

```

RestConnectio... DBServiceRes... AddCheckStopP... GenerateAppli... AddCheckStopP... AddCheckStopP... USRAOServicin... *U
73
74
75
76 @PostMapping(value = "/private/v1/bank/account/check/addStopPay", produces = { "application/json" })
77 ResponseEntity<AddCheckStopPayResponse> addStopPay(
78     @RequestHeader(value = "client_id", required = false) String clientId,
79     @RequestHeader(value = "Authorization", required = false) String authorization,
80     @RequestHeader(value = "Accept", required = true) String acceptLanguage,
81     @RequestHeader(value = "uuid", required = true) String uuid,
82     @RequestHeader(value = "businessCode", required = true) String businessCode,
83     @RequestHeader(value = "CountryCode", required = true) String countryCode,
84     @RequestHeader(value = "channelId", required = true) String channelId,
85     @RequestHeader(value = "Content-Type", required = true) String contentType,
86     @Valid @RequestBody AddCheckStopPayRequest addCheckStopPayRequest);
87 }
88
89
90
91
92
93
94
95
96
97

```

```

1 package com.citi.na.usrao.servicingorder.model.dto;
2
3 import com.citi.na.usrao.servicingorder.validator.AddCheckStopPayRequestValid;
4
5
6 AddCheckStopPayRequestValid
7 public class AddCheckStopPayRequest {
8
9     private String stopPayType;
10    private String destinationSystem;
11    private String messageSubProcessingIndicator;
12    private String sourceSystemId;
13    private String fimp;
14    private String accountId;
15    private String checkStartRange;
16    private String checkEndRange;
17    private String sessionEndTime;
18    private String duplicationDetectionId;
19    private String checkAmount;
20    private String payeeName;
21    private String feeAmount;
22    private String reason;
23
24    public String getStopPayType() {
25        return stopPayType;
26    }
27    public void setStopPayType(String stopPayType) {
28        this.stopPayType = stopPayType;
29    }
30    public String getDestinationSystem() {
31        return destinationSystem;
32    }
33    public void setDestinationSystem(String destinationSystem) {
34        this.destinationSystem = destinationSystem;
35    }
36    public String getMessageSubProcessingIndicator() {
37        return messageSubProcessingIndicator;
38    }
39    public void setMessageSubProcessingIndicator(String messageSubProcessingIndicator) {
40        this.messageSubProcessingIndicator = messageSubProcessingIndicator;
41    }
42    public String getSourceSystemId() {
43        return sourceSystemId;
44    }
45    public void setSourceSystemId(String sourceSystemId) {
46        this.sourceSystemId = sourceSystemId;
47    }
48    public String getFimp() {

```

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

AddCheckStopPayRequestValid - 17 matches in workspace (*.java)


```
ckStopPayRequestValid.java - Spring Tool Suite 4

package com.citi.na.usrao.servicingorder.validator;
import java.lang.annotation.Documented;
import java.lang.annotation.Target;
import javax.validation.Constraint;
import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;
import javax.validation.Payload;

@Constraint(validatedBy = AddCheckStopPayRequestValidator.class)
@Target({ElementType.TYPE, ElementType.FIELD, ElementType.PARAMETER, ElementType.CONSTRUCTOR, ElementType.LOCAL_VARIABLE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface AddCheckStopPayRequestValid {

    String message() default "Please provide correct request";

    Class[] groups() default {};

    Class extends Payload[] payload() default {};
}

I
```

```
ckStopPayRequestValid.java - Spring Tool Suite 4

package com.citi.na.usrao.servicingorder.validator;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class AddCheckStopPayRequestValidator
    implements ConstraintValidator<AddCheckStopPayRequestValid, AddCheckStopPayRequest> {

    private List<String> reasonCode = new ArrayList<>();
    private static final String AMOUNT_PATTERN = "^[0-9]{0,15}[0-9]{0,15}[0-9]{1,3}$";
    private static final String NUMERIC_PATTERN = "^[0-9]+$";
    private static final String TIME_PATTERN = "^(\\d{4})\\-(0[1-9]|1[0-2])\\-(\\d{2}|\\d{1}[0-9]|3[01])\\T((01|\\d{2}|2[0-3])\\:([0-5]\\d|\\:([0-5]\\d)?)?$";
    private static final String CHECKBOOK_NUMBER_PATTERN = "^[0-9]{10}$";
    private static final String MESSAGE_SUB_PROCESSING_INDICATOR_PATTERN = "^[1|2|3|4]$";
