** **final** String ***PRIVACY\_REQUEST\_STATUS*** = "I";

**public** **static** **final** String ***NULL*** = "null";

**public** **static** **final** String ***CALLING\_THE\_API\_WITH\_THE\_RESOURCE\_PATH*** = "Calling the API with the resource path : ";

**public** **static** **final** **int** ***DELTA\_IN\_MILLISECONDS*** = 40000;

**public** **static** **final** **int** ***MILLISECONDS\_FOR\_3\_SECONDS*** = 3000;

**public** **static** **final** String ***PATH\_PARAMETER\_DELIMITER*** = "/";

**public** **static** **final** String ***PRIVACY\_REQUEST\_ID\_PATH\_PARAMETER*** = "?privacyRequestId=";

**public** **static** **final** String ***SERVICE\_FUNCTION\_CODE\_PATH\_PARAMETER*** = "&serviceFunctionCode=";

**public** **static** **final** String ***ALGORITHM*** = "MD5";

**public** **static** **final** String ***SALT\_STRING*** = "GDPSECURESALTKEY";

**public** **static** **final** String ***RQST\_ENTRY*** = "RQST\_ENTRY";

**public** **static** **final** String ***RESP\_RECVD*** = "RESP\_RECVD";

**public** **static** **final** String ***RQST\_SENT*** = "RQST\_SENT";

**public** **static** **final** String ***PRDCT\_CD*** = "prdct\_cd";

**public** **static** **final** String ***PRDCT\_PRVCY\_RQST\_ID*** = "prdct\_prvcy\_rqst\_id";

**public** **static** **final** String ***RESPONSE\_RECORD\_NAME\_SEPARATOR*** = "#";

**public** **static** **final** String ***SENDER\_NAME*** = "senderName";

**public** **static** **final** String ***REPLY\_TO*** = "replyTo";

**public** **static** **final** String ***RETURN\_PATH*** = "returnPath";

**public** **static** **final** String ***CONTACT\_INFO*** = "contactInfo";

**public** **static** **final** String ***RECIPIENT\_FIRSTNAME*** = "recipientFirstname";

**public** **static** **final** String ***RECIPIENT\_LASTNAME*** = "recipientLastname";

**public** **static** **final** String ***PROCESSING\_PARAMETERS*** = "processingParameters";

**public** **static** **final** String ***SITE\_ID\_VALUE*** = "822";

**public** **static** **final** String ***M5\_BVM7*** = "1-1M5BVM7";

**public** **static** **final** String ***YES*** = "Y";

**public** **static** **final** String ***REPORTING*** = "reporting";

**public** **static** **final** String ***EMAIL\_DELIVERY\_CHANNEL*** = "emailDeliveryChannel";

**public** **static** **final** String ***MASTER\_CARD*** = "MASTER\_CARD";

**public** **static** **final** String ***CLIENT\_MESSAGE\_ID*** = "clientMessageID";

**public** **static** **final** String ***RQST\_PBLSH*** = "RQST\_PBLSH";

**public** **static** **final** String ***CHAR\_SET*** = StandardCharsets.***UTF\_8***.name();

**public** **static** **final** String ***DATE\_SUBMITTED*** = "dateSubmitted";

**public** **static** **final** String ***PARAM\_SEPARATOR*** = "/";

**public** **static** **final** String ***PRODUCT\_STATUS\_CHANGE*** = "PRD\_ST\_CHG";

**public** **static** **final** String ***AUD\_DESC*** = "aud\_desc";

**public** **static** **final** String ***PRODUCT\_REQUEST\_STATUS\_CHANGED\_TO*** = "Product Request status changed to";

**public** **static** **final** String ***EMAIL\_CURRENT\_TIME\_REGEX*** = "MM/dd/yy hh:mm:ss";

**public** **static** **final** String ***ERROR\_EMAIL\_CURRENT\_TIME*** = "createdDate";

**public** **static** **final** String ***DUE\_DAYS*** = "dueDays";

**public** **static** **final** String ***DUE\_DAYS\_MANUAL\_VALUE*** = "5";

**public** **static** **final** String ***DUE\_DAYS\_ERROR\_MESSAGE\_VALUE*** = "2";

**public** **static** **final** String ***REPLY\_EMAIL*** = "mailTo";

**public** **static** **final** String ***REPLY\_EMAIL\_VALUE*** = "biz-ops@mailbox.com";

**public** **static** **final** String ***PHONE\_NUMBER\_VALUE*** = "1-800-000-0000";

**public** **static** **final** String ***PHONE\_NUMBER*** = "callTo";

**public** **static** **final** String ***MANUAL\_RESPONSE\_SUBJECT\_LINE*** = "ACTION REQUIRED: GDPR Individual Data Access Request#";

**public** **static** **final** String ***ERROR\_RESPONSE\_SUBJECT\_LINE*** = "ACTION REQUIRED: Error encountered validating response for Request#";

**public** **static** **final** String ***ERROR\_TEMPLATE*** = "822/822\_GDPR\_error-email-template.xsl";// Needs to be modified

