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. Liveness properties
5. Utility Functions
6. *gRPC* Methods
7. Services Process
8. Scan Task Process
9. Main Program Process

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

EXTENDS TLC, $Integers$, $Sequences$, $Randomization$, $FiniteSets$, $Json$

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*.

$info\_response \triangleq ToJson([saplingheight \mapsto 1])$

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

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

An empty response to Register

$register\_response \triangleq ToJson([empty \mapsto \{\}])$

An empty response to Delete

$delete\_response \triangleq ToJson([empty \mapsto \{\}])$

An empty response to Clear

$clear\_response \triangleq ToJson([empty \mapsto \{\}])$

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

$subscribe\_response \triangleq ToJson([empty \mapsto \{\}])$

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

$status\_response \triangleq ToJson([empty \mapsto \{\}])$

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.

$service\_requests \triangleq \{\text{“waiting”}, \text{“info”}, \text{“results”}, \text{“clear”}, \text{“status”}, \text{“register”}, \text{“delete”}, \text{“subscribe”}\}$

--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.

$service\_request = \text{“waiting”}$;

The number of batches the configuration has.

$number\_of\_batches = 0$;

The counter for the number of batches.

$counter = 1$;

define

THE TYPE INVARIANT :

$TypeInvariant \triangleq$

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.

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

LIVENESS PROPERTIES :

$ScanTaskLiveness \triangleq$

```
        The ScanTask process always reaches a waiting state.
        ◇(scan_task_status = "waiting")
    ServiceLiveness ≜
        The Services process always reaches a waiting state.
        ◇(service_request = "waiting")
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 ≠ {} then
            number_of_batches := number_of_batches + 1 ;
        end if ;
    CheckBatch2:
        if GrpcViewingKeysBatch2 ≠ {} then
            number_of_batches := number_of_batches + 1 ;
        end if ;
    CheckBatch3:
        if GrpcViewingKeysBatch3 ≠ {} 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 ∈ 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 ∈ 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 ∈ 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 ∈ 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 ∈ 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 ∈ 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 ;
ClearRequestFlag:
    service_request := "waiting" ;
```

```
    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.
fair 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 ∪ {key_to_be_served} ;
        scan_task_status := "waiting" ;
    elsif scan_task_status = "deleting" then
      Deleting:
        scan_tasks := scan_tasks \ {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 ≠ {} then
      FromZebradConfig:
        call add_config_keys(ConfigViewingKeys) ;
    end if ;
  ListeningGuard:
    if GrpcViewingKeys ≠ ⟨⟩ then
```

```
            GetTotalIterationsToMake:
                call get_config_number_of_batches() ;
            ListeningMode:
                while counter ≤ 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) = "f6c668a3" ∧ chksum(tla) = "868960ef")
  Parameter keys of procedure add_config_keys at line 131 col 27 changed to keys_
  Parameter keys of procedure get_results at line 152 col 23 changed to keys_g
  Parameter keys of procedure clear_results at line 163 col 25 changed to keys_c
  Parameter keys of procedure get_status at line 174 col 22 changed to keys_ge
  Parameter keys of procedure register_keys at line 185 col 25 changed to keys_r
  Parameter keys of procedure delete_keys at line 196 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 ≜

    ∧ response ∈ STRING

    ∧ scan_task_status ∈ scan_task_statuses
```

$\land\ key\_to\_be\_served \in \text{STRING}$

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

$ScanTaskLiveness \triangleq$

$\Diamond(scan\_task\_status = \text{“waiting”})$

$ServiceLiveness \triangleq$

$\Diamond(service\_request = \text{“waiting”})$

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

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

$ProcSet \triangleq \{\text{“SERVICES”}\} \cup \{\text{“SCAN TASK”}\} \cup \{\text{“MAIN”}\}$

