```
┌──────────────────── MODULE CompositionalSpec ────────────────────┐
EXTENDS Naturals, Sequences, TLC

CONSTANT Apps, Permissions

EnvironmentId ≜ "EnvironmentId"
ApsId ≜ "ApsId"
MaxAllowedUpdates ≜ 2

 **CONSTANTS **
Null ≜ "NULL"
PERMISSION_DENIED ≜ "PERMISSION_DENIED"
PERMISSION_ALLOWED ≜ "PERMISSION_ALLOWED"
PERMISSION_ONLY_ONCE ≜ "PERMISSION_ONLY_ONCE"
PERMISSION_WHILE_USING_APP ≜ "PERMISSION_WHILE_USING_APP"
ACTION_RECORD_AUDIO ≜ "ACTION_RECORD_AUDIO"
DATUM_SYSTEM_STATE ≜ "DATUM_SYSTEM_STATE"
PERMISSION_TYPE_URI ≜ "PERMISSION_TYPE_URI"
PERMISSION_TYPE_CUSTOM ≜ "PERMISSION_TYPE_CUSTOM"
PERMISSION_TYPE_NORMAL ≜ "PERMISSION_TYPE_NORMAL"
PERMISSION_TYPE_RUNTIME ≜ "PERMISSION_TYPE_RUNTIME"
PERMISSION_TYPE_SPECIAL  ≜ "PERMISSION_TYPE_SPECIAL"
PERMISSION_TYPE_SIGNATURE ≜ "PERMISSION_TYPE_SIGNATURE"
PROTECTION_LEVEL_NORMAL ≜ "PROTECTION_LEVEL_NORMAL"
PROTECTION_LEVEL_APP_OP  ≜ "PROTECTION_LEVEL_APP_OP"
PROTECTION_LEVEL_SIGNATURE  ≜ "PROTECTION_LEVEL_SIGNATURE"
PROTECTION_LEVEL_DANGEROUS ≜ "PROTECTION_LEVEL_DANGEROUS"
 **END OF CONSTANTS **

 **ENUMS **
Boolean ≜ {TRUE, FALSE, Null}
DatumType ≜ {DATUM_SYSTEM_STATE}
ActionType ≜ {ACTION_RECORD_AUDIO}
ConsentType ≜ {PERMISSION_DENIED, PERMISSION_ALLOWED,
               PERMISSION_ONLY_ONCE, PERMISSION_WHILE_USING_APP}
ProtectionLevel ≜ {PROTECTION_LEVEL_NORMAL, PROTECTION_LEVEL_SIGNATURE,
                   PROTECTION_LEVEL_DANGEROUS, PROTECTION_LEVEL_APP_OP}
PermissionType ≜ {PERMISSION_TYPE_NORMAL, PERMISSION_TYPE_SIGNATURE,
                  PERMISSION_TYPE_RUNTIME, PERMISSION_TYPE_SPECIAL,
                  PERMISSION_TYPE_URI, PERMISSION_TYPE_CUSTOM}
 **END OF ENUMS **

** --algorithm Universe
{
   variables env_vars =
   [actions ↦ {},
 applications ↦
  [a ∈ Apps ↦ [installed ↦ FALSE, version ↦ 0, terminated ↦ FALSE]],
   data ↦ {}, permissions ↦ [p ∈ Permissions ↦ [a ∈ Apps ↦ Null]],
   permission_groups ↦ {}] ;

app_vars = [a ∈ Apps ↦
               [manifest ↦ [p ∈ Permissions ↦ Null],
                signature ↦ {}, private_keys ↦ {}, public_key ↦ {},
                certificate ↦ {}, package ↦ {}, services ↦ {},
                receivers ↦ {}, activities ↦ {}, content_providers ↦ {}]] ;

aps_vars = [permission_history ↦ {}] ;

procedure installApp( app ) { INSTALL_APP: env_vars.applications[app].installed := TRUE ; return ;
procedure uninstallApp( app ) { UNINSTALL_APP: env_vars.applications[app].installed := FALSE ; re
procedure updateApp( app ) { UPDATE_APP: env_vars.applications[app].version := env_vars.applica
procedure terminate( app ) { TERMINATED: env_vars.applications[self].terminated := TRUE ; return
procedure declarePermission( app, perm ) { DECLARE_PERMISSION: app_vars[app].manifest[perm]
 procedure revokePermission(app){REVOKE_PERMISSION : return; }
 procedure grantUriPermission(app){GRANT_URI_PERMISSION : return; }
 procedure revokeUriPermission(app){REVOKE_URI_PERMISSION : return; }
 procedure checkUriPermission(app){CHECK_URI_PERMISSION : return; }
procedure checkSelfPermission( app ) { CHECK_SELF_PERMISSION: return ;  }
procedure shouldShowRequestPermissionRationale( app ) { SHOULD_SHOW_REQUEST_PERMISSIO
procedure requestPermission( app ) { REQUEST_PERMISSION: return ;  }
 procedure requestMultiplePermissions(app){REQUEST_MULTIPLE_PERMISSIONS : return; }
 procedure removeUnusedPermissions(app){REMOVE_UNUSED_PERMISSIONS : return; }

fair process ( EnvNext = EnvironmentId )
    variables i ;
{
    EnvBegin:- while ( FALSE )
    {
        skip ;
     } ;
 }

fair process ( AppNext ∈ Apps )
    variables i ;
{
    AppBegin:- while ( env_vars.applications[self].terminated ≠ TRUE )
    {
     either
     {
      call terminate(self) ;
      }
     or
```

```
        {
         if ( $env\_vars.applications[self].installed = \text{TRUE}$ )
         {
          either
          {
            call $uninstallApp(self)$ ;
           }
          or
          {
           either
           {
            if ( $env\_vars.applications[self].version < MaxAllowedUpdates$ )
            {
             call $updateApp(self)$ ;
             }
            }
           or
           {
            skip ;
            }
           }
          }
         else
         {
          $DECLARING\_PERMISSION$: with ( $p \in Permissions$ ) { call $declarePermission(self, p)$ ;  } ;
          $INSTALLING\_APP$: call $installApp(self)$ ;
          }
         }
        } ;
    }

   fair process ( $ApsNext = ApsId$ )
       variables $i$ ;
   {
       $ApsBegin$:- while ( FALSE )
       {
           skip ;
        } ;
    }
}
```
**