    private static final String FIMP_PATTERN = "^[0-9]{3}$";
    private static final String IMPACS = "IMPACS";
    private static final String TPS = "TPS";
    private static final String CHECK = "CHECK";

    @Override
    public boolean isValid(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
        boolean result = true;
        arg1.disableDefaultConstraintViolation();

        if (arg0 == null) {
            arg1.buildConstraintViolationWithTemplate("Request cannot be empty").addConstraintViolation();
            result = false;
        } else {
            result = validateRequest(arg0, arg1);
        }
        return result;
    }

    private boolean validateRequest(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
        boolean result = true;
    }
}
```



```
CheckStopPayRequestValidator.java - Spring Tool Suite 4

50 private boolean validateRequest(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
51     boolean result = true;
52     if (!mandatoryFieldsValidation(arg0, arg1) || !patternCheck(arg0, arg1)) {
53         result = false;
54     } else if (!arg0.getStopPayType().equals(CHECK) || arg0.getStopPayType().equals("ACH")) {
55         arg1.buildConstraintViolationWithTemplate("StopPayType should be either CHECK or ACH")
56             .addConstraintViolation();
57         result = false;
58     } else if (!arg0.getDestinationSystem().equals(TPS) || arg0.getDestinationSystem().equals(IMPACS)) {
59         arg1.buildConstraintViolationWithTemplate("DestinationSystem should be either TPS or IMPACS")
60             .addConstraintViolation();
61         result = false;
62     } else if (arg0.getStopPayType().equals("ACH") && arg0.getDestinationSystem().equals(TPS)) {
63         arg1.buildConstraintViolationWithTemplate("DestinationSystem value TPS not allowed for StopPayType value ACH")
64             .addConstraintViolation();
65         result = false;
66     }
67     return result;
68 }
69
70 private boolean mandatoryFieldsValidation(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
71     boolean result = true;
72     if (nullOrEmptyCheck(arg0.getDestinationSystem())) {
73         arg1.buildConstraintViolationWithTemplate("DestinationSystem should not be empty").addConstraintViolation();
74         result = false;
75     } else {
76         result = validateDestinationSystem(arg0, arg1);
77     }
78     if (nullOrEmptyCheck(arg0.getStopPayType())) {
79         arg1.buildConstraintViolationWithTemplate("StopPayType should not be empty").addConstraintViolation();
80         result = false;
81     } else {
82         if (arg0.getStopPayType().equals(CHECK) && arg0.getDestinationSystem().equals(IMPACS)
83             && (nullOrEmptyCheck(arg0.getCheckStartRange()))) {
84             arg1.buildConstraintViolationWithTemplate("CheckStartRange should not be empty for Stop pay type CHECK")
85                 .addConstraintViolation();
86             result = false;
87         }
88     }
89     if (nullOrEmptyCheck(arg0.getSessionDateTime())) {
90         arg1.buildConstraintViolationWithTemplate("SessionDateTime should not be empty and it should be in EST Time zone")
91             .addConstraintViolation();
92         result = false;
93     }
94     if (nullOrEmptyCheck(arg0.getAccountId())) {
95         arg1.buildConstraintViolationWithTemplate("AccountId should not be empty")
96             .addConstraintViolation();
97         result = false;
98     }
99 }
100
101 private boolean validateDestinationSystem(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
102     boolean result = true;
103     if (TPS.equals(arg0.getDestinationSystem())) {
104         if (nullOrEmptyCheck(arg0.getMessageSubProcessingIndicator())) {
105             arg1.buildConstraintViolationWithTemplate("MessageSubProcessingIndicator should not be empty for DestinationSystem TPS")
106                 .addConstraintViolation();
107             result = false;
108         } else {
109             result = validationBasedOnMessageSubProcessingIndicator(arg0, arg1);
110         }
111     } else if (IMPACS.equals(arg0.getDestinationSystem())) {
112         if (nullOrEmptyCheck(arg0.getFeeAmount())) {
113             arg1.buildConstraintViolationWithTemplate("FeeAmount should not be empty for DestinationSystem IMPACS")
114                 .addConstraintViolation();
