MODULE *client_integration*

External client support for Zebra specification

The specs simulates a call to the *z_getnewaccount* rpc method as a starting point which calls the *create_account* procedure in the *zcash_client_backend* side. The rpc method then sends the key to the scan task to start scanning and to the memory wallet who adds the key to the accounts set. The zebra scanner eventually sends a block to the memory wallet and the memory wallet adds the block to the blocks set.

The memory wallet is a simple algorithm that listens for requests and sends adding requests to the scan task. The scan task listens for requests from the services process and adds tasks to the scan task set. The scan task also adds account to the memory wallet and either sends "scanned" blocks to the memory wallet or does nothing more.

The main process is the entry point of the model and calls the *z_getnewaccount* rpc method.

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

$StatusWaiting \triangleq$ "waiting"
$StatusAdding \triangleq$ "adding"
$CreateAccountServiceRequest \triangleq$ "create_account"

--algorithm *client_integration*

variables

A string that will be used as a response to any of the *gRPC* method calls, initially empty.

$response =$ "" **;**

The current service request flag, initially listening for requests.

$service\_request = StatusWaiting$ **;**

The current status of the scan task, initially listening for requests.

$scan\_task\_status = StatusWaiting$ **;**

The set of scan tasks that are currently being processed, initially empty.

$scan\_tasks = \{\}$ **;**

The key that will be served to the client after a create account request.

$key\_to\_be\_served =$ "" **;**

The block that will be served to the client after a scan task finds a relevant block, initially empty.

$block\_to\_be\_served = [height \mapsto 0,\ hash \mapsto$ "000000"$]$ **;**

The set of accounts that in the memory wallet, initially empty.

$accounts = \{\}$ **;**

The set of blocks in the memory wallet, initially empty.

$blocks = \{\}$ **;**

Keep track of the last inserted accouint id.

$last\_account\_id = 0$ **;**

define

Ensure that whenever a block is available, it eventually gets inserted into the memory wallet.

$LIVENESS\_BLOCK\_INSERTION \triangleq$
$\quad \land\ block\_to\_be\_served.height > 0$
$\quad \Rightarrow \Diamond(\forall b \in blocks : b = block\_to\_be\_served)$

Ensure that an account is not added twice.

```
    $SAFETY\_ACCOUNT\_ADDITION \triangleq$
        $\land \forall a \in accounts :$
            $\land a.account\_id \geq 0$
            $\land \forall b \in accounts : b.account\_id \neq a.account\_id$
     Ensure that the account id is incremented properly.
    $SAFETY\_ACCOUNT\_ID\_INCREMENT \triangleq$
        $\land \forall a, b \in accounts : a.account\_id < b.account\_id$
     Ensure that a block is not inserted multiple times.
    $SAFETY\_BLOCK\_INSERTION \triangleq$
        $\land \forall b \in blocks :$
            $\land b.height > 0$
            $\land \forall c \in blocks : c.height \neq b.height$
     Ensure that the service request always return to listening after adding.
    $SERVICE\_REQUEST\_TRANSITION \triangleq$
        $\land service\_request = StatusAdding$
          $\Rightarrow \Diamond(service\_request = StatusWaiting)$
     Ensure that all accounts have a non empty ufvk and that blocks have non zero hash.
    $INDUCTIVE\_INVARIANT \triangleq$
        $\land (\forall acc \in accounts : acc.ufvk \neq \text{""})$
        $\land (\forall blk \in blocks : blk.hash \neq \text{"000000"})$
        $\land service\_request \in \{StatusWaiting, StatusAdding, CreateAccountServiceRequest\}$
end define ;
```

UTILITY PROCEDURES:

– Procedure to initiate the *RPC* call for account creation. This sets the *service_request* flag to signal that an account creation request has been received and is being processed.

```
procedure $z\_getnewaccount()$
begin
    $GetNewAccountRPC:$
        $service\_request := CreateAccountServiceRequest$ ;
end procedure ;
```

The *create_account in the zcash_client_backend side.*

```
procedure $create\_account\_zcash\_client\_backend()$
begin
    $CreateAccountZcashClientBackend:$
        $response :=$ "zxviews..." ;
        return ;
end procedure ;
```