$Init \triangleq$ *Global variables*
$\land\ scan\_tasks = \{\}$
$\land\ response = \text{“”}$
$\land\ scan\_task\_status = \text{“waiting”}$
$\land\ key\_to\_be\_served = \text{“”}$
$\land\ service\_request = \text{“waiting”}$
$\land\ number\_of\_batches = 0$
$\land\ counter = 1$
Procedure add_config_keys
$\land\ keys\_ = [self \in ProcSet \mapsto defaultInitValue]$
Procedure get_results
$\land\ keys\_g = [self \in ProcSet \mapsto defaultInitValue]$
Procedure clear_results
$\land\ keys\_c = [self \in ProcSet \mapsto defaultInitValue]$
Procedure get_status
$\land\ keys\_ge = [self \in ProcSet \mapsto defaultInitValue]$
Procedure register_keys
$\land\ keys\_r = [self \in ProcSet \mapsto defaultInitValue]$
Procedure delete_keys
$\land\ keys\_d = [self \in ProcSet \mapsto defaultInitValue]$
Procedure scan
$\land\ keys = [self \in ProcSet \mapsto defaultInitValue]$
Process scantask
$\land\ inner\_state = \{\}$
$\land\ stack = [self \in ProcSet \mapsto \langle\rangle]$
$\land\ pc = [self \in ProcSet \mapsto \text{CASE } self = \text{“SERVICES”} \to \text{“Services”}$
$\square\ self = \text{“SCAN TASK”} \to \text{“GetScanTasks”}$
$\square\ self = \text{“MAIN”} \to \text{“ConfigGuard”}]$

$$
\begin{aligned}
CheckBatch1(self) \triangleq\ & \land\ pc[self] = \text{“CheckBatch1”} \\
& \land\ \text{IF}\ GrpcViewingKeysBatch1 \neq \{\} \\
& \qquad \text{THEN}\ \land\ number\_of\_batches' = number\_of\_batches + 1 \\
& \qquad \text{ELSE}\ \land\ \text{TRUE} \\
& \qquad\qquad\ \land\ \text{UNCHANGED}\ number\_of\_batches \\
& \land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“CheckBatch2”}] \\
& \land\ \text{UNCHANGED}\ \langle scan\_tasks,\ response,\ scan\_task\_status, \\
& \qquad\qquad\qquad key\_to\_be\_served,\ service\_request, \\
& \qquad\qquad\qquad counter,\ stack,\ keys\_,\ keys\_g,\ keys\_c, \\
& \qquad\qquad\qquad keys\_ge,\ keys\_r,\ keys\_d,\ keys, \\
& \qquad\qquad\qquad inner\_state\rangle
\end{aligned}
$$

$$
\begin{aligned}
CheckBatch2(self) \triangleq\ & \land\ pc[self] = \text{“CheckBatch2”} \\
& \land\ \text{IF}\ GrpcViewingKeysBatch2 \neq \{\} \\
& \qquad \text{THEN}\ \land\ number\_of\_batches' = number\_of\_batches + 1 \\
& \qquad \text{ELSE}\ \land\ \text{TRUE} \\
& \qquad\qquad\ \land\ \text{UNCHANGED}\ number\_of\_batches \\
& \land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“CheckBatch3”}] \\
& \land\ \text{UNCHANGED}\ \langle scan\_tasks,\ response,\ scan\_task\_status, \\
& \qquad\qquad\qquad key\_to\_be\_served,\ service\_request, \\
& \qquad\qquad\qquad counter,\ stack,\ keys\_,\ keys\_g,\ keys\_c, \\
& \qquad\qquad\qquad keys\_ge,\ keys\_r,\ keys\_d,\ keys, \\
& \qquad\qquad\qquad inner\_state\rangle
\end{aligned}
$$

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

$$
\begin{aligned}
get\_config\_number\_of\_batches(self) \triangleq\ & CheckBatch1(self) \\
& \lor\ CheckBatch2(self) \\
& \lor\ CheckBatch3(self)
\end{aligned}
$$

$$
\begin{aligned}
AddConfigKeys(self) \triangleq\ & \land\ pc[self] = \text{“AddConfigKeys”} \\
& \land\ \exists\, key \in keys\_[self] : \\
& \qquad \land\ key\_to\_be\_served' = key \\
& \qquad \land\ scan\_task\_status' = \text{“adding”} \\
& \qquad \land\ pc' = [pc\ \text{EXCEPT}\ ![self] = Head(stack[self]).pc] \\
& \qquad \land\ keys\_' = [keys\_\ \text{EXCEPT}\ ![self] = Head(stack[self]).keys\_]
\end{aligned}
$$

$$\begin{array}{l} \qquad \land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\ \land \text{ UNCHANGED } \langle scan\_tasks,\ response,\ service\_request, \\ \qquad\qquad number\_of\_batches,\ counter,\ keys\_g, \\ \qquad\qquad keys\_c,\ keys\_ge,\ keys\_r,\ keys\_d,\ keys, \\ \qquad\qquad inner\_state\rangle \end{array}$$