115             result = false;
116         }
117     }
118     if (CHECK.equals(arg0.getStopPayType()) && nullOrEmptyCheck(arg0.getReason())) {
119         arg1.buildConstraintViolationWithTemplate("Reason should not be empty for StopPayType CHECK and DestinationSystem IMPACS")
120             .addConstraintViolation();
121         result = false;
122     }
123     return result;
124 }
125
126 private boolean validationBasedOnMessageSubProcessingIndicator(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
127     boolean result = true;
128     if (nullOrEmptyCheck(arg0.getFimp())) {
129         arg1.buildConstraintViolationWithTemplate("Fimp should not be empty for DestinationSystem TPS")
130             .addConstraintViolation();
131         result = false;
132     }
133     return result;
134 }
135
136 private boolean nullOrEmptyCheck(String str) {
137     return str == null || str.isEmpty();
138 }
139
140 private boolean patternCheck(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
141     boolean result = true;
142     if (!arg0.getCheckStartRange().matches("\\d{4}-\\d{2}-\\d{2}")) {
143         arg1.buildConstraintViolationWithTemplate("CheckStartRange should be in YYYY-MM-DD format")
144             .addConstraintViolation();
145         result = false;
146     }
147     return result;
148 }
```

```
101 }
102
103 if (nullOrEmptyCheck(arg0.getAccountId())) {
104     arg1.buildConstraintViolationWithTemplate("AccountId should not be empty").addConstraintViolation();
105     result = false;
106 }
107 if (IMPACS.equals(arg0.getDestinationSystem()) && nullOrEmptyCheck(arg0.getCheckAmount())) {
108     arg1.buildConstraintViolationWithTemplate("CheckAmount should not be empty").addConstraintViolation();
109     result = false;
110 }
111 if (nullOrEmptyCheck(arg0.getSourceSystemId())) {
112     arg1.buildConstraintViolationWithTemplate("SourceSystemId should not be empty").addConstraintViolation();
113     result = false;
114 }
115 return result;
116 }
117
118 private boolean validateDestinationSystem(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
119     boolean result = true;
120     if (TPS.equals(arg0.getDestinationSystem())) {
121         if (nullOrEmptyCheck(arg0.getMessageSubProcessingIndicator())) {
122             arg1.buildConstraintViolationWithTemplate("MessageSubProcessingIndicator should not be empty for DestinationSystem TPS")
123                 .addConstraintViolation();
124             result = false;
125         } else {
126             result = validationBasedOnMessageSubProcessingIndicator(arg0, arg1);
127         }
128     } else if (IMPACS.equals(arg0.getDestinationSystem())) {
129         if (nullOrEmptyCheck(arg0.getFeeAmount())) {
130             arg1.buildConstraintViolationWithTemplate("FeeAmount should not be empty for DestinationSystem IMPACS")
131                 .addConstraintViolation();
132             result = false;
133         }
134     }
135     if (CHECK.equals(arg0.getStopPayType()) && nullOrEmptyCheck(arg0.getReason())) {
136         arg1.buildConstraintViolationWithTemplate("Reason should not be empty for StopPayType CHECK and DestinationSystem IMPACS")
137             .addConstraintViolation();
138         result = false;
139     }
140     return result;
141 }
142
143 private boolean validationBasedOnMessageSubProcessingIndicator(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
144     boolean result = true;
145     if (nullOrEmptyCheck(arg0.getFimp())) {
146         arg1.buildConstraintViolationWithTemplate("Fimp should not be empty for DestinationSystem TPS")
147             .addConstraintViolation();
148         result = false;
149     }
150     return result;
151 }
152
153 private boolean nullOrEmptyCheck(String str) {
154     return str == null || str.isEmpty();
155 }
```



```
152 }
153
154 private boolean validationBasedOnMessageSubProcessingIndicator(AddCheckStopPayRequest arg0,
155     ConstraintValidatorContext arg1) {
156     boolean result = true;
157
158     if (("1".equals(arg0.getMessageSubProcessingIndicator()) || "2".equals(arg0.getMessageSubProcessingIndicator()))
159         && !nullOrEmptyCheck(arg0.getCheckAmount())) {
160         arg1.buildConstraintViolationWithTemplate(
161             "CheckAmount should not be empty for MessageSubProcessingIndicator : 1, 2")
