

```
rest/resttemplate/DBServiceRestTemplate.java - Spring Tool Suite 4

RestConnectionUtil.java DBServiceRestTemplate.java
1 package com.citigroup.usrao.accountopening.rest.resttemplate;
2
3 import java.net.URI;
4
56
57 @Component
58 public class DBServiceRestTemplate {
59
60     private static Logger logger = LoggerFactory.getLogger(DBServiceRestTemplate.class.getName());
61
62     @Autowired
63     RestTemplate restTemplate;
64
65     @Value("${service.name.db.service}")
66     private String onBoardingService;
67
68     @Value("${service.url.db.service}")
69     private String onBoardingServiceUrl;
70
71     @Autowired
72     private RestConnectionUtil connectionUtils;
73
74     @Value("${pcf.environment}")
75     private String serviceEnv;
76
77
78     @HystrixCommand(commandKey = "hystrixCommandKeyAcapsId", threadPoolKey = "hystrixThreadPoolAcapsId", fallbackMethod = "getAcapsIDFallback")
79     public String getAcapsID(String uuid, String businessCode, String countryCode, String channelId, String applicationID) {
80
81         String response = "";
82         try {
83             HttpEntity<String> requestEntity = new HttpEntity<>(CommonUtility.getHttpHeaders(uuid, businessCode, countryCode, channelId, null, null));
84             logger.debug("Request Entity: {} ", requestEntity);
85             URI serviceURI = null;
86             Map<String, String> paramMap = new HashMap<String, String>();
87             paramMap.put("applicationId", applicationID);
88             serviceURI = connectionUtils.getParamURI(onBoardingService, "/USRTL/DBServices/getAcapsId/{applicationId}", paramMap);
89             logger.info("DBServiceRestTemplate URI: {} ", serviceURI);
90             if (serviceURI != null) {
91                 ResponseEntity<String> result = restTemplate.exchange(serviceURI.toString(), HttpMethod.GET, requestEntity, String.class);
92                 response = result.getBody();
93                 logger.debug("ResponseEntity: {} ", response);
94             }
95         } catch (Exception ex) {
96             CommonUtility.LogException(ex, "DBServiceRestTemplate.getAcapsID");
97         }
98         return response;
99     }
100
101     public String getAcapsIDFallback(String uuid, String businessCode, String countryCode, String channelId, String applicationID) {
102         String response = "";
103         logger.error("Hystrix circuit breaker has tripped in DBServiceRestTemplate.getAcapsID()");
104         return response;
105     }
106 }

Markers Properties Servers Snippets Console Search JUnit Coverage History Call Hierarchy Boot Dashboard Debug

RestConnectionUtil.java DBServiceRestTemplate.java
1 package com.citigroup.usrao.accountopening.rest.resttemplate;
2
3 import java.net.URI;
4
23
24 @Service
25 public class RestConnectionUtil {
26
27     private static Logger LOG = LoggerFactory.getLogger(RestConnectionUtil.class.getName());
28
29     @Autowired
30     DiscoveryClient discoveryClient;
31
32     @Value("${spring.profiles.active}")
33     String activeProfile;
34
35     @Value("${resttemplate.httpconnecttimeout:50000}")
36     int timeout;
37
38     @Value("#{null}")
39     private Random random;
40
41     public URI getURI(String serviceName, String endPoint) throws ServiceNotFoundException {
42
43         String sharedServiceUrl = getServiceURL(serviceName);
44
45         UriComponentsBuilder builder = UriComponentsBuilder.fromHttpUrl(sharedServiceUrl + endPoint);
46