**public** **static** **final** String ***MANUAL\_TEMPLATE*** = "822/822\_GDPR\_en\_US.xsl";

**public** **static** **final** String ***ERROR\_DESCRIPTION*** = "errorData";

**public** **static** **final** String ***ERROR\_KEY*** = "errorKey";

**public** **static** **final** String ***ERROR\_VALUE*** = "errorValue";

**public** **static** **final** String ***COMPLETED\_DATE*** = "dateCompleted";

**public** **static** **final** String ***STLOUIS\_TIMEZONE*** = "UTC-06:00";

**public** **static** **final** String ***IN\_PROGRESS*** = "I";

**public** **static** **final** String ***COMPLETED*** = "C";

**public** **static** **final** String ***REQ\_TYPE\_DELETE*** = "D";

**public** **static** **final** String ***REQ\_TYPE\_UPDATE*** = "U";

**public** **static** **final** String ***REQ\_TYPE\_VIEW*** = "V";

**public** **static** **final** String ***SERV\_INTG\_TYPE\_PUBSUB*** = "P";

**public** **static** **final** String ***SERV\_INTG\_TYPE\_MANUAL*** = "M";

**public** **static** **final** String ***SERV\_INTG\_TYPE\_API*** = "A";

**public** **static** **final** String ***BUSINESS\_CNTXT\_B2B*** = "B2B";

**public** **static** **final** String ***BUSINESS\_CNTXT\_B2C*** = "B2C";

**public** **static** **final** String ***BUSINESS\_CNTXT\_B2E*** = "B2E";

**public** **static** **final** String ***DOMAIN\_BUSINESS*** = "Business";

**public** **static** **final** String ***DOMAIN\_CONSUMER*** = "Consumer";

**public** **static** **final** String ***INSERT\_INTO\_GDP\_OWNER\_RESP\_REC*** = "INSERT INTO resp\_rec(\r\n";

**public** **static** **final** String ***RESP\_REC\_ID\_SERV\_FUNC\_PRVCY\_RQST\_ID\_PRDCT\_PRVCY\_RQST\_ID\_RESP\_REC\_NAM\_RESP\_DATA\_TXT*** = " resp\_rec\_id, serv\_func\_prvcy\_rqst\_id, prdct\_prvcy\_rqst\_id, resp\_rec\_nam, resp\_data\_txt, \r\n";

**public** **static** **final** String ***RESP\_REC\_STAT\_CD\_RESP\_RVW\_TS\_DSCLSR\_SW\_DEL\_SW\_DEL\_PRDCT\_RQST\_ID\_HOLD\_RLSE\_DT*** = " resp\_rec\_stat\_cd, resp\_rvw\_ts, dsclsr\_sw, del\_sw, del\_prdct\_rqst\_id, hold\_rlse\_dt)\r\n";

**public** **static** **final** String ***RESPONSE\_RECORD\_ID*** = "responseRecordId";

**public** **static** **final** String ***PRODUCTS*** = "products";

**public** **static** **final** String ***PRODUCT\_PRIVACY\_REQUEST\_ID*** = "productPrivacyRequestId";

**public** **static** **final** String ***SERVICE\_FUNCTIONS*** = "serviceFunctions";

**public** **static** **final** String ***SERVICE\_FUNCTION\_PRIVACY\_REQUEST\_ID*** = "serviceFunctionPrivacyRequestId";

**public** **static** **final** String ***RECORDS*** = "records";

**public** **static** **final** String ***RECORD\_ID*** = "recordId";

**public** **static** **final** String ***PRIVACY\_REQUEST\_ID*** = "privacyRequestId";

**public** **static** **final** String ***PRODUCT\_IDS*** = "productIds";

**public** **static** **final** String ***FILTER\_BY*** = "filterBy";

**public** **static** **final** String ***LOCALE*** = "locale";

**public** **static** **final** String ***APP\_AUTH\_USERNAME\_GDPUSER*** = "app.auth.username.gdpuser";

**public** **static** **final** String ***APP\_AUTH\_KEY*** = "app.auth.password";

**public** **static** **final** String ***DATE\_FORMAT\_US\_CST*** = "dd MMM yyyy 'at' hh:mm a 'US CST'";

**public** **static** **final** String ***RESP\_REC\_STAT\_CD\_RESP\_RVW\_TS\_DSCLSR\_SW\_DEL\_SW\_DEL\_PRDCT\_RQST\_ID\_HOLD\_RLSE\_DT\_DEL\_PRVCY\_RQST\_ID*** = " resp\_rec\_stat\_cd, resp\_rvw\_ts, dsclsr\_sw, del\_sw, del\_prdct\_rqst\_id, hold\_rlse\_dt, del\_prvcy\_rqst\_id)\r\n";

**public** **static** **final** String ***DEFAULT\_LANG*** = "en-us";

**public** **static** **final** String ***APOSTROPHE*** = "'";

**public** **static** **final** String ***COMMA*** = ",";

**public** **static** **final** String ***GET*** = "GET";

**public** **static** **final** String ***DELETE*** = "DELETE";

**public** **static** **final** String ***POST*** = "POST";

**public** **static** **final** String ***AWAITING\_RESPONSE\_STATUS*** = "A";

**public** **static** **final** String ***AUTHORIZATION*** = "Authorization";

**public** **static** **final** String ***PRODUCT\_PRIVACY\_REQ\_ID*** = "productPrivacyReqId";

**public** **static** **final** String ***RESP\_REC\_STAT\_CD\_RESP\_RVW\_TS\_DSCLSR\_SW\_DEL\_SW\_DEL\_PRDCT\_RQST\_ID\_HOLD\_RLSE\_DT\_DEL\_PRV\_RQST\_ID*** = " resp\_rec\_stat\_cd, resp\_rvw\_ts, dsclsr\_sw, del\_sw, del\_prdct\_rqst\_id, hold\_rlse\_dt, del\_serv\_func\_rqst\_id, del\_prvcy\_rqst\_id)\r\n";

**public** **static** **final** String ***EMAIL\_SITE\_ID*** = "822";

**public** **static** **final** String ***EMAIL\_CAMPAIGN\_ID*** = "1-1M5BVM7";

**public** **static** **final** String ***SERVICE\_FUNCTION\_PRIVACY\_REQUEST\_ID\_COLUMN*** = "serv\_func\_prvcy\_rqst\_id";

**public** **static** **final** String ***REVIEWED\_TIME*** = "reviewdTime";

**public** **static** **final** String ***NUM\_DAYS\_IN\_QUEUE*** = "noOfDaysInQueue";

**public** **static** **final** String ***OFFSET*** = "offset";

**public** **static** **final** String ***LIMIT*** = "limit";

**public** **static** **final** String ***SORT\_ORDER*** = "sortorder";

**public** **static** **final** String ***COLUMN\_NAME*** = "columnname";

**public** **static** **final** String ***PRIVACY\_REQUEST\_STATUS\_READY\_TO\_PROCESS*** = "AP";

**public** **static** **final** String ***PRIVACY\_REQUEST\_STATUS\_PENDING\_REVIEW*** = "PR";

**public** **static** **final** String ***AUDIT\_HIST\_AUD\_TYPE\_CD\_REQ\_UPDATE*** = "RQST\_UPDT";

**public** **static** **final** String ***EMPTY\_REQUEST\_BODY*** = "{}";

**public** **static** **final** String ***UPDATE\_REQUEST\_TYPE*** = "U";

**public** **static** **final** String ***RQST\_DELETE\_ALL*** = "RQST\_DA";

**public** **static** **final** String ***REQUEST\_TYPE\_VIEW*** = "View";

**public** **static** **final** String ***REQUEST\_TYPE\_DELETE\_ALL*** = "Delete All";

**public** **static** **final** String ***PRIVACY\_NOTE*** = "privacyNotice";

**public** **static** **final** String ***PRIVACY\_NOTE\_VALUE*** = "https://www.mastercard.us/en-us/about-mastercard/what-we-do/privacy.html";