162             .addConstraintViolation();
163         result = false;
164     }
165
166     if (("2".equals(arg0.getMessageSubProcessingIndicator()) || "3".equals(arg0.getMessageSubProcessingIndicator()))
167         || "4".equals(arg0.getMessageSubProcessingIndicator()))
168         && !nullOrEmptyCheck(arg0.getCheckStartRange())) {
169         arg1.buildConstraintViolationWithTemplate(
170             "CheckStartRange should not be empty for MessageSubProcessingIndicator : 2, 3, 4")
171             .addConstraintViolation();
172         result = false;
173     }
174
175     if ("4".equals(arg0.getMessageSubProcessingIndicator()) && !nullOrEmptyCheck(arg0.getCheckEndRange())) {
176         arg1.buildConstraintViolationWithTemplate(
177             "CheckEndRange should not be empty for MessageSubProcessingIndicator : 4").addConstraintViolation();
178         result = false;
179     }
180
181     return result;
182 }
183
184 private boolean nullOrEmptyCheck(String field) {
185     boolean result = false;
186
187     if (field == null || field.isEmpty())
188         result = true;
189
190     return result;
191 }
192
193 private boolean patternCheck(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {
194     boolean result = true;
195
196     if (!patternCheck(NUMERIC_PATTERN, arg0.getAccountId())) {
197         arg1.buildConstraintViolationWithTemplate("AccountId should be numeric").addConstraintViolation();
198         result = false;
199     }
200
201     if (arg0.getCheckStartRange() != null && !arg0.getCheckStartRange().isEmpty()
202         && !patternCheck(CHECKBOOK_NUMBER_PATTERN, arg0.getCheckStartRange())) {
203         arg1.buildConstraintViolationWithTemplate("CheckStartRange should be 10 digit numeric value")
204             .addConstraintViolation();
205     }
206 }
```

```
207
208     && !patternCheck(CHECKBOOK_NUMBER_PATTERN, arg0.getCheckEndRange())) {
209         arg1.buildConstraintViolationWithTemplate("CheckEndRange should be 10 digit numeric value")
210             .addConstraintViolation();
211         result = false;
212     }
213
214     if (arg0.getCheckAmount() != null && !arg0.getCheckAmount().isEmpty()
215         && !patternCheck(AMOUNT_PATTERN, arg0.getCheckAmount())) {
216         arg1.buildConstraintViolationWithTemplate(
217             "Check Amount should be numeric. Allowed integer-part is 15 digit & fraction-part is 3 digit")
218             .addConstraintViolation();
219         result = false;
220     }
221
222     if (IMPACS.equals(arg0.getDestinationSystem()) && !patternCheck(AMOUNT_PATTERN, arg0.getFeeAmount())) {
223         arg1.buildConstraintViolationWithTemplate(
224             "Fee Amount should be numeric. Allowed integer-part is 15 digit & fraction-part is 3 digit")
225             .addConstraintViolation();
226         result = false;
227     }
228
229     if (!patternCheck(TIME_PATTERN, arg0.getSessionDateTime())) {
230         arg1.buildConstraintViolationWithTemplate(
231             "SessionDateTime should be in EST Time zone and in format 'yyyy-MM-dd'T'HH:mm:ss'. Example: 2021-01-01T23:00:00")
232             .addConstraintViolation();
233         result = false;
234     }
235
236     if (arg0.getSourceSystemId().equals("03")) {
237         arg1.buildConstraintViolationWithTemplate("SourceSystemId not onboarded").addConstraintViolation();
238         result = false;
239     }
240
241     if (IMPACS.equals(arg0.getDestinationSystem()) && arg0.getReason() != null
242         && !reasonCode.contains(arg0.getReason())) {
243         arg1.buildConstraintViolationWithTemplate("ReasonCode not valid").addConstraintViolation();
244         result = false;
245     }
246
247     if (TPS.equals(arg0.getDestinationSystem())
248         && !patternCheck(MESSAGE_SUB_PROCESSING_INDICATOR_PATTERN, arg0.getMessageSubProcessingIndicator())) {
249         arg1.buildConstraintViolationWithTemplate(
250             "Allowed values of MessageSubProcessingIndicator are 1, 2, 3 and 4").addConstraintViolation();
251         result = false;
252     }
253
254     if (TPS.equals(arg0.getDestinationSystem()) && !patternCheck(FIMP_PATTERN, arg0.getFimp())) {
255         arg1.buildConstraintViolationWithTemplate("Fimp should be 3 digit numeric value").addConstraintViolation();
256         result = false;
257     }
258
259     return result;
260 }
```