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

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

$get\_info(self) \triangleq InfoServiceRequest(self)$

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

$get\_results(self) \triangleq ResultsServiceRequest(self)$

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

$clear\_results(self) \triangleq ClearServiceRequest(self)$

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

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

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

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

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

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

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

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

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

$$
\begin{aligned}
scan(self) \triangleq\ & RegisterServiceRequestFromScan(self) \\
& \lor\ ResultsServiceRequestFromScan(self) \\
& \lor\ SubscribeServiceRequestFromScan(self)
\end{aligned}
$$

$$
\begin{array}{l}
\mathit{Services} \triangleq \ \land\ \mathit{pc}[\text{“SERVICES”}] = \text{“Services”} \\
\qquad \land\ \text{IF } \mathit{service\_request} = \text{“info”} \\
\qquad\qquad \text{THEN } \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\text{“SERVICES”}] = \text{“Info”}] \\
\qquad\qquad \text{ELSE } \land\ \text{IF } \mathit{service\_request} = \text{“results”} \\
\qquad\qquad\qquad\qquad \text{THEN } \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\text{“SERVICES”}] = \text{“Results”}] \\
\qquad\qquad\qquad\qquad \text{ELSE } \land\ \text{IF } \mathit{service\_request} = \text{“clear”} \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\text{“SERVICES”}] = \text{“Clear”}] \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } \land\ \text{IF } \mathit{service\_request} = \text{“status”} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\text{“SERVICES”}] = \text{“Sta} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } \land\ \text{IF } \mathit{service\_request} = \text{“register”} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\text{“SERVIC} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } \land\ \text{IF } \mathit{service\_request} = \text{“delete”} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEP} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } \land\ \text{IF } \mathit{service\_reque} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } \land\ \mathit{pc}' \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } \land\ \mathit{pc}' \\
\qquad \land\ \text{UNCHANGED } \langle \mathit{scan\_tasks}, \mathit{response}, \mathit{scan\_task\_status}, \\
\qquad\qquad\qquad\qquad \mathit{key\_to\_be\_served}, \mathit{service\_request}, \\
\qquad\qquad\qquad\qquad \mathit{number\_of\_batches}, \mathit{counter}, \mathit{stack}, \mathit{keys\_}, \mathit{keys\_g}, \\
\qquad\qquad\qquad\qquad \mathit{keys\_c}, \mathit{keys\_ge}, \mathit{keys\_r}, \mathit{keys\_d}, \mathit{keys}, \mathit{inner\_state}\rangle
\end{array}
$$

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

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

$$
\begin{array}{l}
\mathit{Clear} \triangleq \ \land\ \mathit{pc}[\text{“SERVICES”}] = \text{“Clear”} \\
\qquad \land\ \text{IF } \mathit{key\_to\_be\_served} \in \mathit{scan\_tasks} \\
\qquad\qquad \text{THEN } \land\ \mathit{response}' = \mathit{clear\_response} \\
\qquad\qquad \text{ELSE } \land\ \mathit{response}' = \text{“Error: key not found.”} \\
\qquad \land\ \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\text{“SERVICES”}] = \text{“ClearRequestFlag”}] \\
\qquad \land\ \text{UNCHANGED } \langle \mathit{scan\_tasks}, \mathit{scan\_task\_status}, \mathit{key\_to\_be\_served}, \\
\qquad\qquad\qquad\qquad \mathit{service\_request}, \mathit{number\_of\_batches}, \mathit{counter}, \mathit{stack},
\end{array}
$$

