MODULE *grpc*

Specification for the zebra-*grpc* crate design and it's relationship with the zebra-scanner crate and zebrad configuration file. It can handle the scan task functionality and how the *grpc* methods can add or delete information to the scanning database.

The spec is written in *PlusCal* and it's meant to be used with the *TLC* model checker.

The spec is divided in two parts: the first part is the *PlusCal* spec and the second part is translated TLA+ code.

The spec is divided in the following sections:

1. Configuration Constants
2. Global Variables
3. Type Invariants
4. Utility Functions
5. *gRPC* Methods
6. Services Process
7. Scan Task Process
8. Main Program Process

For more information visit: https://*github.com*/oxarbitrage/zebra-*grpc*-scan-spec)

EXTENDS *TLC*, *Integers*, *Sequences*, *Randomization*, *FiniteSets*

CONFIGURATION CONSTANTS:

The set of keys as strings to be added to the scan task from the config file.

CONSTANT *ConfigViewingKeys*

We have 3 batches of keys so we can try different combinations, including duplicated keys.

A set of keys as strings.

CONSTANT $GrpcViewingKeysBatch1$

A second set of keys as strings.

CONSTANT $GrpcViewingKeysBatch2$

A third set of keys as strings.

CONSTANT $GrpcViewingKeysBatch3$

The maximum number of scan tasks that can be added to the scan task set.

CONSTANT *MaxScanTasks*

GLOBAL VARIABLES:

A sequence of batches with keys to call *grpc* methods. Currently we have 3 batches.

$GrpcViewingKeys \triangleq \langle GrpcViewingKeysBatch1, GrpcViewingKeysBatch2, GrpcViewingKeysBatch3 \rangle$

A dummy response to an *Info request*.

$info\_response \triangleq [saplingheight \mapsto 1]$

A random list of transations to be used as a Results response.

$results\_response \triangleq [transactions \mapsto RandomSetOfSubsets(1, 3, 1\,..\,10)]$

An empty response to Register

$register\_response \triangleq [empty \mapsto \{\}]$

An empty response to Delete

$delete\_response \triangleq [empty \mapsto \{\}]$

An empty response to Clear

$clear\_response \triangleq [empty \mapsto \{\}]$

An empty response to Subscribe TODO: which should be a channel with updates.

$subscribe\_response \triangleq [empty \mapsto \{\}]$

An empty response to Status TODO: which should have some data from the scan task for the key.

$status\_response \triangleq [empty \mapsto \{\}]$

The set of statuses a scan task can be on at any given time.

$scan\_task\_statuses \triangleq \{\text{“waiting”}, \text{“adding”}, \text{“deleting”}\}$

The set of valid service requests.