The put_block in the zcash_client_backend side.

```
procedure $put\_block\_zcash\_client\_backend()$
begin
    $PutBlockZcashClientBackend:$
        $blocks := blocks \cup \{block\_to\_be\_served\}$ ;
```

```
end procedure ;

  SERVICES PROCESS :

  Listen for requests and send adding requests to scan task.
process $services$ = “SERVICES”
begin
    Services:
          We only have one service request in this algorithm.
        if $service\_request = CreateAccountServiceRequest$ then
            CallZcashClientBackend:
                call $create\_account\_zcash\_client\_backend()$ ;
            SendKey:
                $key\_to\_be\_served := response$ ;
            CreateAccount:
                $scan\_task\_status := StatusAdding$ ;
        end if ;
    ServicesLoop:
        goto Services ;
end process ;


  SCAN TASK PROCESS :

  Listen for requests from the services process and :
− Add tasks to the scan task set.
− Add account to the memory wallet.
− Either send “scanned” blocks to the memory wallet or do nothing more.
process $scantask$ = “SCAN TASK”
variables $inner\_state = \{\}$, $inner\_accounts = \{\}$, $inner\_blocks = \{\}$, $inner\_last\_account\_id = 0$ ;
begin
    GetGlobals:
        $inner\_state := scan\_tasks$ ;
        $inner\_accounts := accounts$ ;
        $inner\_last\_account\_id := last\_account\_id$ ;
    ScanTask:
        if $scan\_task\_status = StatusAdding$ then
            AddingAccount:
                $accounts := inner\_accounts \cup \{[account\_id \mapsto last\_account\_id + 1,\ ufvk \mapsto key\_to\_be\_serve$
                $scan\_tasks := inner\_state \cup \{key\_to\_be\_served\}$ ;
                $scan\_task\_status := StatusWaiting$ ;
                $last\_account\_id := inner\_last\_account\_id + 1$ ;
        end if ;
    SendBlock:
        either
            $block\_to\_be\_served := [height \mapsto 1,\ hash \mapsto$ “111111”$]$ ;
            call put_block_zcash_client_backend() ;
        or
            skip ;
        end either ;
    ScanTaskLoop:
        goto ScanTask ;
end process ;

 MAIN PROCESS :

process Main = "MAIN"
begin
    CreteAccountCall:
         The RPC is the entry point of the model.
        call z_getnewaccount() ;
    End:
        skip ;
end process ;

end algorithm ;
 BEGIN TRANSLATION(chksum(pcal) = "33d228a9" ∧ chksum(tla) = "4d91d949")
VARIABLES response, service_request, scan_task_status, scan_tasks,
          key_to_be_served, block_to_be_served, accounts, blocks,
          last_account_id, pc, stack

 define statement
LIVENESS_BLOCK_INSERTION ≜
    $\land$ $block\_to\_be\_served.height > 0$
    $\Rightarrow \Diamond(\forall b \in blocks : b = block\_to\_be\_served)$

SAFETY_ACCOUNT_ADDITION ≜
    $\land \forall a \in accounts$ :
          $\land$ $a.account\_id \geq 0$
          $\land \forall b \in accounts : b.account\_id \neq a.account\_id$

SAFETY_ACCOUNT_ID_INCREMENT ≜
    $\land \forall a, b \in accounts : a.account\_id < b.account\_id$

SAFETY_BLOCK_INSERTION ≜
    $\land \forall b \in blocks$ :
          $\land$ $b.height > 0$
          $\land \forall c \in blocks : c.height \neq b.height$

SERVICE_REQUEST_TRANSITION ≜
    $\land$ $service\_request = StatusAdding$
        $\Rightarrow \Diamond(service\_request = StatusWaiting)$

INDUCTIVE_INVARIANT ≜
```

$$\begin{array}{l}
\land\ (\forall\, acc \in accounts : acc.ufvk \neq \text{“”})\\
\land\ (\forall\, blk \in blocks : blk.hash \neq \text{“000000”})\\
\land\ service\_request \in \{StatusWaiting,\ StatusAdding,\ CreateAccountServiceRequest\}
\end{array}$$

VARIABLES $inner\_state,\ inner\_accounts,\ inner\_blocks,\ inner\_last\_account\_id$

$$\begin{array}{ll}
vars \triangleq & \langle response,\ service\_request,\ scan\_task\_status,\ scan\_tasks,\\
 & key\_to\_be\_served,\ block\_to\_be\_served,\ accounts,\ blocks,\\
 & last\_account\_id,\ pc,\ stack,\ inner\_state,\ inner\_accounts,\\
 & inner\_blocks,\ inner\_last\_account\_id\rangle
\end{array}$$