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

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

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

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

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

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

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

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

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

$ScanTask \triangleq \land pc[\text{"SCAN TASK"}] = \text{"ScanTask"}$
$\land$ IF $Cardinality(scan\_tasks) > MaxScanTasks$
THEN $\land pc' = [pc \text{ EXCEPT } ![\text{"SCAN TASK"}] = \text{"BoundError"}]$
ELSE $\land$ IF $scan\_task\_status = \text{"adding"}$
THEN $\land pc' = [pc \text{ EXCEPT } ![\text{"SCAN TASK"}] = \text{"Adding"}]$
ELSE $\land$ IF $scan\_task\_status = \text{"deleting"}$

```
                                        THEN  ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “Deleting”]
                                        ELSE  ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “StoreScanTas
                ∧ 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⟩

BoundError ≜ ∧ pc[“SCAN TASK”] = “BoundError”
             ∧ response′ = “Error: max scan tasks reached.”
             ∧ scan_task_status′ = “waiting”
             ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “StoreScanTasks”]
             ∧ UNCHANGED ⟨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⟩

Adding ≜ ∧ pc[“SCAN TASK”] = “Adding”
         ∧ inner_state′ = (inner_state ∪ {key_to_be_served})
         ∧ scan_task_status′ = “waiting”
         ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “StoreScanTasks”]
         ∧ UNCHANGED ⟨scan_tasks, response, key_to_be_served,
                      service_request, number_of_batches, counter, stack,
                      keys_, keys_g, keys_c, keys_ge, keys_r, keys_d, keys⟩

Deleting ≜ ∧ pc[“SCAN TASK”] = “Deleting”
           ∧ scan_tasks′ = scan_tasks \ {key_to_be_served}
           ∧ scan_task_status′ = “waiting”
           ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “StoreScanTasks”]
           ∧ UNCHANGED ⟨response, 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⟩

StoreScanTasks ≜ ∧ pc[“SCAN TASK”] = “StoreScanTasks”
                 ∧ scan_tasks′ = inner_state
                 ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “ScanTaskLoop”]
                 ∧ UNCHANGED ⟨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⟩

ScanTaskLoop ≜ ∧ pc[“SCAN TASK”] = “ScanTaskLoop”
               ∧ pc′ = [pc EXCEPT ![“SCAN TASK”] = “ScanTask”]
               ∧ 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,
```

$inner\_state\rangle$

$scantask \triangleq GetScanTasks \lor ScanTask \lor BoundError \lor Adding \lor Deleting$
$\lor StoreScanTasks \lor ScanTaskLoop$

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

$FromZebradConfig \triangleq \land pc[\text{"MAIN"}] = \text{"FromZebradConfig"}$
$\land \land keys\_' = [keys\_ \text{ EXCEPT } ![\text{"MAIN"}] = ConfigViewingKeys]$
$\land stack' = [stack \text{ EXCEPT } ![\text{"MAIN"}] = \langle [procedure \mapsto \text{"add\_config\_keys"},$
$pc \mapsto \text{"ListeningGuard"},$
$keys\_ \mapsto keys\_[\text{"MAIN"}]]\rangle$
$\circ stack[\text{"MAIN"}]]$
$\land pc' = [pc \text{ EXCEPT } ![\text{"MAIN"}] = \text{"AddConfigKeys"}]$
$\land \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status,$
$key\_to\_be\_served, service\_request,$
$number\_of\_batches, counter, keys\_g, keys\_c,$
$keys\_ge, keys\_r, keys\_d, keys, inner\_state\rangle$

$ListeningGuard \triangleq \land pc[\text{"MAIN"}] = \text{"ListeningGuard"}$
$\land \text{IF } GrpcViewingKeys \neq \langle\rangle$
$\text{THEN } \land pc' = [pc \text{ EXCEPT } ![\text{"MAIN"}] = \text{"GetTotalIterationsToMake"}]$
$\text{ELSE } \land pc' = [pc \text{ EXCEPT } ![\text{"MAIN"}] = \text{"End"}]$
$\land \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status,$
$key\_to\_be\_served, service\_request,$
$number\_of\_batches, counter, stack, keys\_,$
$keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d,$
$keys, inner\_state\rangle$

$GetTotalIterationsToMake \triangleq \land pc[\text{"MAIN"}] = \text{"GetTotalIterationsToMake"}$
$\land stack' = [stack \text{ EXCEPT } ![\text{"MAIN"}] = \langle [procedure \mapsto \text{"get\_config\_number\_}$
$pc \mapsto \text{"ListeningMode"}]\rangle$
$\circ stack[\text{"MAIN"}]]$
$\land pc' = [pc \text{ EXCEPT } ![\text{"MAIN"}] = \text{"CheckBatch1"}]$
$\land \text{UNCHANGED } \langle scan\_tasks, response,$
$scan\_task\_status, key\_to\_be\_served,$
$service\_request, number\_of\_batches,$
$counter, keys\_, keys\_g, keys\_c,$