$service\_requests \triangleq \{\text{“waiting”}, \text{“adding”}, \text{“deleting”}\}$

```
--algorithm grpc
variables
    The scan tasks are a set that is initially empty.
    $scan\_tasks = \{\}$ ;
    A string that will be used as a response to any of the gRPC method calls.
    $response = \text{“”}$ ;
    The status of the scan task, initially listening.
    $scan\_task\_status = \text{“waiting”}$ ;
    A key to be passed to any of the services, and also added or deleted to/from
    the scan task at a given instant, initially empty.
    $key\_to\_be\_served = \text{“”}$ ;
    The current service request flag.
    $service\_request = \text{“none”}$ ;
    The number of batches the configuration has.
    $number\_of\_batches = 0$ ;
    The counter for the number of batches.
    $counter = 1$ ;

THE TYPE INVARIANTS :
define
  $TypeInvariant \triangleq$
      The response is always in the STRING domain
      $\land\ response \in \text{STRING}$
      The scan task status is always in the scan task statuses set.
      $\land\ scan\_task\_status \in scan\_task\_statuses$
      The key to be served is always in the STRING domain.
      $\land\ key\_to\_be\_served \in \text{STRING}$
      The service request is always in the service requests set.
      $\land\ service\_request \in service\_requests$
end define ;

UTILITY FUNCTIONS::
  Helper function to get the number of non empty batches the configuration has.
procedure get_config_number_of_batches()
begin
    CheckBatch1:
        if $GrpcViewingKeysBatch1 \neq \{\}$ then
            $number\_of\_batches := number\_of\_batches + 1$ ;
        end if ;
    CheckBatch2:
        if $GrpcViewingKeysBatch2 \neq \{\}$ then
            $number\_of\_batches := number\_of\_batches + 1$ ;
        end if ;
    CheckBatch3:
        if $GrpcViewingKeysBatch3 \neq \{\}$ then
            $number\_of\_batches := number\_of\_batches + 1$ ;
        end if ;
        return ;
end procedure ;

  Call the scan task to add keys coming from the config file.
procedure add_config_keys(keys)
begin
    AddConfigKeys:
        with $key \in keys$ do
            $key\_to\_be\_served := key$ ;
            $scan\_task\_status$ := "adding" ;
            return ;
        end with ;
end procedure ;

  GRPC METHODS :

  The get_info grpc method.
procedure get_info()
begin
    InfoServiceRequest:
        $service\_request$ := "info" ;
        return ;
end procedure ;

  The get_results grpc method.
procedure get_results(keys)
begin
    ResultsServiceRequest:
        with $key \in keys$ do
            $key\_to\_be\_served := key$ ;
            $service\_request$ := "results" ;
        return;
    end with ;
end procedure ;

 The clear_results grpc method.
procedure clear_results(keys)
begin
  ClearServiceRequest:
    with $key \in keys$ do
        key_to_be_served := key ;
        service_request := "clear" ;
        return;
    end with ;
end procedure ;

 The get_status grpc method.
procedure get_status(keys)
begin
  StatusServiceRequest:
    with $key \in keys$ do
        key_to_be_served := key ;
        service_request := "status" ;
        return;
    end with ;
end procedure ;

 The register_keys grpc method.
procedure register_keys(keys)
begin
  RegisterServiceRequest:
    with $key \in keys$ do
        key_to_be_served := key ;
        service_request := "register" ;
        return;
    end with ;
end procedure ;

 The delete_keys grpc method.
procedure delete_keys(keys)
begin
  DeleteServiceRequest:
    with $key \in keys$ do
        key_to_be_served := key ;
        service_request := "delete" ;
        return;
    end with ;
end procedure ;

  The scan grpc method.
  The method call 3 services one next to the other.
procedure scan(keys)
begin
    RegisterServiceRequestFromScan:
        with key ∈ keys do
            key_to_be_served := key ;
            service_request := "register" ;
        end with ;
    ResultsServiceRequestFromScan:
        with key ∈ keys do
            key_to_be_served := key ;
            service_request := "results" ;
        end with ;
    SubscribeServiceRequestFromScan:
        with key ∈ keys do
            key_to_be_served := key ;
            service_request := "subscribe" ;
            return;
        end with ;
end procedure ;

  SERVICES PROCESS :

  Listen for requests, send requests to scan task where is needed and provide responses.
process services = "SERVICES"
begin
    Services:
        if service_request = "info" then
            Info:
                response := info_response ;
        elsif service_request = "results" then
            Results:
                if key_to_be_served ∈ scan_tasks then
                    response := results_response ;
                else
                    response := "Error: key not found." ;
                end if ;
        elsif service_request = "clear" then
            Clear:
                if key_to_be_served ∈ scan_tasks then
                    response := clear_response ;
                else
                    response := "Error: key not found." ;
            end if ;
        elsif service_request = "status" then
          Status:
            if key_to_be_served ∈ scan_tasks then
              response := status_response ;
             else
              response := "Error: key not found." ;
            end if ;
        elsif service_request = "register" then
          Register:
            if key_to_be_served ∈ scan_tasks then
              KeyError:
                response := "Error: key already in scan task." ;
             else
              Success:
                scan_task_status := "adding" ;
                response := register_response ;
            end if ;
        elsif service_request = "delete" then
          Delete:
            if key_to_be_served ∈ scan_tasks then
              scan_task_status := "deleting" ;
              response := delete_response ;
             else
              response := "Error: key not found." ;
            end if ;
        elsif service_request = "subscribe" then
          Subscribe:
            if key_to_be_served ∈ scan_tasks then
              response := subscribe_response ;
             else
              response := "Error: key not found." ;
            end if ;
      end if ;
    Make the process loops forever.
   ServicesLoop:
      goto Services ;
end process ;

 SCAN TASK PROCESS :

 Listen for requests from the services process, add or remove tasks to the scan task set.
process scantask = "SCAN TASK"
variables inner_state = {} ;
begin
```

```
    $GetScanTasks$:
        $inner\_state := scan\_tasks$ ;
    $ScanTask$:
        if $Cardinality(scan\_tasks) > MaxScanTasks$ then
            $BoundError$:
                $response :=$ "Error: max scan tasks reached." ;
                $scan\_task\_status :=$ "waiting" ;
        elsif $scan\_task\_status =$ "adding" then
            $Adding$:
                $inner\_state := inner\_state \cup \{key\_to\_be\_served\}$ ;
                $scan\_task\_status :=$ "waiting" ;
        elsif $scan\_task\_status =$ "deleting" then
            $Deleting$:
                $scan\_tasks := scan\_tasks \setminus \{key\_to\_be\_served\}$ ;
                $scan\_task\_status :=$ "waiting" ;
        end if ;
    $StoreScanTasks$:
        $scan\_tasks := inner\_state$ ;
     Make the process loops forever.
    $ScanTaskLoop$:
        goto $ScanTask$ ;
end process ;

  MAIN PROCESS :

  Calls all grpc methods with the given keys.
process $Main =$ "MAIN"
begin
    $ConfigGuard$:
        if $ConfigViewingKeys \neq \{\}$ then
            $FromZebradConfig$:
                call $add\_config\_keys(ConfigViewingKeys)$ ;
        end if ;
    $ListeningGuard$:
        if $GrpcViewingKeys \neq \langle\rangle$ then
            $GetTotalIterationsToMake$:
                call $get\_config\_number\_of\_batches()$ ;
            $ListeningMode$:
                while $counter \leq number\_of\_batches$ do
                    $GetInfoCall$:
                        call $get\_info()$ ;
                    $RegisterKeysCall$:
                        call $register\_keys(GrpcViewingKeys[counter])$ ;
                    $GetStatusCall$:
                        call $get\_status(GrpcViewingKeys[counter])$ ;
                    $GetResultsCall$:
                        call $get\_results(GrpcViewingKeys[counter])$ ;
                    $ClearResultsCall$:
                        call $clear\_results(GrpcViewingKeys[counter])$ ;
                    $DeleteKeysCall$:
                        call $delete\_keys(GrpcViewingKeys[counter])$ ;
                    $ScanCall$:
                        call $scan(GrpcViewingKeys[counter])$ ;
                    $IncrementCounter$:
                        $counter := counter + 1$ ;
                end while ;
                goto $End$ ;
        end if ;
    $End$:
        skip ;

end process ;
end algorithm ;
  BEGIN TRANSLATION(chksum(pcal) = "e23a373d" ∧ chksum(tla) = "ca9f015")
  Parameter keys of procedure add_config_keys at line 94 col 27 changed to keys_
  Parameter keys of procedure get_results at line 113 col 23 changed to keys_g
  Parameter keys of procedure clear_results at line 124 col 25 changed to keys_c
  Parameter keys of procedure get_status at line 135 col 22 changed to keys_ge
  Parameter keys of procedure register_keys at line 146 col 25 changed to keys_r
  Parameter keys of procedure delete_keys at line 157 col 23 changed to keys_d
CONSTANT $defaultInitValue$
VARIABLES $scan\_tasks, response, scan\_task\_status, key\_to\_be\_served,$
          $service\_request, number\_of\_batches, counter, pc, stack$

  define statement
$TypeInvariant \triangleq$

  $\land\ response \in$ STRING

  $\land\ scan\_task\_status \in scan\_task\_statuses$

  $\land\ key\_to\_be\_served \in$ STRING

  $\land\ service\_request \in service\_requests$

VARIABLES $keys\_, keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state$

$vars \triangleq \langle scan\_tasks, response, scan\_task\_status, key\_to\_be\_served,$
          $service\_request, number\_of\_batches, counter, pc, stack, keys\_,$
          $keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$ProcSet \triangleq$ {"SERVICES"} ∪ {"SCAN TASK"} ∪ {"MAIN"}
```

```
Init ≜  Global variables
        ∧ scan_tasks = {}
        ∧ response = “”
        ∧ scan_task_status = “waiting”
        ∧ key_to_be_served = “”
        ∧ service_request = “none”
        ∧ number_of_batches = 0
        ∧ counter = 1
         Procedure add_config_keys
        ∧ keys_ = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure get_results
        ∧ keys_g = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure clear_results
        ∧ keys_c = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure get_status
        ∧ keys_ge = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure register_keys
        ∧ keys_r = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure delete_keys
        ∧ keys_d = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure scan
        ∧ keys = [self ∈ ProcSet ↦ defaultInitValue]
         Process scantask
        ∧ inner_state = {}
        ∧ stack = [self ∈ ProcSet ↦ ⟨⟩]
        ∧ pc = [self ∈ ProcSet ↦ CASE self = “SERVICES” → “Services”
                                   □  self = “SCAN TASK” → “GetScanTasks”
                                   □  self = “MAIN” → “ConfigGuard”]

CheckBatch1(self) ≜ ∧ pc[self] = “CheckBatch1”
                    ∧ IF GrpcViewingKeysBatch1 ≠ {}
                          THEN ∧ number_of_batches′ = number_of_batches + 1
                          ELSE ∧ TRUE
                               ∧ UNCHANGED number_of_batches
                    ∧ pc′ = [pc EXCEPT ![self] = “CheckBatch2”]
                    ∧ UNCHANGED ⟨scan_tasks, response, scan_task_status,
                                 key_to_be_served, service_request,
                                 counter, stack, keys_, keys_g, keys_c,
                                 keys_ge, keys_r, keys_d, keys,
                                 inner_state⟩

CheckBatch2(self) ≜ ∧ pc[self] = “CheckBatch2”
                    ∧ IF GrpcViewingKeysBatch2 ≠ {}
                          THEN ∧ number_of_batches′ = number_of_batches + 1
                          ELSE ∧ TRUE
                    $\land$ UNCHANGED $number\_of\_batches$
                $\land$ $pc' = [pc$ EXCEPT $![self] =$ "CheckBatch3"$]$
                $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                                  $key\_to\_be\_served, service\_request,$
                                  $counter, stack, keys\_, keys\_g, keys\_c,$
                                  $keys\_ge, keys\_r, keys\_d, keys,$
                                  $inner\_state \rangle$

$CheckBatch3(self) \triangleq$ $\land$ $pc[self] =$ "CheckBatch3"
                $\land$ IF $GrpcViewingKeysBatch3 \neq \{\}$
                      THEN $\land$ $number\_of\_batches' = number\_of\_batches + 1$
                      ELSE $\land$ TRUE
                           $\land$ UNCHANGED $number\_of\_batches$
                $\land$ $pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
                $\land$ $stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
                $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                                  $key\_to\_be\_served, service\_request,$
                                  $counter, keys\_, keys\_g, keys\_c, keys\_ge,$
                                  $keys\_r, keys\_d, keys, inner\_state \rangle$

$get\_config\_number\_of\_batches(self) \triangleq$ $CheckBatch1(self)$
                                      $\lor$ $CheckBatch2(self)$
                                      $\lor$ $CheckBatch3(self)$

$AddConfigKeys(self) \triangleq$ $\land$ $pc[self] =$ "AddConfigKeys"
                  $\land$ $\exists key \in keys\_[self]$ :
                       $\land$ $key\_to\_be\_served' = key$
                       $\land$ $scan\_task\_status' =$ "adding"
                       $\land$ $pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
                       $\land$ $keys\_' = [keys\_$ EXCEPT $![self] = Head(stack[self]).keys\_]$
                       $\land$ $stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
                  $\land$ UNCHANGED $\langle scan\_tasks, response, service\_request,$
                                    $number\_of\_batches, counter, keys\_g,$
                                    $keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
                                    $inner\_state \rangle$

$add\_config\_keys(self) \triangleq AddConfigKeys(self)$

$InfoServiceRequest(self) \triangleq$ $\land$ $pc[self] =$ "InfoServiceRequest"
                      $\land$ $service\_request' =$ "info"
                      $\land$ $pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
                      $\land$ $stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
                      $\land$ UNCHANGED $\langle scan\_tasks, response,$
                                        $scan\_task\_status, key\_to\_be\_served,$
                                        $number\_of\_batches, counter, keys\_,$
                                        $keys\_g, keys\_c, keys\_ge, keys\_r,$
```