**public** **static** **final** String ***GDP\_B2B\_WEB\_MANUALRESPONSE\_REQUEST\_ID*** = "/gdp-b2b-web/manualresponse?requestId=";

**public** **static** **final** String ***INTERNAL*** = "Internal";

**public** **static** **final** String ***EXTERNAL*** = "External";

**public** **static** **final** String ***DISPATCH\_TYPE*** = "dispatchType";

**public** **static** **final** String ***DECLINED\_DATE*** = "declinedTimeStamp";

**public** **static** **final** String ***SPACE*** = " ";

**public** **static** **final** String ***SEARCH\_KEY*** = "searchKey";

**public** **static** **final** String ***FALSE*** = "false";

**public** **static** **final** String ***TRUE*** = "true";

**public** **static** **final** String ***DB\_TIME\_REGEX*** = "yyyy-MM-dd HH:mm:ss";

**public** **static** **final** String ***TIME\_ZONE*** = "America/Chicago";

**public** **static** **final** String ***DOUBLE\_ENCODER\_OF\_SLASH*** = "%252F";

**public** **static** **final** String ***FAILED\_API\_RESPONSE\_IS*** = "Failed API response is: ";

**public** **static** **final** String ***DATA\_ACCESS\_SYSTEM*** = "Data Access System";

**public** **static** **final** String ***GDP\_SUPER\_USER\_NAME*** = "GDP Super User";

**public** **static** **final** String ***GDP\_SUPER\_USER\_ID*** = "gdpsuperuser";

**public** **static** **final** String ***COLON\_SEPARATOR*** = "\\s\*:\\s\*";

**public** **static** **final** String ***COLON*** = ":";

**public** **static** **final** String ***PRVCY\_REQ\_IDENTITY\_PROOF\_ID*** = "privacyRequestIdentityProofId";

**public** **static** **final** String ***REVEIEWER\_USER\_ID*** = "reviewerUserId";

**public** **static** **final** String ***LOG\_APPROVE\_FLAG*** = "Verifying is Approve flag";

**public** **static** **final** String ***APPROVE*** = "approve";

**public** **static** **final** String ***DECL\_REASON\_COMMENT*** = "declineReasonComment";

**public** **static** **final** String ***PUBLISHED\_DOCUMENTS*** = "Published:\n";

**public** **static** **final** String ***INTERNAL\_COMMENT\_TEXT\_IS\_INCORRECT*** = "Internal Comment Text is incorrect";

**public** **static** **final** String ***NEW\_LINE\_SEPARATOR*** = "\n";

**public** **static** **final** String ***DOCUMENT\_DOWNLOAD\_PATH*** = "./src/main/resources/config/TestData/documentDownload/%s";

**public** **static** **final** String ***EXTERNAL\_COMMENTS*** = "Message to Requester on review";

**public** **static** **final** String ***INTERNAL\_COMMENTS*** = "Add Internal Comments on review";

// Document upload constants

**public** **static** **final** String ***UPLOAD\_DESCRIPTION*** = "uploadDescription";

**public** **static** **final** String ***FILE*** = "file";

**public** **static** **final** String ***DOCUMENT\_SCOPE*** = "documentScope";

**public** **static** **final** String ***APPRV\_REASON\_COMMENT*** = "Additional Description: Dear Requester - We have verified your identity and approved. You may go ahead and access the documents.";

**public** **static** **final** String ***APPRV\_REASON\_COMMENT\_2*** = "Kind Regards.";

**public** **static** **final** String ***LABEL\_IDEN\_TYPE*** = "Identification Type: ";

**public** **static** **final** String ***COMMENTS*** = "comments";

**public** **static** **final** String ***ID\_PROOF\_TYPE\_CODE*** = "idProofTypeCode";

**public** **static** **final** String ***OTHER\_ID\_PROOF\_NAME*** = "otherIdProofName";

**public** **static** **final** String ***FIRST\_NAME*** = "firstName";

**public** **static** **final** String ***LAST\_NAME*** = "lastName";

**public** **static** **final** String ***LAST\_FOUR\_DIGITS\_ID\_PROOF*** = "lastFourDigitsOfIdProof";

**public** **static** **final** String ***ID\_PROOF\_DOCUMENTS*** = "idProofDocuments";

**public** **static** **final** String ***PRIVACY\_REQUEST\_DOCUMENT\_UPLOAD\_ID\_TEXT*** = "privacyRequestDocumentUploadId";

**public** **static** **final** String ***COLUMN\_NAME\_TXT*** = "columnName";

**public** **static** **final** String ***SORT\_ORDER\_TXT*** = "sortOrder";

**public** **static** **final** String ***AKER\_PROXY\_REQUEST\_TIMEOUT*** = "request-timeout";

**public** **static** **final** String ***REQUEST\_TYPE*** = "requestType";

**public** **static** **final** String ***FIRST\_NAME\_COLUMN*** = "first\_nam";

**public** **static** **final** String ***FIRSTNAME\_XSL\_KEY*** = "body.salutation.firstName";

**public** **static** **final** String ***REQUESTID\_XSL\_KEY*** = "body.requestId";

**public** **static** **final** String ***REQUESTURL\_XSL\_KEY*** = "body.requestURL";

**public** **static** **final** String ***MOCK\_EMAILID\_XSL\_KEY*** = "mockEmailId";

**public** **static** **final** String ***SEARCH\_REQUEST\_ID*** = "<REQUESTID>";

**public** **static** **final** String ***DELETE\_REQUEST\_ID*** = "DeleteReqId";

**public** **static** **final** String ***VIEW\_REQUEST\_ID*** = "ViewReqId";

**public** **static** **final** String ***EVENT\_TYPE\_RQST\_SBMT*** = "RQST\_SBMT";

**public** **static** **final** String ***EVENT\_TYPE\_RQST\_DELAY*** = "RQST\_DELAY";

**public** **static** **final** String ***EVENT\_TYPE\_EMAIL\_VER*** = "EMAIL\_VER";

**public** **static** **final** String ***EVENT\_TYPE\_EMAIL\_RMNDR*** = "EMAIL\_RMNDR";

**public** **static** **final** String ***DATEOFREQUEST\_XSL\_KEY*** = "body.dateOfRequest";

**public** **static** **final** String ***POI\_PURGE\_AUD\_DESC*** = "POI Documents purge Action completed for Privacyrequest: %s.";