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

$$\begin{array}{ll}
Init \triangleq & \textit{Global variables}\\
 & \land\ response = \text{“”}\\
 & \land\ service\_request = StatusWaiting\\
 & \land\ scan\_task\_status = StatusWaiting\\
 & \land\ scan\_tasks = \{\}\\
 & \land\ key\_to\_be\_served = \text{“”}\\
 & \land\ block\_to\_be\_served = [height \mapsto 0,\ hash \mapsto \text{“000000”}]\\
 & \land\ accounts = \{\}\\
 & \land\ blocks = \{\}\\
 & \land\ last\_account\_id = 0\\
 & \textit{Process scantask}\\
 & \land\ inner\_state = \{\}\\
 & \land\ inner\_accounts = \{\}\\
 & \land\ inner\_blocks = \{\}\\
 & \land\ inner\_last\_account\_id = 0\\
 & \land\ stack = [self \in ProcSet \mapsto \langle\rangle]\\
 & \land\ pc = [self \in ProcSet \mapsto \text{CASE } self = \text{“SERVICES”} \rightarrow \text{“Services”}\\
 & \qquad\qquad\qquad\qquad\qquad \Box\ \ self = \text{“SCAN TASK”} \rightarrow \text{“GetGlobals”}\\
 & \qquad\qquad\qquad\qquad\qquad \Box\ \ self = \text{“MAIN”} \rightarrow \text{“CreteAccountCall”}]
\end{array}$$

$$\begin{array}{ll}
GetNewAccountRPC(self) \triangleq & \land\ pc[self] = \text{“GetNewAccountRPC”}\\
 & \land\ service\_request' = CreateAccountServiceRequest\\
 & \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Error”}]\\
 & \land\ \text{UNCHANGED } \langle response,\ scan\_task\_status,\\
 & \qquad scan\_tasks,\ key\_to\_be\_served,\\
 & \qquad block\_to\_be\_served,\ accounts,\ blocks,\\
 & \qquad last\_account\_id,\ stack,\ inner\_state,\\
 & \qquad inner\_accounts,\ inner\_blocks,\\
 & \qquad inner\_last\_account\_id\rangle
\end{array}$$

$$z\_getnewaccount(self) \triangleq GetNewAccountRPC(self)$$

$$\begin{array}{ll}
CreateAccountZcashClientBackend(self) \triangleq & \land\ pc[self] = \text{“CreateAccountZcashClientBackend”}\\
 & \land\ response' = \text{“zxviews...”}
\end{array}$$

$$\begin{array}{l}
\land\ pc' = [pc\ \textsc{except}\ ![self] = Head(stack[self]).pc]\\
\land\ stack' = [stack\ \textsc{except}\ ![self] = Tail(stack[self])]\\
\land\ \textsc{unchanged}\ \langle service\_request,\\
\qquad scan\_task\_status,\\
\qquad scan\_tasks,\\
\qquad key\_to\_be\_served,\\
\qquad block\_to\_be\_served,\\
\qquad accounts,\ blocks,\\
\qquad last\_account\_id,\\
\qquad inner\_state,\\
\qquad inner\_accounts,\\
\qquad inner\_blocks,\\
\qquad inner\_last\_account\_id\rangle
\end{array}$$

$$create\_account\_zcash\_client\_backend(self) \triangleq CreateAccountZcashClientBackend(self)$$

$$\begin{array}{ll}
PutBlockZcashClientBackend(self) \triangleq & \land\ pc[self] = \text{“PutBlockZcashClientBackend”}\\
& \land\ blocks' = (blocks \cup \{block\_to\_be\_served\})\\
& \land\ pc' = [pc\ \textsc{except}\ ![self] = \text{“Error”}]\\
& \land\ \textsc{unchanged}\ \langle response,\ service\_request,\\
& \qquad scan\_task\_status,\\
& \qquad scan\_tasks,\\
& \qquad key\_to\_be\_served,\\
& \qquad block\_to\_be\_served,\\
& \qquad accounts,\ last\_account\_id,\\
& \qquad stack,\ inner\_state,\\
& \qquad inner\_accounts,\\
& \qquad inner\_blocks,\\
& \qquad inner\_last\_account\_id\rangle
\end{array}$$

$$put\_block\_zcash\_client\_backend(self) \triangleq PutBlockZcashClientBackend(self)$$