$$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \mathit{keys\_d},\ \mathit{keys},\ \mathit{inner\_state}\rangle$$

$$\mathit{get\_info}(\mathit{self}) \triangleq \mathit{InfoServiceRequest}(\mathit{self})$$

$$\begin{array}{ll}
\mathit{ResultsServiceRequest}(\mathit{self}) \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“ResultsServiceRequest”} \\
& \land\ \exists\, \mathit{key} \in \mathit{keys\_g}[\mathit{self}] : \\
& \quad \land\ \mathit{key\_to\_be\_served}' = \mathit{key} \\
& \quad \land\ \mathit{service\_request}' = \text{“results”} \\
& \quad \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{pc}] \\
& \quad \land\ \mathit{keys\_g}' = [\mathit{keys\_g}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{keys\_g}] \\
& \quad \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Tail}(\mathit{stack}[\mathit{self}])] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{scan\_tasks},\ \mathit{response}, \\
& \qquad\qquad\qquad \mathit{scan\_task\_status}, \\
& \qquad\qquad\qquad \mathit{number\_of\_batches},\ \mathit{counter}, \\
& \qquad\qquad\qquad \mathit{keys\_},\ \mathit{keys\_c},\ \mathit{keys\_ge},\ \mathit{keys\_r}, \\
& \qquad\qquad\qquad \mathit{keys\_d},\ \mathit{keys},\ \mathit{inner\_state}\rangle
\end{array}$$

$$\mathit{get\_results}(\mathit{self}) \triangleq \mathit{ResultsServiceRequest}(\mathit{self})$$

$$\begin{array}{ll}
\mathit{ClearServiceRequest}(\mathit{self}) \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“ClearServiceRequest”} \\
& \land\ \exists\, \mathit{key} \in \mathit{keys\_c}[\mathit{self}] : \\
& \quad \land\ \mathit{key\_to\_be\_served}' = \mathit{key} \\
& \quad \land\ \mathit{service\_request}' = \text{“clear”} \\
& \quad \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{pc}] \\
& \quad \land\ \mathit{keys\_c}' = [\mathit{keys\_c}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{keys\_c}] \\
& \quad \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Tail}(\mathit{stack}[\mathit{self}])] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{scan\_tasks},\ \mathit{response}, \\
& \qquad\qquad\qquad \mathit{scan\_task\_status}, \\
& \qquad\qquad\qquad \mathit{number\_of\_batches},\ \mathit{counter},\ \mathit{keys\_}, \\
& \qquad\qquad\qquad \mathit{keys\_g},\ \mathit{keys\_ge},\ \mathit{keys\_r},\ \mathit{keys\_d}, \\
& \qquad\qquad\qquad \mathit{keys},\ \mathit{inner\_state}\rangle
\end{array}$$