**public** **static** **final** String ***REQUEST\_PURGE\_AUD\_DESC*** = "Privacy request purging";

**public** **static** **final** String ***PHONE\_NUMBERS*** = "phoneNumbers";

**public** **static** **final** **int** ***NO\_CONTENT\_STATUS\_CODE*** = 204;

**public** **static** **final** **int** ***TIMED\_OUT\_STATUS\_CODE*** = 408;

**public** **static** **final** String ***PRIVACY\_REQUEST\_APPROVALS\_BATCH*** = "PrivacyRequestApprovalsBatch";

**public** **static** **final** **int** ***SUCCESS\_STATUS\_CODE*** = 200;

**public** **static** **final** String ***ISMANUAL*** = "isManual";

**public** **static** **final** String ***VERIFIED*** = "V";

**public** **static** **final** String ***CURING\_ATTEMPTS*** = "curingAttemptsCount";

**public** **static** **final** String ***IN\_CURING\_DAYS*** = "daysInCuring";

**public** **static** **final** String ***UPLOADFILEPATH*** = "./src/main/resources/config/TestData/";

**public** **static** **final** String ***SEARCH\_KEYS\_CARD\_DETAILS\_FIRST\_CARD\_NUMBER*** = "searchKeys.cardDetails[0].cardNumber";

**public** **static** **final** String ***CARD\_DETAILS\_FIRST\_CARD\_NUMBER*** = "cardDetails[0].cardNumber";

**public** **static** **final** String ***EXPORT\_CARD\_NUMBERS\_LIST*** = "products[0].serviceFunctions[0].records[2].recordData.cardDetails.findAll{it}.cardNumber";

**public** **static** **final** String ***REQUEST\_ID\_REPLACER*** = "<REQUEST\_ID>";

**public** **static** **final** String ***AUD\_CODE\_REPLACER*** = "<AUD\_CODE>";

**public** **static** **final** String ***POA*** = "POA";

**public** **static** **final** String ***RWS*** = "RWS";

**public** **static** **final** String ***PAS\_WD*** = "password";

**public** **static** **final** String ***PRODUCT\_VERIFICATION\_STATUS\_CODE*** = "prdct\_ver\_stat\_cd";

**public** **static** **final** String ***AGRMT\_TYPE*** = "agrmt\_type";

**public** **static** **final** String ***ADDR\_TYPE*** = "addr\_type";

**public** **static** **final** String ***DOC\_TYPE*** = "doc\_type";

**public** **static** **final** String ***EXPRT\_ACTN*** = "EXPRT\_ACTN";

**public** **static** **final** String ***FILE\_REPO\_ID*** = "fileRepoId";

**public** **static** **final** String ***USER\_ID*** = "userId";

**public** **static** **final** String ***DATE*** = "date";

**public** **static** **final** String ***BULK\_UPLOAD\_FILE\_PATH*** = "./src/main/resources/config/TestData/%s";

**public** **static** **final** String ***TILDE\_SEPERATOR*** = "~";

**public** **static** **final** String ***BULK\_REQUEST\_UPLOAD\_ID*** = "bulkRequestUploadId";

**public** **static** **final** String ***BULK\_REQUEST\_DEL\_DRAFT*** = "BULK\_RQST\_DEL\_DRAFT";

**private** Constants() {

}

}

**package** com.mastercard.testing.gdp.api.aspect;

**import** com.mastercard.testing.gdp.api.exception.GDPAPIException;

**import** org.apache.commons.logging.Log;

**import** org.apache.commons.logging.LogFactory;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.Around;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Pointcut;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.List;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

@Aspect

@Component

**public** **class** TestDataProviderAspect **implements** TestDataExpressionResolver, StepArgumentProvider {

**public** **static** **final** Pattern ***TEST\_DATA\_EXPRESSION*** = Pattern.*compile*("([A-z])\\{(.\*)\\}$");

**private** **static** **final** Log ***LOG*** = LogFactory.*getLog*(TestDataProviderAspect.**class**);

@Autowired

**private** List<TestDataProvider> testDataProviders;

**private** ThreadLocal<Object[]> lastArguments = **new** ThreadLocal<>();

@Pointcut(value = "@annotation(org.jbehave.core.annotations.Given)")

**public** **void** givenMethod() {

// This method is to be implemented if required.

}

@Pointcut(value = "@annotation(org.jbehave.core.annotations.When)")

**public** **void** whenMethod() {

// This method is to be implemented if required.

}

@Pointcut(value = "@annotation(org.jbehave.core.annotations.Then)")

**public** **void** thenMethod() {

// This method is to be implemented if required.

}

@Around("givenMethod() || whenMethod() || thenMethod()")

**public** Object methodInterceptor(ProceedingJoinPoint jp) **throws** Throwable {

String methodName = jp.toShortString();

***LOG***.info("Running Method = " + methodName);

Object[] arguments = jp.getArgs();

**for** (**int** i = 0; i < arguments.length; i++) {

Object argument = arguments[i];

**if** (argument != **null**) {

**if** (argument **instanceof** String) {

arguments[i] = resolveExpression((String) argument);

} **else** **if** (argument **instanceof** Collection && ((Collection) argument).size() == 1) {

arguments[i] = setArgumentsForCollection(arguments, i, (Collection) argument);

}

}

}

lastArguments.set(arguments);

**try** {

**return** jp.proceed(arguments);

} **catch** (Exception ex) {

**throw** **new** GDPAPIException(ex.getCause().getMessage(), ex);

}

}

**private** Object setArgumentsForCollection(Object[] arguments, **int** i, Collection argument) {

Object collectionElement = argument.iterator().next();