47         LOG.info("{} ***** serviceName URL {} URLEncode: {}", serviceName, builder.toUriString(),
48             builder.build().encode().toUri());
49
50         URI uri = builder.build().encode().toUri();
51         LOG.info("URI: {} ", uri);
52         return uri;
53     }
54
55     public URI getParamURI(String serviceName, String endPoint, Map<String, String> paramMap)
56         throws ServiceNotFoundException {
57         String sharedServiceUrl = getServiceURL(serviceName);
58
59         UriComponentsBuilder builder = UriComponentsBuilder.fromHttpUrl(sharedServiceUrl + endPoint);
60
61         LOG.info("{} ***** serviceName URL {} URLEncode: {}", serviceName, builder.toUriString(),
62             builder.build().encode().toUri());
63
64         URI uri = builder.buildAndExpand(paramMap).encode().toUri();
65         LOG.info("URI: {} ", uri);
66         return uri;
67     }
68
69     public String getServiceURL(String serviceName) throws ServiceNotFoundException {
70
71         LOG.debug("Eureka Service Name = {}", serviceName);
72         String serviceURL = "";
73         // TODO: Add logic to get service URL from Eureka
74     }
75 }
```



```
RestConnectionUtil.java DBServiceRestTemplate.java
68
69 public String getServiceURL(String serviceName) throws ServiceNotFoundException {
70
71     LOG.debug("Eureka Service Name = {}", serviceName);
72     String serviceURL = "";
73     if (activeProfile.contains("LOCAL")) {
74         if (serviceName.equals("OGN-C-OnboardingMongoDB-UsRetail"))
75             serviceURL = "http://localhost:7071";
76         else if (serviceName.equals("RTL-PDM-D-CheckbookWS"))
77             serviceURL = "http://rtl-pdm-d-checkbookws-dev2.cfapps-gti-dev.nam.nsroot.net";
78         else if (serviceName.equals("OGN-D-AccountOpening-UsRetail"))
79             serviceURL = "http://xlg-fdn-p-psg-uat3.cfapps-gti-dev.nam.nsroot.net";
80         else if (serviceName.equals("RTL-OGN-C-ONBOARDINGODB"))
81             serviceURL = "http://xlg-fdn-p-psg-dev2.cfapps-gti-dev.nam.nsroot.net";
82         else if (serviceName.equals("OGN-D-CustomerRefDataMgt-KYC"))
83             serviceURL = "http://localhost:7051";
84         else if (serviceName.equals("RTL-DM-D-CustomerForms"))
85             serviceURL = "http://localhost:7074";
86         else
87             serviceURL = "http://xlg-fdn-p-psg-dev2.cfapps-gti-dev.nam.nsroot.net";
88     } else {
89         try {
90             if (this.discoveryClient != null) {
91                 List<ServiceInstance> instances = this.discoveryClient.getInstances(serviceName);
92                 LOG.debug("All service instances = {}", instances);
93                 if (instances != null && instances.isEmpty()) {
94                     ServiceInstance instance = instances.get(random.nextInt(instances.size()));
95                     if (instance != null) {
96                         URI uri = instance.getUri();
97                         if (uri != null) {
98                             serviceURL = uri.toString();
99                             LOG.info("Service url is : {}", serviceURL);
100                         }
101                     }
102                 } else {
103                     throw new ServiceNotFoundException("Service not found: " + serviceName);
104                 }
105             } else {
106                 LOG.warn("Discovery Client is null in getServiceURL");
107             }
108         } catch (ServiceNotFoundException snfe) {
109             CommonUtility.LogException(snfe, "RestConnectionUtil.getServiceURL");