$$\mathit{clear\_results}(\mathit{self}) \triangleq \mathit{ClearServiceRequest}(\mathit{self})$$

$$\begin{array}{ll}
\mathit{StatusServiceRequest}(\mathit{self}) \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“StatusServiceRequest”} \\
& \land\ \exists\, \mathit{key} \in \mathit{keys\_ge}[\mathit{self}] : \\
& \quad \land\ \mathit{key\_to\_be\_served}' = \mathit{key} \\
& \quad \land\ \mathit{service\_request}' = \text{“status”} \\
& \quad \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{pc}] \\
& \quad \land\ \mathit{keys\_ge}' = [\mathit{keys\_ge}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{keys\_ge}] \\
& \quad \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Tail}(\mathit{stack}[\mathit{self}])] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{scan\_tasks},\ \mathit{response}, \\
& \qquad\qquad\qquad \mathit{scan\_task\_status}, \\
& \qquad\qquad\qquad \mathit{number\_of\_batches},\ \mathit{counter}, \\
& \qquad\qquad\qquad \mathit{keys\_},\ \mathit{keys\_g},\ \mathit{keys\_c},\ \mathit{keys\_r}, \\
& \qquad\qquad\qquad \mathit{keys\_d},\ \mathit{keys},\ \mathit{inner\_state}\rangle
\end{array}$$

$$get\_status(self) \triangleq StatusServiceRequest(self)$$

$$RegisterServiceRequest(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“RegisterServiceRequest”} \\
\land\ \exists\, key \in keys\_r[self] : \\
\qquad \land\ key\_to\_be\_served' = key \\
\qquad \land\ service\_request' = \text{“register”} \\
\qquad \land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
\qquad \land\ keys\_r' = [keys\_r \text{ EXCEPT } ![self] = Head(stack[self]).keys\_r] \\
\qquad \land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
\land\ \text{UNCHANGED } \langle scan\_tasks, response, \\
\qquad\qquad\qquad scan\_task\_status, \\
\qquad\qquad\qquad number\_of\_batches, counter, \\
\qquad\qquad\qquad keys\_, keys\_g, keys\_c, keys\_ge, \\
\qquad\qquad\qquad keys\_d, keys, inner\_state\rangle
\end{array}$$

$$register\_keys(self) \triangleq RegisterServiceRequest(self)$$

$$DeleteServiceRequest(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“DeleteServiceRequest”} \\
\land\ \exists\, key \in keys\_d[self] : \\
\qquad \land\ key\_to\_be\_served' = key \\
\qquad \land\ service\_request' = \text{“delete”} \\
\qquad \land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
\qquad \land\ keys\_d' = [keys\_d \text{ EXCEPT } ![self] = Head(stack[self]).keys\_d] \\
\qquad \land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
\land\ \text{UNCHANGED } \langle scan\_tasks, response, \\
\qquad\qquad\qquad scan\_task\_status, \\
\qquad\qquad\qquad number\_of\_batches, counter, \\
\qquad\qquad\qquad keys\_, keys\_g, keys\_c, keys\_ge, \\
\qquad\qquad\qquad keys\_r, keys, inner\_state\rangle
\end{array}$$

$$delete\_keys(self) \triangleq DeleteServiceRequest(self)$$

$$RegisterServiceRequestFromScan(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“RegisterServiceRequestFromScan”} \\
\land\ \exists\, key \in keys[self] : \\
\qquad \land\ key\_to\_be\_served' = key \\
\qquad \land\ service\_request' = \text{“register”} \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“ResultsServiceRequestFromScan”}] \\
\land\ \text{UNCHANGED } \langle scan\_tasks, response, \\
\qquad\qquad\qquad scan\_task\_status, \\
\qquad\qquad\qquad number\_of\_batches, \\
\qquad\qquad\qquad counter, stack, keys\_, \\
\qquad\qquad\qquad keys\_g, keys\_c, \\
\qquad\qquad\qquad keys\_ge, keys\_r, \\
\qquad\qquad\qquad keys\_d, keys, \\
\qquad\qquad\qquad inner\_state\rangle
\end{array}$$