**if** (collectionElement **instanceof** String) {

arguments[i] = resolveListExpression((String) collectionElement);

}

**return** arguments[i];

}

@Override

**public** String resolveExpression(String expression) {

StringBuilder result = **new** StringBuilder();

**for** (String subExpression : expression.split("\\+")) {

Matcher matcher = ***TEST\_DATA\_EXPRESSION***.matcher(subExpression);

String prefix = "default";

**if** (matcher.find()) {

prefix = matcher.group(1);

subExpression = matcher.group(2);

}

result.append(getDataProvider(prefix).getValue(subExpression));

}

**return** result.toString();

}

@Override

**public** List<String> resolveListExpression(String expression) {

List<String> result = **new** ArrayList<>();

**for** (String subExpression : expression.split("\\+")) {

Matcher matcher = ***TEST\_DATA\_EXPRESSION***.matcher(subExpression);

String prefix = "default";

**if** (matcher.find()) {

prefix = matcher.group(1);

subExpression = matcher.group(2);

}

result.addAll(getDataProvider(prefix).getList(subExpression));

}

**return** result;

}

@Override

**public** Object[] getLastStepArguments() {

Object[] arguments = lastArguments.get();

lastArguments.remove();

**return** arguments;

}

**private** TestDataProvider getDataProvider(String prefix) {

**for** (TestDataProvider testDataProvider : testDataProviders) {

**if** (testDataProvider.support(prefix)) {

**return** testDataProvider;

}

}

**throw** **new** IllegalArgumentException("Unsupported expression type '" + prefix + "{...}");

}

}

**package** com.mastercard.testing.gdp.api.configuration;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan

**public** **class** BDDBaseRestToolsConfig {

@Bean

**public** ObjectMapper objectMapper() {

**return** **new** ObjectMapper();

}

}

**package** com.mastercard.testing.gdp.api.configuration;

**import** org.apache.http.impl.client.CloseableHttpClient;

**import** org.springframework.beans.factory.FactoryBean;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.annotation.Import;

@Configuration

@Import({ BDDBaseRestToolsConfig.**class** })

**public** **class** BDDRestToolsConfig {

@Bean

**public** FactoryBean<CloseableHttpClient> getFactory() {

**return** **new** HttpClientFactory();

}

}

**package** com.mastercard.testing.gdp.api.stories;

**import** com.epam.reportportal.jbehave.ReportPortalFormat;

**import** com.github.valfirst.jbehave.junit.monitoring.JUnitReportingRunner;

**import** com.mastercard.quality.engineering.mtaf.jbehave.configuration.MastercardJBehaveStories;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.RallyStoryReporter;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.alm.ALMService;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.alm.rally.RallyIntegrationConfiguration;

**import** com.mastercard.quality.engineering.mtaf.jbehave.reporters.alm.rally.RallyService;

**import** com.mastercard.testing.gdp.api.configuration.RestConfiguration;

**import** com.mastercard.testing.gdp.api.reporting.FailedStoriesReporter;

**import** com.mastercard.testing.gdp.api.reporting.RPFreemarkerViewGenerator;

**import** com.mastercard.testing.gdp.api.reporting.GdpRallyStoryReporter;

**import** org.apache.commons.lang3.StringUtils;

**import** org.jbehave.core.configuration.Configuration;

**import** org.jbehave.core.configuration.MostUsefulConfiguration;

**import** org.jbehave.core.embedder.StoryControls;

**import** org.jbehave.core.io.CodeLocations;

**import** org.jbehave.core.io.LoadFromClasspath;

**import** org.jbehave.core.io.StoryFinder;

**import** org.jbehave.core.reporters.CrossReference;

**import** org.jbehave.core.reporters.Format;

**import** org.jbehave.core.reporters.StoryReporter;

**import** org.jbehave.core.reporters.StoryReporterBuilder;

**import** org.jbehave.core.steps.ParameterControls;

**import** org.junit.runner.RunWith;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** java.net.URISyntaxException;

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

@RunWith(JUnitReportingRunner.**class**)

**public** **class** GdpRestStories **extends** MastercardJBehaveStories {

**public** GdpRestStories() {

context = getAnnotatedApplicationContext();

}

@Override

**public** ApplicationContext getAnnotatedApplicationContext() {

**return** **new** AnnotationConfigApplicationContext(RestConfiguration.**class**);

}

@Override

**public** List<String> storyPaths() {

List<String> paths = **null**;

String storyName = System.*getProperty*("storyName");

**if** (StringUtils.*isNotEmpty*(storyName)) {

**if** (storyName.endsWith(".story") && !storyName.contains(",")) {

paths = (**new** StoryFinder()).findPaths(CodeLocations.*codeLocationFromClass*(**this**.getClass()),

"\*\*/stories/\*\*/" + storyName, "");

} **else** **if** (storyName.endsWith(".story") && storyName.contains(",")) {

List<String> stories = Arrays.*asList*(storyName.split(","));

List<String> includes = **new** ArrayList<>();

stories.stream().forEach(story -> includes.add("\*\*/stories/\*\*/" + story));

paths = (**new** StoryFinder()).findPaths(CodeLocations.*codeLocationFromClass*(**this**.getClass()), includes,

**new** ArrayList<>());

} **else** {

paths = (**new** StoryFinder()).findPaths(CodeLocations.*codeLocationFromClass*(**this**.getClass()),

Arrays.*asList*("\*\*/stories/folder".replace("folder", storyName + "/\*\*/\*.story")),

Arrays.*asList*("\*\*/excluded/\*\*/\*.story", "\*\*/\*.txt"));

}

} **else** {

paths = (**new** StoryFinder()).findPaths(CodeLocations.*codeLocationFromClass*(**this**.getClass()),

"\*\*/stories/\*\*/\*.story", "\*\*/stories/exclude/\*\*/\*.story");

}

**return** paths;

}

@Override

**public** Configuration configuration() {

Configuration configuration;

configuration = **new** MostUsefulConfiguration()

.useStoryControls(

**new** StoryControls().doResetStateBeforeScenario(**true**).doSkipScenariosAfterFailure(**false**))

.useStoryLoader(**new** LoadFromClasspath(**this**.getClass()))

.useStoryReporterBuilder(**this**.getStoryReporterBuilder())

.useViewGenerator(**new** RPFreemarkerViewGenerator())