110             throw snfe;
111         } catch (Exception e) {
112             CommonUtility.LogException(e, "RestConnectionUtil.getServiceURL");
113             LOG.info(serviceURL);
114         }
115     }
116     return serviceURL;
117 }
118
119 public ClientHttpRequestFactory clientHttpRequestFactory() {
120     ClientHttpRequestFactory clientHttpRequestFactory = new HttpComponentsClientHttpRequestFactory();
121     clientHttpRequestFactory.setConnectTimeout(timeout);
122     return clientHttpRequestFactory;
123 }
124
125 @PostConstruct
126 public void initialiseRestConnectionUtil() {
127     random = new Random();
128 }
129 }
130
```

Members calling 'getParamURI(String, String, Map<String, String>)' - in workspace

File	Line	Call
URI - com.citigroup.usrao.accountopening.rest.resttemplate	311	netParamURI(onboardingService, "AJSRT/DBServices/netAcctn/initialisation")



```
JSRAccountOpeningAPIController.java - Spring Tool Suite 4
RestConnectionUtil.java DBServiceRestTemplate.java USRAccountOpeningAPI.java USRAccountOpeningAPIController.java ApplicationIDService.java

278 }
279
280 @Override
281 public ResponseEntity<CustomerSearchResponseDTO> searchCustomer(@ApiParam(value = "The client_id generated during consumer onboarding", required = false) @Request
282 @ApiParam(value = "This is the OAuth Token generated by the APIM Session.", required = false) @RequestHeader(value = "Authorization", required = false) Str
283 @ApiParam(value = "Unique 128 bit random UUID generated uniquely for every request", required = false) @RequestHeader(value = "uid", required = false) Str
284 @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode, @RequestHeader(value = "CountryCode", required = false) Str
285 @ApiParam(value = "Customer details to be retrieved", required = true) @Valid @RequestBody CustomerSearchRequestDTO customerSearchRequest) {
286
287     CustomerSearchResponseDTO customerSearchResponse = customerSearchService.searchCustomer(customerSearchRequest);
288
289     return new ResponseEntity<>(customerSearchResponse, HttpStatus.OK);
290 }
291
292 @Override
293 public ResponseEntity<GetApplicationIdResponseDTO> getApplicationId(@ApiParam(value = "The client_id generated during consumer onboarding", required = false)
294 @RequestHeader(value = "client_id", required = false) String clientId,
295 @ApiParam(value = "This is the OAuth Token generated by the APIM Session.", required = false) @RequestHeader(value = "Authorization", required = false) Str
296 String authorization, @ApiParam(value = "HTTP Accept-Language header", required = false) @RequestHeader(value = "Accept", required = false) String acceptLanguage,
297
298 @ApiParam(value = "Unique 128 bit random UUID generated uniquely for every request", required = false) @RequestHeader(value = "uid", required = false)
299 String uid, @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
300 @RequestHeader(value = "CountryCode", required = false, defaultValue = "US") String countryCode, @RequestHeader(value = "channelId", required = false) Str
301 defaultHeader = "RTLA0") String channelId, @ApiParam(value = "Content-Types that are acceptable for the response", required = false)
302 @RequestHeader(value = "Content-Type", required = false) String contentType, @RequestParam(value = "channelType", required = false) String channelType,
303 @RequestParam(value = "avokaTrackingID", required = false) String avokaTrackingID, @RequestParam(value = "userRole", required = false) String userRole) {