$$ResultsServiceRequestFromScan(self) \triangleq \land\ pc[self] = \text{“ResultsServiceRequestFromScan”}$$

```
                                 $\land \exists\, key \in keys[self]$ :
                                     $\land\ key\_to\_be\_served' = key$
                                     $\land\ service\_request' =$ “results”
                                 $\land\ pc' = [pc$ EXCEPT $![self] =$ “SubscribeServiceRequestFromScan”
                                 $\land$ UNCHANGED $\langle scan\_tasks, response,$
                                                  $scan\_task\_status,$
                                                  $number\_of\_batches,$
                                                  $counter, stack, keys\_,$
                                                  $keys\_g, keys\_c, keys\_ge,$
                                                  $keys\_r, keys\_d, keys,$
                                                  $inner\_state\rangle$

$SubscribeServiceRequestFromScan(self) \triangleq$ $\land\ pc[self] =$ “SubscribeServiceRequestFromScan”
                                                  $\land \exists\, key \in keys[self]$ :
                                                      $\land\ key\_to\_be\_served' = key$
                                                      $\land\ service\_request' =$ “subscribe”
                                                      $\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
                                                      $\land\ keys' = [keys$ EXCEPT $![self] = Head(stack[self]).keys]$
                                                      $\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
                                                  $\land$ UNCHANGED $\langle scan\_tasks, response,$
                                                                   $scan\_task\_status,$
                                                                   $number\_of\_batches,$
                                                                   $counter, keys\_,$
                                                                   $keys\_g, keys\_c,$
                                                                   $keys\_ge, keys\_r,$
                                                                   $keys\_d, inner\_state\rangle$

$scan(self) \triangleq RegisterServiceRequestFromScan(self)$
                $\lor\ ResultsServiceRequestFromScan(self)$
                $\lor\ SubscribeServiceRequestFromScan(self)$

$Services \triangleq$ $\land\ pc[$“SERVICES”$] =$ “Services”
             $\land$ IF $service\_request =$ “info”
                   THEN $\land\ pc' = [pc$ EXCEPT $![$“SERVICES”$] =$ “Info”$]$
                   ELSE $\land$ IF $service\_request =$ “results”
                              THEN $\land\ pc' = [pc$ EXCEPT $![$“SERVICES”$] =$ “Results”$]$
                              ELSE $\land$ IF $service\_request =$ “clear”
                                         THEN $\land\ pc' = [pc$ EXCEPT $![$“SERVICES”$] =$ “Clear”$]$
                                         ELSE $\land$ IF $service\_request =$ “status”
                                                    THEN $\land\ pc' = [pc$ EXCEPT $![$“SERVICES”$] =$ “Sta
                                                    ELSE $\land$ IF $service\_request =$ “register”
                                                               THEN $\land\ pc' = [pc$ EXCEPT $![$“SERVIC
                                                               ELSE $\land$ IF $service\_request =$ “delete”
                                                                          THEN $\land\ pc' = [pc$ EXCEP
                                                                          ELSE $\land$ IF $service\_reque$
                                                                                     THEN $\land\ pc'$
```

```
                                                                        ELSE  $\land$ $pc'$
         $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                           $key\_to\_be\_served, service\_request,$
                           $number\_of\_batches, counter, stack, keys\_, keys\_g,$
                           $keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$Info \triangleq$ $\land$ $pc[$“SERVICES”$]$ = “Info”
          $\land$ $response' = info\_response$
          $\land$ $pc' = [pc$ EXCEPT $![$“SERVICES”$]$ = “ServicesLoop”$]$
          $\land$ UNCHANGED $\langle scan\_tasks, scan\_task\_status, key\_to\_be\_served,$
                        $service\_request, number\_of\_batches, counter, stack,$
                        $keys\_, keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
                        $inner\_state\rangle$

$Results \triangleq$ $\land$ $pc[$“SERVICES”$]$ = “Results”
             $\land$ IF $key\_to\_be\_served \in scan\_tasks$
                  THEN $\land$ $response' = results\_response$
                  ELSE $\land$ $response'$ = “Error: key not found.”
             $\land$ $pc' = [pc$ EXCEPT $![$“SERVICES”$]$ = “ServicesLoop”$]$
             $\land$ UNCHANGED $\langle scan\_tasks, scan\_task\_status, key\_to\_be\_served,$
                           $service\_request, number\_of\_batches, counter, stack,$
                           $keys\_, keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d,$
                           $keys, inner\_state\rangle$

$Clear \triangleq$ $\land$ $pc[$“SERVICES”$]$ = “Clear”
           $\land$ IF $key\_to\_be\_served \in scan\_tasks$
                THEN $\land$ $response' = clear\_response$
                ELSE $\land$ $response'$ = “Error: key not found.”
           $\land$ $pc' = [pc$ EXCEPT $![$“SERVICES”$]$ = “ServicesLoop”$]$
           $\land$ UNCHANGED $\langle scan\_tasks, scan\_task\_status, key\_to\_be\_served,$
                         $service\_request, number\_of\_batches, counter, stack,$
                         $keys\_, keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
                         $inner\_state\rangle$

$Status \triangleq$ $\land$ $pc[$“SERVICES”$]$ = “Status”
            $\land$ IF $key\_to\_be\_served \in scan\_tasks$
                 THEN $\land$ $response' = status\_response$
                 ELSE $\land$ $response'$ = “Error: key not found.”
            $\land$ $pc' = [pc$ EXCEPT $![$“SERVICES”$]$ = “ServicesLoop”$]$
            $\land$ UNCHANGED $\langle scan\_tasks, scan\_task\_status, key\_to\_be\_served,$
                          $service\_request, number\_of\_batches, counter, stack,$
                          $keys\_, keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
                          $inner\_state\rangle$

$Register$   $\triangleq$ $\land$ $pc[$“SERVICES”$]$ = “Register”
               $\land$ IF $key\_to\_be\_served \in scan\_tasks$
            THEN  $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"KeyError"}]$
            ELSE  $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"Success"}]$
      $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                       $key\_to\_be\_served, service\_request,$
                       $number\_of\_batches, counter, stack, keys\_, keys\_g,$
                       $keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$KeyError \triangleq$ $\land\ pc[\text{"SERVICES"}] = \text{"KeyError"}$
      $\land\ response' = \text{"Error: key already in scan task."}$
      $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"ServicesLoop"}]$
      $\land$ UNCHANGED $\langle scan\_tasks, scan\_task\_status, key\_to\_be\_served,$
                       $service\_request, number\_of\_batches, counter, stack,$
                       $keys\_, keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d,$
                       $keys, inner\_state\rangle$

$Success \triangleq$ $\land\ pc[\text{"SERVICES"}] = \text{"Success"}$
     $\land\ scan\_task\_status' = \text{"adding"}$
     $\land\ response' = register\_response$
     $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"ServicesLoop"}]$
     $\land$ UNCHANGED $\langle scan\_tasks, key\_to\_be\_served, service\_request,$
                      $number\_of\_batches, counter, stack, keys\_, keys\_g,$
                      $keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$Delete \triangleq$ $\land\ pc[\text{"SERVICES"}] = \text{"Delete"}$
    $\land$ IF $key\_to\_be\_served \in scan\_tasks$
          THEN  $\land\ scan\_task\_status' = \text{"deleting"}$
                $\land\ response' = delete\_response$
          ELSE  $\land\ response' = \text{"Error: key not found."}$
                $\land$ UNCHANGED $scan\_task\_status$
    $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"ServicesLoop"}]$
    $\land$ UNCHANGED $\langle scan\_tasks, key\_to\_be\_served, service\_request,$
                     $number\_of\_batches, counter, stack, keys\_, keys\_g,$
                     $keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$Subscribe \triangleq$ $\land\ pc[\text{"SERVICES"}] = \text{"Subscribe"}$
      $\land$ IF $key\_to\_be\_served \in scan\_tasks$
            THEN  $\land\ response' = subscribe\_response$
            ELSE  $\land\ response' = \text{"Error: key not found."}$
      $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"ServicesLoop"}]$
      $\land$ UNCHANGED $\langle scan\_tasks, scan\_task\_status, key\_to\_be\_served,$
                       $service\_request, number\_of\_batches, counter,$
                       $stack, keys\_, keys\_g, keys\_c, keys\_ge, keys\_r,$
                       $keys\_d, keys, inner\_state\rangle$

$ServicesLoop \triangleq$ $\land\ pc[\text{"SERVICES"}] = \text{"ServicesLoop"}$
         $\land\ pc' = [pc$ EXCEPT $![\text{"SERVICES"}] = \text{"Services"}]$
```

```
                $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                                  $key\_to\_be\_served, service\_request,$
                                  $number\_of\_batches, counter, stack, keys\_,$
                                  $keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
                                  $inner\_state\rangle$

$services \triangleq Services \lor Info \lor Results \lor Clear \lor Status \lor Register$
              $\lor KeyError \lor Success \lor Delete \lor Subscribe$
              $\lor ServicesLoop$

$GetScanTasks \triangleq$ $\land pc[$“SCAN TASK”$] =$ “GetScanTasks”
                 $\land inner\_state' = scan\_tasks$
                 $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “ScanTask”$]$
                 $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                                   $key\_to\_be\_served, service\_request,$
                                   $number\_of\_batches, counter, stack, keys\_,$
                                   $keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys\rangle$

$ScanTask \triangleq$ $\land pc[$“SCAN TASK”$] =$ “ScanTask”
             $\land PrintT(Cardinality(scan\_tasks))$
             $\land$ IF $Cardinality(scan\_tasks) > MaxScanTasks$
                  THEN $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “BoundError”$]$
                  ELSE $\land$ IF $scan\_task\_status =$ “adding”
                            THEN $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “Adding”$]$
                            ELSE $\land$ IF $scan\_task\_status =$ “deleting”
                                      THEN $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “Deleting”$]$
                                      ELSE $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “StoreScanTas
             $\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
                               $key\_to\_be\_served, service\_request,$
                               $number\_of\_batches, counter, stack, keys\_, keys\_g,$
                               $keys\_c, keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$BoundError \triangleq$ $\land pc[$“SCAN TASK”$] =$ “BoundError”
               $\land response' =$ “Error: max scan tasks reached.”
               $\land scan\_task\_status' =$ “waiting”
               $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “StoreScanTasks”$]$
               $\land$ UNCHANGED $\langle scan\_tasks, key\_to\_be\_served, service\_request,$
                                 $number\_of\_batches, counter, stack, keys\_, keys\_g,$
                                 $keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
                                 $inner\_state\rangle$

$Adding \triangleq$ $\land pc[$“SCAN TASK”$] =$ “Adding”
           $\land inner\_state' = (inner\_state \cup \{key\_to\_be\_served\})$
           $\land scan\_task\_status' =$ “waiting”
           $\land pc' = [pc$ EXCEPT $![$“SCAN TASK”$] =$ “StoreScanTasks”$]$
           $\land$ UNCHANGED $\langle scan\_tasks, response, key\_to\_be\_served,$
```