.useParameterControls((**new** ParameterControls()).useDelimiterNamedParameters(**true**));

**return** configuration;

}

**private** **boolean** isRPEnabled() {

**return** (System.*getProperty*("rp.enable") != **null** && Boolean.*valueOf*(System.*getProperty*("rp.enable")));

}

@Override

**protected** Format[] storyFormat() {

Format[] defaultFormats = **super**.storyFormat();

**if** (isRPEnabled()) {

List<Format> formatList = **new** ArrayList<>(Arrays.*asList*(defaultFormats));

formatList.add(ReportPortalFormat.***INSTANCE***);

**return** formatList.toArray(**new** Format[formatList.size()]);

} **else** {

**return** defaultFormats;

}

}

@Override

**protected** RallyStoryReporter getRallyReport() {

RallyIntegrationConfiguration rallyIntegrationConfiguration = context

.getBean(RallyIntegrationConfiguration.**class**);

String testSetDescription = getTestSetDescription(rallyIntegrationConfiguration);

rallyIntegrationConfiguration.withTestSetDesc(testSetDescription);

ALMService rallyALMService = **null**;

**try** {

rallyALMService = **new** RallyService(rallyIntegrationConfiguration);

} **catch** (URISyntaxException exception) {

*LOG*.error("URI Syntax Error for Rally ALM Service", exception);

}

**return** **new** GdpRallyStoryReporter(rallyALMService);

}

@Override

**protected** StoryReporterBuilder getStoryReporterBuilder() {

StoryReporterBuilder storyReporterBuilder;

storyReporterBuilder = ((**new** StoryReporterBuilder()).withFormats(**this**.storyFormat())

.withReporters(**this**.getReporters()).withFailureTraceCompression(**true**)

.withCrossReference(**new** CrossReference()));

**return** storyReporterBuilder;

}

**private** String getRallyTestSetDescription() {

String testSetName = System.*getProperty*("rally.testSetDesc");

**if** (testSetName != **null**)

**return** testSetName;

**else**

**return** "";

}

**private** String getTestSetDescription(RallyIntegrationConfiguration rallyIntegrationConfiguration) {

String testSetDescription = "Regression";

String runtimeTestSetDescription = **this**.getRallyTestSetDescription();

**if** (!runtimeTestSetDescription.isEmpty()) {

testSetDescription = runtimeTestSetDescription;

} **else** **if** (!rallyIntegrationConfiguration.getTestSetDesc().isEmpty()) {

testSetDescription = rallyIntegrationConfiguration.getTestSetDesc();

}

**if** (!testSetDescription.contains("\_")) {

String dateFormat = "MM-dd-yyyy";

SimpleDateFormat simpleDateFormat = **new** SimpleDateFormat(dateFormat);

String currentDateTime = simpleDateFormat.format(**new** Date());

testSetDescription = String.*format*("%s\_%s", testSetDescription, currentDateTime);

}

*LOG*.info("Test Set Name: " + testSetDescription);

**return** testSetDescription;

}

@Override

**protected** StoryReporter[] getReporters() {

StoryReporter[] storyReporters = **super**.getReporters();

List<StoryReporter> storyReportersList = **new** ArrayList<>(Arrays.*asList*(storyReporters));

storyReportersList.add(**new** FailedStoriesReporter());

**return** storyReportersList.toArray(**new** StoryReporter[storyReportersList.size()]);

}

}

**package** com.mastercard.testing.gdp.restassured.utils;

**import** org.apache.poi.ss.usermodel.DataFormatter;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.xssf.usermodel.XSSFWorkbook;

**import** org.springframework.stereotype.Component;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

@Component

**public** **final** **class** ExcelReader {

**private** ExcelReader() {

}

**public** **static** Map<String, String> getExcelData(String filePath) **throws** IOException {

Map<String, String> excelValidations = **new** HashMap<>();

DataFormatter dataFormatter = **new** DataFormatter();

InputStream excelFileToRead = **new** FileInputStream(filePath);

XSSFWorkbook xssfWorkBook = **new** XSSFWorkbook(excelFileToRead);

Sheet sheet = xssfWorkBook.getSheetAt(0);

Iterator<Row> rows = sheet.rowIterator();

rows.next();

**while** (rows.hasNext()) {

Row currentRow = rows.next();

String rowNumber = dataFormatter.formatCellValue(currentRow.getCell(0));

String errorDescription = dataFormatter.formatCellValue(currentRow.getCell(1));

excelValidations.put(rowNumber, errorDescription);

}

**return** excelValidations;

}

}

package com.mastercard.testing.gdp.restassured.utils;

import com.mastercard.testing.gdp.api.domain.Context;

import com.mastercard.testing.gdp.restassured.domain.ConsumerRequest;

import com.mastercard.testing.gdp.restassured.domain.User;

import com.mastercard.testing.gdp.restassured.domain.UserLogin;

import io.restassured.path.json.JsonPath;

import io.restassured.specification.RequestSpecification;

import org.apache.commons.io.FileUtils;

import org.apache.commons.io.IOUtils;

import org.apache.commons.io.filefilter.WildcardFileFilter;

import org.apache.commons.lang3.StringUtils;

import org.jbehave.core.annotations.Named;

import org.jetbrains.annotations.NotNull;

import java.io.\*;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.\*;

import java.util.zip.ZipEntry;

import java.util.zip.ZipFile;

import static com.mastercard.testing.gdp.api.helper.constants.Constants.COMMA;

public final class StepUtils {

private StepUtils() {

}

public static UserLogin getUserLogin(@Named("$userName") String userName) {

if (StringUtils.equals(userName, "duplicateUser")) {

return User.getDuplicateUserDetails();

} else if (StringUtils.equals(userName, "gdpUser") || StringUtils.equals(userName, "gdpuser")) {

return User.getGdpUserDetails();

} else if (StringUtils.equals(userName, "testUser1") || StringUtils.equals(userName, "testuser1")) {

return User.getTestUser1Details();

} else if (StringUtils.equals(userName, "newRequestUser1")) {

return User.getNewRequestUser1Details();

} else if (StringUtils.equals(userName, "newRequestUser2")) {

return User.getNewRequestUser2Details();

} else if (StringUtils.equals(userName, "hotjukes")) {

return User.getHotJukesDetails();