304
305     GetApplicationIdResponseDTO applicationIdResponse = applicationIDService.getApplicationId(uid, businessCode, countryCode, channelId, channelType,
306 avokaTrackingID, userRole);
307
308     return new ResponseEntity<>(applicationIdResponse, HttpStatus.OK);
309 }
310
311 @Override
312 public ResponseEntity<GetBranchDetailsResponseDTO> getBranchDetailsPost(@ApiParam(value = "The client_id generated during consumer onboarding", required = false) {
313 @ApiParam(value = "This is the OAuth Token generated by the APIM Session.", required = false) @RequestHeader(value = "Authorization", required = false) Str
314 @ApiParam(value = "Unique 128 bit random UUID generated uniquely for every request", required = false) @RequestHeader(value = "uid", required = false) Str
315 @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode, @RequestHeader(value = "CountryCode", required = false) Str
316 @ApiParam(value = "getBranchDetails Request", required = true) @Valid @RequestBody GetBranchDetailsRequestDTO branchChannelRequest) {
317
318     GetBranchDetailsResponseDTO response = branchDetailsService.getBranchDetails(uid, businessCode, countryCode, channelId, branchChannelRequest);
319
320     return new ResponseEntity<>(response, HttpStatus.OK);
321 }
322
323 @Override
324 public ResponseEntity<PostalCodeDetailsResponseDTO> getPostalCodeDetails(String clientId, String authorization, String acceptLanguage, String uid, String busines
325 PostalCodeDetailsResponseDTO response = postalCodeAndGoverningStateDetailsService.getPostalCodeDetailsResponse(uid, businessCode, countryCode, channelId, pos
326
327     return new ResponseEntity<>(response, HttpStatus.OK);
328 }
329
330 public ResponseEntity<Boolean> savePendSession(String clientId, String authorization, String acceptLanguage, String uid,
331 String contentType, String businessCode, String countryCode, String channelId, PendSessionDTO pendRequest) {
332
333     return new ResponseEntity<>(true, HttpStatus.OK);
334 }
335
336 }
337
338 }
339
340 }
341
342 }
343
344 }
345
346 }
347
348 }
349
350 }
351
352 }
353
354 }
355
356 }
357
358 }
359
360 }
361
362 }
363
364 }
365
366 }
367
368 }
369
370 }
371
372 }
373
374 }
375
376 }
377
378 }
379
380 }
381
382 }
383
384 }
385
386 }
387
388 }
389
390 }
391
392 }
393
394 }
395
396 }
397
398 }
399
400 }
401
402 }
403
404 }
405
406 }
407
408 }
409
410 }
411
412 }
413
414 }
415
416 }
417
418 }
419
420 }
421
422 }
423
424 }
425
426 }
427
428 }
429
430 }
431
432 }
433
434 }
435
436 }
437
438 }
439
440 }
441
442 }
443
444 }
445
446 }
447
448 }
449
450 }
451
452 }
453
454 }
455
456 }
457
458 }
459
460 }
461
462 }
463
464 }
465
466 }
467
468 }
469
470 }
471
472 }
473
474 }
475
476 }
477
478 }
479
480 }
481
482 }
483
484 }
485
486 }
487
488 }
489
490 }
491
492 }
493
494 }
495
496 }
497
498 }
499
500 }
501
502 }
503
504 }
505
506 }
507
508 }
509
510 }
511
512 }
513
514 }
515
516 }
517
518 }
519
520 }
521
522 }
523
524 }
525
526 }
527
528 }
529
530 }
531
532 }
533
534 }
535
536 }
537
538 }
539
540 }
541
542 }
543
544 }
545
546 }
547
548 }
549
550 }
551
552 }
553
554 }
555
556 }
557
558 }
559
560 }
561
562 }
563
564 }
565
566 }
567
568 }
569
570 }
571
572 }
573
574 }
575
576 }
577
578 }
579
580 }
581
582 }
583
584 }
585
586 }
587
588 }
589
590 }
591
592 }
593
594 }
595
596 }
597
598 }
599
600 }
601
602 }
603
604 }
605
606 }
607
608 }
609
610 }
611
612 }
613
614 }
615
616 }
617
618 }
619
620 }
621
622 }
623
624 }
625
626 }
627
628 }
629
630 }