$$\begin{array}{l}
\qquad\qquad\qquad \mathit{service\_request},\ \mathit{number\_of\_batches},\ \mathit{counter},\ \mathit{stack},\\
\qquad\qquad\qquad \mathit{keys\_},\ \mathit{keys\_g},\ \mathit{keys\_c},\ \mathit{keys\_ge},\ \mathit{keys\_r},\ \mathit{keys\_d},\ \mathit{keys}\rangle
\end{array}$$

$$\begin{array}{lll}
\mathit{Deleting} & \triangleq & \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“Deleting”}\\
 & & \land\ \mathit{scan\_tasks}' = \mathit{scan\_tasks} \setminus \{\mathit{key\_to\_be\_served}\}\\
 & & \land\ \mathit{scan\_task\_status}' = \text{“waiting”}\\
 & & \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“SCAN TASK”}] = \text{“StoreScanTasks”}]\\
 & & \land\ \text{UNCHANGED}\ \langle \mathit{response},\ \mathit{key\_to\_be\_served},\ \mathit{service\_request},\\
 & & \qquad\qquad \mathit{number\_of\_batches},\ \mathit{counter},\ \mathit{stack},\ \mathit{keys\_},\ \mathit{keys\_g},\\
 & & \qquad\qquad \mathit{keys\_c},\ \mathit{keys\_ge},\ \mathit{keys\_r},\ \mathit{keys\_d},\ \mathit{keys},\ \mathit{inner\_state}\rangle
\end{array}$$

$$\begin{array}{lll}
\mathit{StoreScanTasks} & \triangleq & \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“StoreScanTasks”}\\
 & & \land\ \mathit{scan\_tasks}' = \mathit{inner\_state}\\
 & & \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“SCAN TASK”}] = \text{“ScanTaskLoop”}]\\
 & & \land\ \text{UNCHANGED}\ \langle \mathit{response},\ \mathit{scan\_task\_status},\ \mathit{key\_to\_be\_served},\\
 & & \qquad\qquad \mathit{service\_request},\ \mathit{number\_of\_batches},\ \mathit{counter},\\
 & & \qquad\qquad \mathit{stack},\ \mathit{keys\_},\ \mathit{keys\_g},\ \mathit{keys\_c},\ \mathit{keys\_ge},\\
 & & \qquad\qquad \mathit{keys\_r},\ \mathit{keys\_d},\ \mathit{keys},\ \mathit{inner\_state}\rangle
\end{array}$$

$$\begin{array}{lll}
\mathit{ScanTaskLoop} & \triangleq & \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“ScanTaskLoop”}\\
 & & \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“SCAN TASK”}] = \text{“ScanTask”}]\\
 & & \land\ \text{UNCHANGED}\ \langle \mathit{scan\_tasks},\ \mathit{response},\ \mathit{scan\_task\_status},\\
 & & \qquad\qquad \mathit{key\_to\_be\_served},\ \mathit{service\_request},\\
 & & \qquad\qquad \mathit{number\_of\_batches},\ \mathit{counter},\ \mathit{stack},\ \mathit{keys\_},\\
 & & \qquad\qquad \mathit{keys\_g},\ \mathit{keys\_c},\ \mathit{keys\_ge},\ \mathit{keys\_r},\ \mathit{keys\_d},\ \mathit{keys},\\
 & & \qquad\qquad \mathit{inner\_state}\rangle
\end{array}$$

$$\begin{array}{lll}
\mathit{scantask} & \triangleq & \mathit{GetScanTasks} \lor \mathit{ScanTask} \lor \mathit{BoundError} \lor \mathit{Adding} \lor \mathit{Deleting}\\
 & & \quad \lor\ \mathit{StoreScanTasks} \lor \mathit{ScanTaskLoop}
\end{array}$$

$$\begin{array}{lll}
\mathit{ConfigGuard} & \triangleq & \land\ \mathit{pc}[\text{“MAIN”}] = \text{“ConfigGuard”}\\
 & & \land\ \text{IF}\ \ \mathit{ConfigViewingKeys} \neq \{\}\\
 & & \qquad \text{THEN}\ \ \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“MAIN”}] = \text{“FromZebradConfig”}]\\
 & & \qquad \text{ELSE}\ \ \ \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“MAIN”}] = \text{“ListeningGuard”}]\\
 & & \land\ \text{UNCHANGED}\ \langle \mathit{scan\_tasks},\ \mathit{response},\ \mathit{scan\_task\_status},\\
 & & \qquad\qquad \mathit{key\_to\_be\_served},\ \mathit{service\_request},\\
 & & \qquad\qquad \mathit{number\_of\_batches},\ \mathit{counter},\ \mathit{stack},\ \mathit{keys\_},\\
 & & \qquad\qquad \mathit{keys\_g},\ \mathit{keys\_c},\ \mathit{keys\_ge},\ \mathit{keys\_r},\ \mathit{keys\_d},\ \mathit{keys},\\
 & & \qquad\qquad \mathit{inner\_state}\rangle
\end{array}$$