} else {

return User.getCreatedUserDetails(userName);

}

}

public static ConsumerRequest prepareRequest(String userName, RequestSpecification spec) {

return ConsumerRequest.builder().specification(spec).userLogin(StepUtils.getUserLogin(userName)).build();

}

public static ConsumerRequest prepareRequestWithBody(String requestBody, String userName,

RequestSpecification spec) {

return ConsumerRequest.builder().specification(spec).userLogin(StepUtils.getUserLogin(userName))

.requestBody(requestBody).build();

}

public static boolean isB2CRequest(String requestType) {

return StringUtils.equals(Context.B2C.getValue(), requestType);

}

public static String[] getCardNumbers(String cardDetails) {

return cardDetails.split(COMMA);

}

public static JsonPath getResponseAsJsonPath(String response) throws IOException {

decodeResponseToFileFormat(response, getExportZipFilePath());

unzipContentsOfZipFile();

String contentAsString = getFileContentAsString("\*.json", getTempFilePath());

FileUtils.cleanDirectory(new File(getTempFilePath()));

return new JsonPath(contentAsString);

}

public static Map<String, String> getResponseAsJson(String response) throws IOException {

Map<String, String> excelData = new HashMap<>();

decodeResponseToFileFormat(response, getErrorFilePath());

excelData = ExcelReader.getExcelData(getErrorFilePath());

Files.deleteIfExists(Paths.get(getErrorFilePath()));

return excelData;

}

@NotNull

private static String getFileContentAsString(String fileExtension, String filePath) throws IOException {

File dir = new File(filePath);

FileFilter fileFilter = new WildcardFileFilter(fileExtension);

File[] files = dir.listFiles(fileFilter);

Optional<File> first = Arrays.stream(files).findFirst();

return new String(Files.readAllBytes(Paths.get(first.map(File::getPath).orElse(""))), StandardCharsets.UTF\_8);

}

private static void unzipContentsOfZipFile() throws IOException {

try (ZipFile zipFile = new ZipFile(getExportZipFilePath())) {

Enumeration<? extends ZipEntry> entries = zipFile.entries();

while (entries.hasMoreElements()) {

ZipEntry entry = entries.nextElement();

File entryDestination = new File(getTempFilePath(), entry.getName());

if (!entry.isDirectory()) {

entryDestination.getParentFile().mkdirs();

try (InputStream in = zipFile.getInputStream(entry);

OutputStream out = new FileOutputStream(entryDestination)) {

IOUtils.copy(in, out);

}

}

}

}

}

private static void decodeResponseToFileFormat(String response, String filePath) throws IOException {

FileUtils.cleanDirectory(new File(getTempFilePath()));

try (FileOutputStream os = new FileOutputStream(filePath)) {

os.write(Base64.getDecoder().decode(response));

}

}

@NotNull

private static String getExportZipFilePath() {

return getTempFilePath() + File.separator + "export-file.zip";

}

@NotNull

private static String getErrorFilePath() {

return getTempFilePath() + File.separator + "error.xlsx";

}

@NotNull

private static String getTempFilePath() {

return Paths.get("src", "main", "resources", "config", "TestData", "tempData").toAbsolutePath().toString();

}

}

**package** com.mastercard.testing.gdp.api.configuration;

**import** org.apache.http.impl.client.CloseableHttpClient;

**import** org.apache.http.impl.client.HttpClients;

**public** **class** HttpClientFactory **implements** MCClientFactory {

**public** Class<?> getObjectType() {

**return** CloseableHttpClient.**class**;

}

@Override

**public** **boolean** isSingleton() {

**return** **true**;

}

**public** CloseableHttpClient getObject() {

**return** HttpClients.*createDefault*();

}

}

**package** com.mastercard.testing.gdp.api.configuration;

**import** org.apache.http.impl.client.CloseableHttpClient;

**import** org.springframework.beans.factory.FactoryBean;

**public** **interface** MCClientFactory **extends** FactoryBean<CloseableHttpClient> {

CloseableHttpClient getObject();

}

**package** com.mastercard.testing.gdp.api.configuration;

**import** com.mastercard.quality.engineering.mtaf.jbehave.configuration.spring.MTAFJBehaveToolsConfiguration;

**import** org.springframework.context.annotation.\*;

@Configuration

@ComponentScan(basePackages = { "com.mastercard.testing.gdp" })

@PropertySource("config/${env}/application.properties")

@EnableAspectJAutoProxy

@Import({ MTAFJBehaveToolsConfiguration.**class**, BDDRestToolsConfig.**class** })