631
632 }
633
634 }
635
636 }
637
638 }
639
640 }
641
642 }
643
644 }
645
646 }
647
648 }
649
650 }
651
652 }
653
654 }
655
656 }
657
658 }
659
660 }
661
662 }
663
664 }
665
666 }
667
668 }
669
670 }
671
672 }
673
674 }
675
676 }
677
678 }
679
680 }
681
682 }
683
684 }
685
686 }
687
688 }
689
690 }
691
692 }
693
694 }
695
696 }
697
698 }
699
700 }
701
702 }
703
704 }
705
706 }
707
708 }
709
710 }
711
712 }
713
714 }
715
716 }
717
718 }
719
720 }
721
722 }
723
724 }
725
726 }
727
728 }
729
730 }
731
732 }
733
734 }
735
736 }
737
738 }
739
740 }
741
742 }
743
744 }
745
746 }
747
748 }
749
750 }
751
752 }
753
754 }
755
756 }
757
758 }
759
760 }
761
762 }
763
764 }
765
766 }
767
768 }
769
770 }
771
772 }
773
774 }
775
776 }
777
778 }
779
780 }
781
782 }
783
784 }
785
786 }
787
788 }
789
790 }
791
792 }
793
794 }
795
796 }
797
798 }
799
800 }
801
802 }
803
804 }
805
806 }
807
808 }
809
810 }
811
812 }
813
814 }
815
816 }
817
818 }
819
820 }
821
822 }
823
824 }
825
826 }
827
828 }
829
830 }
831
832 }
833
834 }
835
836 }
837
838 }
839
840 }
841
842 }
843
844 }
845
846 }
847
848 }
849
850 }
851
852 }
853
854 }
855
856 }
857
858 }
859
860 }
861
862 }
863
864 }
865
866 }
867
868 }
869
870 }
871
872 }
873
874 }
875
876 }
877
878 }
879
880 }
881
882 }
883
884 }
885
886 }
887
888 }
889
890 }
891
892 }
893
894 }
895
896 }
897
898 }
899
900 }
901
902 }
903
904 }
905
906 }
907
908 }
909
910 }
911
912 }
913
914 }
915
916 }
917
918 }
919
920 }
921
922 }
923
924 }
925
926 }
927
928 }
929
930 }
931
932 }
933
934 }
935
936 }
937
938 }
939
940 }
941
942 }
943
944 }
945
946 }
947
948 }
949
950 }
951
952 }
953
954 }
955
956 }
957
958 }
959
960 }
961
962 }
963
964 }
965
966 }
967
968 }
969
970 }
971
972 }
973
974 }
975
976 }
977
978 }
979
980 }
981
982 }
983
984 }
985
986 }
987
988 }
989
990 }
991
992 }
993
994 }
995
996 }
997
998 }
999
1000 }
```

```
/rest/resttemplate/DBServiceRestTemplate.java - Spring Tool Suite 4
RestConnectionUtil.java DBServiceRestTemplate.java USRAccountOpeningAPI.java USRAccountOpeningAPIController.java ApplicationIDService.java

528
529 @HystrixCommand(commandKey = "hystrixCommandKeyGetApplicationID", threadPoolKey = "hystrixThreadPoolGetApplicationID", fallbackMethod =
530 "getApplicationIDFallback")
531 public GetApplicationIdResponseDTO getApplicationId(String uid, String businessCode, String countryCode, String channelId, String channelType,
532 String avokaTrackingID, String userRole) {
533
534     GetApplicationIdResponseDTO response = new GetApplicationIdResponseDTO();
535
536     try {
537         HttpEntity<String> requestEntity = new HttpEntity<>(CommonUtility.getHttpHeaders(uid, businessCode, countryCode, channelId, null, null));
538         logger.debug("Request Entity: {} ", requestEntity);
539         URI serviceURI = null;
540         serviceURI = connectionUtils.getURI(onboardingService, "/crud/v1/na/bank/retail/getApplicationID");
541         URI finalURI = UriComponentsBuilder.fromUri(serviceURI).queryParam("channelType", channelType).queryParam("avokaTrackingID",
542 avokaTrackingID).queryParam("userRole", userRole).build().toUri();
543         logger.info("DBServiceRestTemplate URI: {} ", serviceURI);
544         if (serviceURI != null) {
545             ResponseEntity<GetApplicationIdResponseDTO> result = restTemplate.exchange(finalURI.toString(), HttpMethod.GET, requestEntity,
546 GetApplicationIdResponseDTO.class);
547             response = result.getBody();
548             logger.debug("ResponseEntity: {} ", response);