$$\begin{array}{lll}
\mathit{FromZebradConfig} & \triangleq & \land\ \mathit{pc}[\text{“MAIN”}] = \text{“FromZebradConfig”}\\
 & & \land\ \land\ \mathit{keys\_}' = [\mathit{keys\_}\ \text{EXCEPT}\ ![\text{“MAIN”}] = \mathit{ConfigViewingKeys}]\\
 & & \quad\ \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\text{“MAIN”}] = \langle [\begin{array}{lll} \mathit{procedure} & \mapsto & \text{“add\_config\_keys”},\\ \mathit{pc} & \mapsto & \text{“ListeningGuard”},\\ \mathit{keys\_} & \mapsto & \mathit{keys\_}[\text{“MAIN”}]]\rangle \end{array}\\
 & & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\ \mathit{stack}[\text{“MAIN”}]]\\
 & & \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“MAIN”}] = \text{“AddConfigKeys”}]
\end{array}$$

```
                          ∧ UNCHANGED ⟨scan_tasks, response, scan_task_status,
                                       key_to_be_served, service_request,
                                       number_of_batches, counter, keys_g, keys_c,
                                       keys_ge, keys_r, keys_d, keys, inner_state⟩

ListeningGuard ≜ ∧ pc["MAIN"] = "ListeningGuard"
                 ∧ IF GrpcViewingKeys ≠ ⟨⟩
                       THEN ∧ pc' = [pc EXCEPT !["MAIN"] = "GetTotalIterationsToMake"]
                       ELSE ∧ pc' = [pc EXCEPT !["MAIN"] = "End"]
                 ∧ UNCHANGED ⟨scan_tasks, response, scan_task_status,
                              key_to_be_served, service_request,
                              number_of_batches, counter, stack, keys_,
                              keys_g, keys_c, keys_ge, keys_r, keys_d,
                              keys, inner_state⟩

GetTotalIterationsToMake ≜ ∧ pc["MAIN"] = "GetTotalIterationsToMake"
                           ∧ stack' = [stack EXCEPT !["MAIN"] = ⟨[procedure ↦  "get_config_number
                                                                  pc        ↦  "ListeningMode"]⟩
                                                                  ∘ stack["MAIN"]]
                           ∧ pc' = [pc EXCEPT !["MAIN"] = "CheckBatch1"]
                           ∧ UNCHANGED ⟨scan_tasks, response,
                                        scan_task_status, key_to_be_served,
                                        service_request, number_of_batches,
                                        counter, keys_, keys_g, keys_c,
                                        keys_ge, keys_r, keys_d, keys,
                                        inner_state⟩

ListeningMode ≜ ∧ pc["MAIN"] = "ListeningMode"
                ∧ IF counter  ≤ number_of_batches
                      THEN ∧ pc' = [pc EXCEPT !["MAIN"] = "GetInfoCall"]
                      ELSE ∧ pc' = [pc EXCEPT !["MAIN"] = "End"]
                ∧ UNCHANGED ⟨scan_tasks, response, scan_task_status,
                             key_to_be_served, service_request,
                             number_of_batches, counter, stack, keys_,
                             keys_g, keys_c, keys_ge, keys_r, keys_d, keys,
                             inner_state⟩

GetInfoCall ≜ ∧ pc["MAIN"] = "GetInfoCall"
              ∧ stack' = [stack EXCEPT !["MAIN"] = ⟨[procedure ↦  "get_info",
                                                     pc        ↦  "RegisterKeysCall"]⟩
                                                     ∘ stack["MAIN"]]
              ∧ pc' = [pc EXCEPT !["MAIN"] = "InfoServiceRequest"]
              ∧ UNCHANGED ⟨scan_tasks, response, scan_task_status,
                           key_to_be_served, service_request,
                           number_of_batches, counter, keys_, keys_g,
                           keys_c, keys_ge, keys_r, keys_d, keys,
```

$$\qquad\qquad inner\_state\rangle$$

$$\begin{array}{lll}
RegisterKeysCall \triangleq & \land\, pc[\text{“MAIN”}] = \text{“RegisterKeysCall”} & \\
 & \land\, \land\, keys\_r' = [keys\_r \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]] & \\
 & \quad \land\, stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle[procedure \mapsto \text{“register\_keys”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \mapsto \text{“GetStatusCall”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad keys\_r \mapsto keys\_r[\text{“MAIN”}]]\rangle & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\, stack[\text{“MAIN”}]] & \\
 & \land\, pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“RegisterServiceRequest”}] & \\
 & \land\, \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status, & \\
 & \qquad key\_to\_be\_served, service\_request, & \\
 & \qquad number\_of\_batches, counter, keys\_, keys\_g, & \\
 & \qquad keys\_c, keys\_ge, keys\_d, keys, inner\_state\rangle &
\end{array}$$

$$\begin{array}{lll}
GetStatusCall \triangleq & \land\, pc[\text{“MAIN”}] = \text{“GetStatusCall”} & \\
 & \land\, \land\, keys\_ge' = [keys\_ge \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]] & \\
 & \quad \land\, stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle[procedure \mapsto \text{“get\_status”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \mapsto \text{“GetResultsCall”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad keys\_ge \mapsto keys\_ge[\text{“MAIN”}]]\rangle & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\, stack[\text{“MAIN”}]] & \\
 & \land\, pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“StatusServiceRequest”}] & \\
 & \land\, \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status, & \\
 & \qquad key\_to\_be\_served, service\_request, & \\
 & \qquad number\_of\_batches, counter, keys\_, keys\_g, & \\
 & \qquad keys\_c, keys\_r, keys\_d, keys, inner\_state\rangle &
\end{array}$$