$$\begin{array}{ll}
Services \triangleq & \land\ pc[\text{“SERVICES”}] = \text{“Services”}\\
& \land\ \textsc{if}\ service\_request = CreateAccountServiceRequest\\
& \qquad \textsc{then}\ \land\ pc' = [pc\ \textsc{except}\ ![\text{“SERVICES”}] = \text{“CallZcashClientBackend”}]\\
& \qquad \textsc{else}\ \ \land\ pc' = [pc\ \textsc{except}\ ![\text{“SERVICES”}] = \text{“ServicesLoop”}]\\
& \land\ \textsc{unchanged}\ \langle response,\ service\_request,\ scan\_task\_status,\\
& \qquad scan\_tasks,\ key\_to\_be\_served,\ block\_to\_be\_served,\\
& \qquad accounts,\ blocks,\ last\_account\_id,\ stack,\\
& \qquad inner\_state,\ inner\_accounts,\ inner\_blocks,\\
& \qquad inner\_last\_account\_id\rangle
\end{array}$$

$$\begin{array}{ll}
CallZcashClientBackend \triangleq & \land\ pc[\text{“SERVICES”}] = \text{“CallZcashClientBackend”}\\
& \land\ stack' = [stack\ \textsc{except}\ ![\text{“SERVICES”}] = \langle[procedure \mapsto \text{“create\_account\_z}\\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad pc \mapsto \text{“SendKey”}]\rangle\\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\ stack[\text{“SERVICES”}]]
\end{array}$$

$$\begin{array}{l}
\land\ pc' = [pc\ \text{EXCEPT}\ ![\textsf{“SERVICES”}] = \textsf{“CreateAccountZcashClientBackend”}]\\
\land\ \text{UNCHANGED}\ \langle \mathit{response}, \mathit{service\_request},\\
\qquad \mathit{scan\_task\_status}, \mathit{scan\_tasks},\\
\qquad \mathit{key\_to\_be\_served}, \mathit{block\_to\_be\_served},\\
\qquad \mathit{accounts}, \mathit{blocks}, \mathit{last\_account\_id},\\
\qquad \mathit{inner\_state}, \mathit{inner\_accounts},\\
\qquad \mathit{inner\_blocks}, \mathit{inner\_last\_account\_id} \rangle
\end{array}$$

$$\mathit{SendKey} \triangleq \begin{array}[t]{l}
\land\ pc[\textsf{“SERVICES”}] = \textsf{“SendKey”}\\
\land\ \mathit{key\_to\_be\_served}' = \mathit{response}\\
\land\ pc' = [pc\ \text{EXCEPT}\ ![\textsf{“SERVICES”}] = \textsf{“CreateAccount”}]\\
\land\ \text{UNCHANGED}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_task\_status},\\
\qquad \mathit{scan\_tasks}, \mathit{block\_to\_be\_served}, \mathit{accounts}, \mathit{blocks},\\
\qquad \mathit{last\_account\_id}, \mathit{stack}, \mathit{inner\_state}, \mathit{inner\_accounts},\\
\qquad \mathit{inner\_blocks}, \mathit{inner\_last\_account\_id} \rangle
\end{array}$$

$$\mathit{CreateAccount} \triangleq \begin{array}[t]{l}
\land\ pc[\textsf{“SERVICES”}] = \textsf{“CreateAccount”}\\
\land\ \mathit{scan\_task\_status}' = \mathit{StatusAdding}\\
\land\ pc' = [pc\ \text{EXCEPT}\ ![\textsf{“SERVICES”}] = \textsf{“ServicesLoop”}]\\
\land\ \text{UNCHANGED}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_tasks},\\
\qquad \mathit{key\_to\_be\_served}, \mathit{block\_to\_be\_served},\\
\qquad \mathit{accounts}, \mathit{blocks}, \mathit{last\_account\_id}, \mathit{stack},\\
\qquad \mathit{inner\_state}, \mathit{inner\_accounts}, \mathit{inner\_blocks},\\
\qquad \mathit{inner\_last\_account\_id} \rangle
\end{array}$$

$$\mathit{ServicesLoop} \triangleq \begin{array}[t]{l}
\land\ pc[\textsf{“SERVICES”}] = \textsf{“ServicesLoop”}\\
\land\ pc' = [pc\ \text{EXCEPT}\ ![\textsf{“SERVICES”}] = \textsf{“Services”}]\\
\land\ \text{UNCHANGED}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_task\_status},\\
\qquad \mathit{scan\_tasks}, \mathit{key\_to\_be\_served},\\
\qquad \mathit{block\_to\_be\_served}, \mathit{accounts}, \mathit{blocks},\\
\qquad \mathit{last\_account\_id}, \mathit{stack}, \mathit{inner\_state},\\
\qquad \mathit{inner\_accounts}, \mathit{inner\_blocks},\\
\qquad \mathit{inner\_last\_account\_id} \rangle
\end{array}$$