549         }
550     } catch (Exception ex) {
551         CommonUtility.LogException(ex, "DBServiceRestTemplate.getApplicationID");
552     }
553     return response;
554 }
555
556 public GetApplicationIdResponseDTO getApplicationIdFallback(String uid, String businessCode, String countryCode, String channelId, String channelType,
557 String avokaTrackingID, String userRole) {
558     GetApplicationIdResponseDTO response = new GetApplicationIdResponseDTO();
559     logger.error("Hystrix circuit breaker has tripped in DBServiceRestTemplate.getApplicationID()");
560     return response;
561 }
562
563 @HystrixCommand(commandKey = "hystrixCommandKeyGetBranchDetails", threadPoolKey = "hystrixThreadPoolGetBranchDetails", fallbackMethod = "getBranchDetailsFallback")
564 public GetBranchDetailsResponseDTO getBranchDetails(String uid, String businessCode, String countryCode, String channelId, GetBranchDetailsRequestDTO request)
565 GetBranchDetailsResponseDTO response = new GetBranchDetailsResponseDTO();
566
567     try {
568         HttpEntity<GetBranchDetailsRequestDTO> requestEntity = new HttpEntity<>(request, CommonUtility.getHttpHeaders(uid, businessCode, countryCode,
569 channelId, null, null));
570
571         logger.debug("Request Entity: {} ", requestEntity);
572         URI serviceURI = null;
573         serviceURI = connectionUtils.getURI(onboardingService, "/USRT/DBServices/getBranchDetails");
574         logger.info("DBServiceRestTemplate URI: {} ", serviceURI);
575         if (serviceURI != null) {
576             ResponseEntity<GetBranchDetailsResponseDTO> result = restTemplate.exchange(serviceURI.toString(), HttpMethod.POST, requestEntity,
577 GetBranchDetailsResponseDTO.class);
578             response = result.getBody();
579             logger.debug("ResponseEntity: {} ", response);
580         }
581     } catch (Exception ex) {
582         CommonUtility.LogException(ex, "DBServiceRestTemplate.getBranchDetails");
583     }
584
585     return response;
586 }
587
588 }
589
590 }
591
592 }
593
594 }
595
596 }
597
598 }
599
600 }
601
602 }
603
604 }
605
606 }
607
608 }
609
610 }
611
612 }
613
614 }
615
616 }
617
618 }
619
620 }
621
622 }
623
624 }
625
626 }
627
628 }
629
630 }
631
632 }
633
634 }
635
636 }
637
638 }
639
640 }
641
642 }
643
644 }
645
646 }
647
648 }
649
650 }
651
652 }
653
654 }
655
656 }
657
658 }
659
660 }
661
662 }
663
664 }
665
666 }
667
668 }
669
670 }
671
672 }
673
674 }
675
676 }
677
678 }
679
680 }
681
682 }
683
684 }
685
686 }
687
688 }
689
690 }
691
692 }
693
694 }
695
696 }
697
698 }
699
700 }
701
702 }
703
704 }
705
706 }
707
708 }
709
710 }
711
712 }
713
714 }
715
716 }
717
718 }
719
720 }
721
722 }
723
724 }
725
726 }
727
728 }
729
730 }
731
732 }
733
734 }
735
736 }
737
738 }
739
740 }
741
742 }
743
744 }
745
746 }
747
748 }
749
750 }
751
752 }
753
754 }
755
756 }
757
758 }
759
760 }
761
762 }
763
764 }
765
766 }
767
768 }
769
770 }
771
772 }
773
774 }
775
776 }
777
778 }
779
780 }
781
782 }
783
784 }
785
786 }
787
788 }
789
790 }
791
792 }
793
794 }
795
796 }
797
798 }
799
800 }
801
802 }
803
804 }
805
806 }
807
808 }
809
810 }
811
812 }
813
814 }
815
816 }
817
818 }
819
820 }
821
822 }
823
824 }
825
826 }
827
828 }
829
830 }
831
832 }
833
834 }
835
836 }
837
838 }
839
840 }
841
842 }
843
844 }
845
846 }
847
848 }
849
850 }
851
852 }
853
854 }
855
856 }
857
858 }
859
860 }
861
862 }
863
864 }
865
866 }
867
868 }
869
870 }
871
872 }
873
874 }
875
876 }
877
878 }
879
880 }
881
882 }
883
884 }