$$\begin{array}{lll}
GetResultsCall \triangleq & \land\, pc[\text{“MAIN”}] = \text{“GetResultsCall”} & \\
 & \land\, \land\, keys\_g' = [keys\_g \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]] & \\
 & \quad \land\, stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle[procedure \mapsto \text{“get\_results”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \mapsto \text{“ClearResultsCall”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad keys\_g \mapsto keys\_g[\text{“MAIN”}]]\rangle & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\, stack[\text{“MAIN”}]] & \\
 & \land\, pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“ResultsServiceRequest”}] & \\
 & \land\, \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status, & \\
 & \qquad key\_to\_be\_served, service\_request, & \\
 & \qquad number\_of\_batches, counter, keys\_, keys\_c, & \\
 & \qquad keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle &
\end{array}$$

$$\begin{array}{lll}
ClearResultsCall \triangleq & \land\, pc[\text{“MAIN”}] = \text{“ClearResultsCall”} & \\
 & \land\, \land\, keys\_c' = [keys\_c \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]] & \\
 & \quad \land\, stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle[procedure \mapsto \text{“clear\_results”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \mapsto \text{“DeleteKeysCall”}, & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad keys\_c \mapsto keys\_c[\text{“MAIN”}]]\rangle & \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\, stack[\text{“MAIN”}]] & \\
 & \land\, pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“ClearServiceRequest”}] & \\
 & \land\, \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status, &
\end{array}$$

$$\begin{array}{l} key\_to\_be\_served,\ service\_request,\\ number\_of\_batches,\ counter,\ keys\_,\ keys\_g,\\ keys\_ge,\ keys\_r,\ keys\_d,\ keys,\ inner\_state\rangle \end{array}$$

$$\begin{array}{l} DeleteKeysCall \triangleq \land\ pc[\text{“MAIN”}] = \text{“DeleteKeysCall”}\\ \quad \land\ \land\ keys\_d' = [keys\_d \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]]\\ \quad\quad \land\ stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle[procedure \mapsto \text{“delete\_keys”},\\ \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \ pc \mapsto \text{“ScanCall”},\\ \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \ keys\_d \mapsto keys\_d[\text{“MAIN”}]]\rangle\\ \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \ \circ\ stack[\text{“MAIN”}]]\\ \quad \land\ pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“DeleteServiceRequest”}]\\ \quad \land\ \text{UNCHANGED } \langle scan\_tasks,\ response,\ scan\_task\_status,\\ \quad\quad\quad key\_to\_be\_served,\ service\_request,\\ \quad\quad\quad number\_of\_batches,\ counter,\ keys\_,\ keys\_g,\\ \quad\quad\quad keys\_c,\ keys\_ge,\ keys\_r,\ keys,\ inner\_state\rangle \end{array}$$

$$\begin{array}{l} ScanCall \triangleq \land\ pc[\text{“MAIN”}] = \text{“ScanCall”}\\ \quad \land\ \land\ keys' = [keys \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]]\\ \quad\quad \land\ stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle[procedure \mapsto \text{“scan”},\\ \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \ pc \mapsto \text{“IncrementCounter”},\\ \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \ keys \mapsto keys[\text{“MAIN”}]]\rangle\\ \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\ \ \circ\ stack[\text{“MAIN”}]]\\ \quad \land\ pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“RegisterServiceRequestFromScan”}]\\ \quad \land\ \text{UNCHANGED } \langle scan\_tasks,\ response,\ scan\_task\_status,\\ \quad\quad\quad key\_to\_be\_served,\ service\_request,\\ \quad\quad\quad number\_of\_batches,\ counter,\ keys\_,\ keys\_g,\ keys\_c,\\ \quad\quad\quad keys\_ge,\ keys\_r,\ keys\_d,\ inner\_state\rangle \end{array}$$

$$\begin{array}{l} IncrementCounter \triangleq \land\ pc[\text{“MAIN”}] = \text{“IncrementCounter”}\\ \quad \land\ counter' = counter + 1\\ \quad \land\ pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“ListeningMode”}]\\ \quad \land\ \text{UNCHANGED } \langle scan\_tasks,\ response,\ scan\_task\_status,\\ \quad\quad\quad key\_to\_be\_served,\ service\_request,\\ \quad\quad\quad number\_of\_batches,\ stack,\ keys\_,\ keys\_g,\\ \quad\quad\quad keys\_c,\ keys\_ge,\ keys\_r,\ keys\_d,\ keys,\\ \quad\quad\quad inner\_state\rangle \end{array}$$

$$\begin{array}{l} End \triangleq \land\ pc[\text{“MAIN”}] = \text{“End”}\\ \quad \land\ \text{TRUE}\\ \quad \land\ pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“Done”}]\\ \quad \land\ \text{UNCHANGED } \langle scan\_tasks,\ response,\ scan\_task\_status,\\ \quad\quad\quad key\_to\_be\_served,\ service\_request,\ number\_of\_batches,\\ \quad\quad\quad counter,\ stack,\ keys\_,\ keys\_g,\ keys\_c,\ keys\_ge,\ keys\_r,\\ \quad\quad\quad keys\_d,\ keys,\ inner\_state\rangle \end{array}$$

$$Main \triangleq ConfigGuard \lor FromZebradConfig \lor ListeningGuard$$

```
                $\lor$ $GetTotalIterationsToMake$ $\lor$ $ListeningMode$ $\lor$ $GetInfoCall$
                $\lor$ $RegisterKeysCall$   $\lor$ $GetStatusCall$ $\lor$ $GetResultsCall$
                $\lor$ $ClearResultsCall$   $\lor$ $DeleteKeysCall$ $\lor$ $ScanCall$
                $\lor$ $IncrementCounter$ $\lor$ $End$

 Allow infinite stuttering to prevent deadlock on termination.
$Terminating \triangleq$ $\land \forall self \in ProcSet : pc[self] =$ "Done"
                 $\land$ UNCHANGED $vars$

$Next \triangleq$ $services \lor scantask \lor Main$
             $\lor (\exists self \in ProcSet :$  $\lor$ $get\_config\_number\_of\_batches(self)$
                                         $\lor$ $add\_config\_keys(self) \lor get\_info(self)$
                                         $\lor$ $get\_results(self) \lor clear\_results(self)$
                                         $\lor$ $get\_status(self) \lor register\_keys(self)$
                                         $\lor$ $delete\_keys(self) \lor scan(self))$
             $\lor$ $Terminating$

$Spec \triangleq$ $Init \land \Box[Next]_{vars}$

$Termination \triangleq$ $\Diamond(\forall self \in ProcSet : pc[self] =$ "Done"$)$

 END TRANSLATION
```

\ * Modification History
\ * Last modified Thu Mar 07 18 : 13 : 46 UYT 2024 by alfredo
\ * Created Wed Feb 21 10 : 40 : 53 UYT 2024 by alfredo