$$\mathit{services} \triangleq \begin{array}[t]{l}
\mathit{Services} \lor \mathit{CallZcashClientBackend} \lor \mathit{SendKey} \lor \mathit{CreateAccount}\\
\quad \lor\ \mathit{ServicesLoop}
\end{array}$$

$$\mathit{GetGlobals} \triangleq \begin{array}[t]{l}
\land\ pc[\textsf{“SCAN TASK”}] = \textsf{“GetGlobals”}\\
\land\ \mathit{inner\_state}' = \mathit{scan\_tasks}\\
\land\ \mathit{inner\_accounts}' = \mathit{accounts}\\
\land\ \mathit{inner\_last\_account\_id}' = \mathit{last\_account\_id}\\
\land\ pc' = [pc\ \text{EXCEPT}\ ![\textsf{“SCAN TASK”}] = \textsf{“ScanTask”}]\\
\land\ \text{UNCHANGED}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_task\_status},\\
\qquad \mathit{scan\_tasks}, \mathit{key\_to\_be\_served}, \mathit{block\_to\_be\_served},\\
\qquad \mathit{accounts}, \mathit{blocks}, \mathit{last\_account\_id}, \mathit{stack},\\
\qquad \mathit{inner\_blocks} \rangle
\end{array}$$

$$
\begin{array}{l}
\mathit{ScanTask} \triangleq \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“ScanTask”} \\
\qquad\qquad \land\ \textsc{if}\ \mathit{scan\_task\_status} = \mathit{StatusAdding} \\
\qquad\qquad\qquad \textsc{then}\ \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \text{“AddingAccount”}] \\
\qquad\qquad\qquad \textsc{else}\ \ \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \text{“SendBlock”}] \\
\qquad\qquad \land\ \textsc{unchanged}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_task\_status}, \\
\qquad\qquad\qquad\qquad \mathit{scan\_tasks}, \mathit{key\_to\_be\_served}, \mathit{block\_to\_be\_served}, \\
\qquad\qquad\qquad\qquad \mathit{accounts}, \mathit{blocks}, \mathit{last\_account\_id}, \mathit{stack}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_state}, \mathit{inner\_accounts}, \mathit{inner\_blocks}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_last\_account\_id} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{AddingAccount} \triangleq \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“AddingAccount”} \\
\qquad\qquad \land\ \mathit{accounts}' = (\mathit{inner\_accounts} \cup \{[\mathit{account\_id} \mapsto \mathit{last\_account\_id} + 1,\ \mathit{ufvk} \mapsto \mathit{key\_to\_b} \\
\qquad\qquad \land\ \mathit{scan\_tasks}' = (\mathit{inner\_state} \cup \{\mathit{key\_to\_be\_served}\}) \\
\qquad\qquad \land\ \mathit{scan\_task\_status}' = \mathit{StatusWaiting} \\
\qquad\qquad \land\ \mathit{last\_account\_id}' = \mathit{inner\_last\_account\_id} + 1 \\
\qquad\qquad \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \text{“SendBlock”}] \\
\qquad\qquad \land\ \textsc{unchanged}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{key\_to\_be\_served}, \\
\qquad\qquad\qquad\qquad \mathit{block\_to\_be\_served}, \mathit{blocks}, \mathit{stack}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_state}, \mathit{inner\_accounts}, \mathit{inner\_blocks}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_last\_account\_id} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{SendBlock} \triangleq \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“SendBlock”} \\
\qquad\quad \land\ \lor\ \land\ \mathit{block\_to\_be\_served}' = [\mathit{height} \mapsto 1,\ \mathit{hash} \mapsto \text{“111111”}] \\
\qquad\qquad\quad \land\ \mathit{stack}' = [\mathit{stack}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \langle [\mathit{procedure} \mapsto \text{“put\_block\_zcash\_client\_} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{pc} \qquad\ \ \mapsto \text{“ScanTaskLoop”}] \rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \circ\ \mathit{stack}[\text{“SCAN TASK”}]] \\
\qquad\qquad\quad \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \text{“PutBlockZcashClientBackend”}] \\
\qquad\qquad \lor\ \land\ \textsc{true} \\
\qquad\qquad\quad \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \text{“ScanTaskLoop”}] \\
\qquad\qquad\quad \land\ \textsc{unchanged}\ \langle \mathit{block\_to\_be\_served}, \mathit{stack} \rangle \\
\qquad\quad \land\ \textsc{unchanged}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_task\_status}, \\
\qquad\qquad\qquad\qquad \mathit{scan\_tasks}, \mathit{key\_to\_be\_served}, \mathit{accounts}, \mathit{blocks}, \\
\qquad\qquad\qquad\qquad \mathit{last\_account\_id}, \mathit{inner\_state}, \mathit{inner\_accounts}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_blocks}, \mathit{inner\_last\_account\_id} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{ScanTaskLoop} \triangleq \land\ \mathit{pc}[\text{“SCAN TASK”}] = \text{“ScanTaskLoop”} \\
\qquad\qquad \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{“SCAN TASK”}] = \text{“ScanTask”}] \\
\qquad\qquad \land\ \textsc{unchanged}\ \langle \mathit{response}, \mathit{service\_request}, \mathit{scan\_task\_status}, \\
\qquad\qquad\qquad\qquad \mathit{scan\_tasks}, \mathit{key\_to\_be\_served}, \\
\qquad\qquad\qquad\qquad \mathit{block\_to\_be\_served}, \mathit{accounts}, \mathit{blocks}, \\
\qquad\qquad\qquad\qquad \mathit{last\_account\_id}, \mathit{stack}, \mathit{inner\_state}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_accounts}, \mathit{inner\_blocks}, \\
\qquad\qquad\qquad\qquad \mathit{inner\_last\_account\_id} \rangle
\end{array}
$$