885
886 }
887
888 }
889
890 }
891
892 }
893
894 }
895
896 }
897
898 }
899
900 }
901
902 }
903
904 }
905
906 }
907
908 }
909
910 }
911
912 }
913
914 }
915
916 }
917
918 }
919
920 }
921
922 }
923
924 }
925
926 }
927
928 }
929
930 }
931
932 }
933
934 }
935
936 }
937
938 }
939
940 }
941
942 }
943
944 }
945
946 }
947
948 }
949
950 }
951
952 }
953
954 }
955
956 }
957
958 }
959
960 }
961
962 }
963
964 }
965
966 }
967
968 }
969
970 }
971
972 }
973
974 }
975
976 }
977
978 }
979
980 }
981
982 }
983
984 }
985
986 }
987
988 }
989
990 }
991
992 }
993
994 }
995
996 }
997
998 }
999
1000 }
```



```

321 public ResponseEntity<Boolean> saveVerification(@RequestHeader(value = "uuid", required = true) String uuid,
322 @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
323 @RequestHeader(value = "countryCode", required = false, defaultValue = "US") String countryCode,
324 @RequestHeader(value = "channelId", required = false, defaultValue = "RTLAO") String channelId,
325 @RequestBody SaveCreditVerificationResultRequestDTO creditVerificationResponse) {
326
327     Boolean res = saveVerificationResultsService.saveCreditVerification(creditVerificationResponse);
328     return new ResponseEntity<>(res, HttpStatus.OK);
329 }
330
331 @GetMapping(value = "/crud/v1/na/bank/retail/getApplicationID", produces = { "application/json" }, consumes = {
332     "application/json" })
333 public ResponseEntity<ApplicationIdResponseDTO> getApplicationID(
334     @RequestHeader(value = "uuid", required = true) String uuid,
335     @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
336     @RequestHeader(value = "countryCode", required = false, defaultValue = "US") String countryCode,
337     @RequestHeader(value = "channelId", required = false, defaultValue = "RTLAO") String channelId,
338     @RequestParam(value = "channelType", required = false) String channelType,
339     @RequestParam(value = "avokaTrackingID", required = false) String avokaTrackingID,
340     @RequestParam(value = "userRole", required = false) String userRole) {
341     return new ResponseEntity<>(
342         generateApplicationIDService.generateApplicationIDResponse(channelType, avokaTrackingID, userRole),
343         HttpStatus.OK);
344 }
345
346 @GetMapping(value = "/private/v1/bank/onboarding/applications/forms/{applicationId}/requestId", produces = {
347     "application/json" }, consumes = { "application/json" })
348 public ResponseEntity<GenerateRequestIdResponse> generateRequestId(
349     @RequestHeader(value = "uuid", required = true) String uuid,
350     @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
351     @RequestHeader(value = "countryCode", required = false, defaultValue = "US") String countryCode,
352     @RequestHeader(value = "channelId", required = false, defaultValue = "RTLAO") String channelId,
353     @RequestHeader(value = "Authorization", required = false) String authorization,
354     @RequestHeader(value = "sid", required = false) String sid,
355     @PathVariable("applicationId") String applicationId) {
356     return new ResponseEntity<>(getRequestIDService.generateRequestIdResponse(applicationId), HttpStatus.OK);
357 }
358
359 @GetMapping(value = "/crud/v1/na/bank/retail/citiScreeningResponse/{applicationID}", produces = {
360     "application/json" }, consumes = { "application/json" })
361 @ResponseBody
362 public ResponseEntity<List<CitiScreeningResultDTO>> getCitiScreeningResults(
363     @RequestHeader(value = "uuid", required = true) String uuid,

```