**public** **class** RestConfiguration {

}

**package** com.mastercard.testing.gdp.api.configuration;

**public** **class** RestServiceException **extends** RuntimeException {

**private** **static** **final** **long** ***serialVersionUID*** = -942235246006018987L;

**public** RestServiceException() {

}

**public** RestServiceException(String message, Throwable cause) {

**super**(message, cause);

}

}

<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.mastercard.gdp.testing</groupId>  
 <artifactId>gdp-mtaf-api</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
 <name>gdp-mtaf-api</name>  
 <url>http://maven.apache.org</url>  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <jbehave.core.version>4.6</jbehave.core.version>  
 <jbehave.execution.threads>1</jbehave.execution.threads>  
 <lombok.version>1.18.10</lombok.version>  
 <aspectj.version>1.9.4</aspectj.version>  
 <mtaf.version>19.3.2.0-SNAPSHOT</mtaf.version>  
 <spring.version>5.1.8.RELEASE</spring.version>  
 <metaFilters></metaFilters>  
 <axon-sdk.version>2.0.5</axon-sdk.version>  
 <json-path.version>2.4.0</json-path.version>  
 <rest-assured.version>4.2.0</rest-assured.version>  
 <oracle.version>12.2.0.1</oracle.version>  
 <commons-io.version>2.8.0</commons-io.version>  
 <sonar.exclusions>pom.xml,  
 src/main/java/com/mastercard/testing/gdp/api/domain/\*\*/\*,  
 src/main/java/com/mastercard/testing/gdp/api/aspect/TestDataProviderAspect.java  
 </sonar.exclusions>  
 <sonar.coverage.exclusions>  
 src/\*\*/\*  
 </sonar.coverage.exclusions>  
 </properties>  
 <dependencies>  
 <!-- AspectJ -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 <version>${aspectj.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>${aspectj.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>com.mastercard.quality.engineering</groupId>  
 <artifactId>mtaf-jbehave-tools</artifactId>  
 <version>${mtaf.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.jbehave</groupId>  
 <artifactId>jbehave-core</artifactId>  
 <version>${jbehave.core.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <version>5.5.2</version>  
 </dependency>  
 <dependency>  
 <groupId>com.jayway.jsonpath</groupId>  
 <artifactId>json-path</artifactId>  
 <version>${json-path.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>com.jayway.jsonpath</groupId>  
 <artifactId>json-path-assert</artifactId>  
 <version>${json-path.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.assertj</groupId>  
 <artifactId>assertj-core</artifactId>  
 <version>3.14.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>${lombok.version}</version>  
 <scope>provided</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 <version>3.9</version>  
 </dependency>  
 <!-- DB Utils -->  
 <dependency>  
 <groupId>commons-dbutils</groupId>  
 <artifactId>commons-dbutils</artifactId>  
 <version>1.7</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-dbcp2</artifactId>  
 <version>2.7.0</version>  
 </dependency>  
 <dependency>  
 <groupId>com.ibatis</groupId>  
 <artifactId>ibatis2-common</artifactId>  
 <version>2.1.7.597</version>  
 </dependency>  
 <dependency>  
 <groupId>org.jline</groupId>  
 <artifactId>jline</artifactId>  
 <version>3.13.2</version>  
 </dependency>  
 <!-- <dependency> <groupId>postgresql</groupId> <artifactId>postgresql</artifactId>  
 <version>9.4.1208-jdbc42-atlassian-hosted</version> </dependency> -->  
 <dependency>  
 <groupId>org.postgresql</groupId>  
 <artifactId>postgresql</artifactId>  
 <version>42.2.9</version>  
 <scope>provided</scope>  
 </dependency>  
 <dependency>  
 <groupId>com.mastercard.apie.axon</groupId>  
 <artifactId>axon-sdk</artifactId>  
 <version>${axon-sdk.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.json</groupId>  
 <artifactId>json</artifactId>  
 <version>20180813</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <!-- RP Portal -->  
 <dependency>  
 <groupId>com.epam.reportportal</groupId>  
 <artifactId>agent-java-jbehave</artifactId>  
 <version>5.0.0-BETA-4</version>  
 </dependency>  
 <dependency>  
 <groupId>com.epam.reportportal</groupId>  
 <artifactId>httpclient-repacked</artifactId>  
 <version>1.0.2</version>  
 </dependency>  
 <dependency>  
 <groupId>com.epam.reportportal</groupId>  
 <artifactId>client-java-core</artifactId>  
 <version>2.7.1</version>  
 </dependency>  
 <dependency>  
 <groupId>com.epam.reportportal</groupId>  
 <artifactId>logger-java-log4j</artifactId>  
 <version>2.6.1</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/com.google.guava/guava -->  
 <dependency>  
 <groupId>com.google.guava</groupId>  
 <artifactId>guava</artifactId>  
 <version>28.1-jre</version>  
 </dependency>  
 <dependency>  
 <groupId>io.rest-assured</groupId>  
 <artifactId>rest-assured</artifactId>  
 <version>${rest-assured.version}</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.rest-assured</groupId>  
 <artifactId>rest-assured</artifactId>  
 <version>${rest-assured.version}</version>  
 <scope>compile</scope>  
 </dependency>  
 <dependency>  
 <groupId>commons-io</groupId>  
 <artifactId>commons-io</artifactId>  
 <version>${commons-io.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>com.oracle</groupId>  
 <artifactId>ojdbc8</artifactId>  
 <version>${oracle.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.poi</groupId>  
 <artifactId>poi-ooxml</artifactId>  
 <version>3.9</version>  
 </dependency>  
 <dependency>  
 <groupId>com.googlecode.json-simple</groupId>  
 <artifactId>json-simple</artifactId>  
 <version>1.1.1</version>  
 </dependency>  
 <dependency>  
 <groupId>com.github.javafaker</groupId>  
 <artifactId>javafaker</artifactId>  
 <version>1.0.2</version>  
 </dependency>  
  
 </dependencies>  
 <build>  
 <resources>  
 <resource>  
 <directory>${basedir}/src/main/resources</directory>  
 <filtering>false</filtering>  
 <excludes>  
 <exclude>reportportal.properties</exclude>  
 </excludes>  
 </resource>  
 <resource>  
 <directory>${basedir}/src/main/resources</directory>  
 <filtering>true</filtering>  
 <includes>  
 <include>reportportal.properties</include>  
 </includes>  
 </resource>  
 </resources>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <!-- <version>3.5</version> -->  
 <configuration>  
 <compilerVersion>1.8</compilerVersion>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <configuration>  
 <excludes>  
 <!--DO NOT ADD ANY FILE TO ARTIFACT AS IT IS NO WHERE USED -->  
 <exclude>\*\*/\*.\*</exclude>  
 </excludes>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.jbehave</groupId>  
 <artifactId>jbehave-maven-plugin</artifactId>  
 <version>${jbehave.core.version}</version>  
 <executions>  
 <execution>  
 <id>run-stories-as-embeddables</id>  
 <phase>integration-test</phase>  
 <configuration>  
 <storyTimeoutInSecs>500</storyTimeoutInSecs>  
 <threads>${jbehave.execution.threads}</threads>  
 <includes>  
 <include>\*\*/GdpRestStories.java</include>  
 </includes>  
 <metaFilters>  
 <metaFilter>${metaFilters}</metaFilter>  
 </metaFilters>  
 <systemProperties>  
 <property>  
 <name>java.awt.headless</name>  
 <value>true</value>  
 </property>  
 </systemProperties>  
 <ignoreFailureInStories>true</ignoreFailureInStories>  
 <ignoreFailureInView>false</ignoreFailureInView>  
 </configuration>  
 <goals>  
 <goal>unpack-view-resources</goal>  
 <goal>run-stories-as-embeddables</goal>  
 </goals>  
 </execution>  
 </executions>  
 <dependencies>  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>1.7.5</version>  
 </dependency>  
  
 </dependencies>  
 </plugin>  
 </plugins>  
 </build>  
</project>