$keys\_ge$, $keys\_r$, $keys\_d$, $keys$,
$inner\_state\rangle$

$ListeningMode \triangleq$ $\land\ pc[\text{"MAIN"}] = \text{"ListeningMode"}$
$\land$ IF $counter \leq number\_of\_batches$
THEN $\land\ pc' = [pc$ EXCEPT $![\text{"MAIN"}] = \text{"GetInfoCall"}]$
ELSE $\land\ pc' = [pc$ EXCEPT $![\text{"MAIN"}] = \text{"End"}]$
$\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
$key\_to\_be\_served, service\_request,$
$number\_of\_batches, counter, stack, keys\_,$
$keys\_g, keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
$inner\_state\rangle$

$GetInfoCall \triangleq$ $\land\ pc[\text{"MAIN"}] = \text{"GetInfoCall"}$
$\land\ stack' = [stack$ EXCEPT $![\text{"MAIN"}] = \langle[procedure \mapsto \text{"get\_info"},$
$pc \mapsto \text{"RegisterKeysCall"}]\rangle$
$\circ\ stack[\text{"MAIN"}]]$
$\land\ pc' = [pc$ EXCEPT $![\text{"MAIN"}] = \text{"InfoServiceRequest"}]$
$\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
$key\_to\_be\_served, service\_request,$
$number\_of\_batches, counter, keys\_, keys\_g,$
$keys\_c, keys\_ge, keys\_r, keys\_d, keys,$
$inner\_state\rangle$

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

$GetStatusCall \triangleq$ $\land\ pc[\text{"MAIN"}] = \text{"GetStatusCall"}$
$\land\ \land\ keys\_ge' = [keys\_ge$ EXCEPT $![\text{"MAIN"}] = GrpcViewingKeys[counter]]$
$\land\ stack' = [stack$ EXCEPT $![\text{"MAIN"}] = \langle[procedure \mapsto \text{"get\_status"},$
$pc \mapsto \text{"GetResultsCall"},$
$keys\_ge \mapsto keys\_ge[\text{"MAIN"}]]\rangle$
$\circ\ stack[\text{"MAIN"}]]$
$\land\ pc' = [pc$ EXCEPT $![\text{"MAIN"}] = \text{"StatusServiceRequest"}]$
$\land$ UNCHANGED $\langle scan\_tasks, response, scan\_task\_status,$
$key\_to\_be\_served, service\_request,$
$number\_of\_batches, counter, keys\_, keys\_g,$

$keys\_c,\ keys\_r,\ keys\_d,\ keys,\ inner\_state\rangle$

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

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

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

$$
\begin{aligned}
ScanCall \triangleq\ & \land pc[\text{“MAIN”}] = \text{“ScanCall”} \\
& \land\ \land keys' = [keys \text{ EXCEPT } ![\text{“MAIN”}] = GrpcViewingKeys[counter]] \\
& \quad \land stack' = [stack \text{ EXCEPT } ![\text{“MAIN”}] = \langle [procedure \mapsto \text{“scan”}, \\
& \qquad\qquad pc \mapsto \text{“IncrementCounter”}, \\
& \qquad\qquad keys \mapsto keys[\text{“MAIN”}]] \rangle \\
& \qquad\qquad \circ stack[\text{“MAIN”}]] \\
& \land pc' = [pc \text{ EXCEPT } ![\text{“MAIN”}] = \text{“RegisterServiceRequestFromScan”}] \\
& \land \text{UNCHANGED } \langle scan\_tasks, response, scan\_task\_status,
\end{aligned}
$$

$key\_to\_be\_served$, $service\_request$,
$number\_of\_batches$, $counter$, $keys\_$, $keys\_g$, $keys\_c$,
$keys\_ge$, $keys\_r$, $keys\_d$, $inner\_state\rangle$

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

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

$Main \triangleq ConfigGuard \lor FromZebradConfig \lor ListeningGuard$
$\lor GetTotalIterationsToMake \lor ListeningMode \lor GetInfoCall$
$\lor RegisterKeysCall \lor GetStatusCall \lor GetResultsCall$
$\lor ClearResultsCall \lor DeleteKeysCall \lor ScanCall$
$\lor IncrementCounter \lor End$

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

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

$Spec \triangleq \land Init \land \Box[Next]_{vars}$
$\land \text{WF}_{vars}(scantask)$

$Termination \triangleq \Diamond(\forall self \in ProcSet : pc[self] = \text{“Done”})$

END TRANSLATION

$\backslash * Modification\ History$

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