$$
\begin{array}{l}
\mathit{scantask} \triangleq \mathit{GetGlobals} \lor \mathit{ScanTask} \lor \mathit{AddingAccount} \lor \mathit{SendBlock} \\
\qquad\qquad \lor\ \mathit{ScanTaskLoop}
\end{array}
$$

$$\mathit{CreteAccountCall} \triangleq \begin{array}[t]{l} \land\, pc[\text{“MAIN”}] = \text{“CreteAccountCall”} \\ \land\, stack' = [stack \ \text{EXCEPT} \ ![\text{“MAIN”}] = \langle [\begin{array}[t]{lcl} procedure & \mapsto & \text{“z\_getnewaccount”}, \\ pc & \mapsto & \text{“End”}] \rangle \end{array} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\, stack[\text{“MAIN”}]] \\ \land\, pc' = [pc \ \text{EXCEPT} \ ![\text{“MAIN”}] = \text{“GetNewAccountRPC”}] \\ \land\, \text{UNCHANGED} \ \langle \begin{array}[t]{l} response, service\_request, \\ scan\_task\_status, scan\_tasks, \\ key\_to\_be\_served, block\_to\_be\_served, \\ accounts, blocks, last\_account\_id, \\ inner\_state, inner\_accounts, inner\_blocks, \\ inner\_last\_account\_id \rangle \end{array} \end{array}$$

$$\mathit{End} \triangleq \begin{array}[t]{l} \land\, pc[\text{“MAIN”}] = \text{“End”} \\ \land\, \text{TRUE} \\ \land\, pc' = [pc \ \text{EXCEPT} \ ![\text{“MAIN”}] = \text{“Done”}] \\ \land\, \text{UNCHANGED} \ \langle \begin{array}[t]{l} response, service\_request, scan\_task\_status, scan\_tasks, \\ key\_to\_be\_served, block\_to\_be\_served, accounts, blocks, \\ last\_account\_id, stack, inner\_state, inner\_accounts, \\ inner\_blocks, inner\_last\_account\_id \rangle \end{array} \end{array}$$

$$\mathit{Main} \triangleq \mathit{CreteAccountCall} \lor \mathit{End}$$

Allow infinite stuttering to prevent deadlock on termination.

$$\mathit{Terminating} \triangleq \begin{array}[t]{l} \land\, \forall\, self \in ProcSet : pc[self] = \text{“Done”} \\ \land\, \text{UNCHANGED} \ vars \end{array}$$

$$\mathit{Next} \triangleq \begin{array}[t]{l} services \lor scantask \lor Main \\ \quad \lor\, (\exists\, self \in ProcSet : \begin{array}[t]{l} \lor\, z\_getnewaccount(self) \\ \lor\, create\_account\_zcash\_client\_backend(self) \\ \lor\, put\_block\_zcash\_client\_backend(self)) \end{array} \\ \quad \lor\, Terminating \end{array}$$

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

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

END TRANSLATION