BEGIN TRANSLATION ($chksum(pcal) =$ "$7be9e7d1$" $\land$ $chksum(tla) =$ "$d9bf6943$")

Process variable i of process $EnvNext$ at line 76 col 19 changed to $i\_$

Process variable i of process $AppNext$ at line 85 col 19 changed to i_A

Parameter app of procedure $installApp$ at line 60 col 26 changed to $app\_$

Parameter *app* of procedure *uninstallApp* at line 61 col 28 changed to app_u
Parameter *app* of procedure *updateApp* at line 62 col 25 changed to app_up
Parameter *app* of procedure terminate at line 63 col 25 changed to app_t
Parameter *app* of procedure *declarePermission* at line 64 col 33 changed to app_d
Parameter *app* of procedure *checkSelfPermission* at line 69 col 35 changed to app_c
Parameter *app* of procedure *shouldShowRequestPermissionRationale* at line 70 col 52 changed to app_s

CONSTANT $defaultInitValue$

VARIABLES $env\_vars,\ app\_vars,\ aps\_vars,\ pc,\ stack,\ app\_,\ app\_u,\ app\_up,\ app\_t,$
$app\_d,\ perm,\ app\_c,\ app\_s,\ app,\ i\_,\ i\_A,\ i$

$$vars \triangleq \langle env\_vars, app\_vars, aps\_vars, pc, stack, app\_, app\_u, app\_up, app\_t, app\_d, perm, app\_c, app\_s, app, i\_, i\_A, i\rangle$$

$$ProcSet \triangleq \{EnvironmentId\} \cup (Apps) \cup \{ApsId\}$$

$Init \triangleq$ Global variables

$\land\ env\_vars = [actions \mapsto \{\},$
$\quad applications \mapsto$
$\quad\quad [a \in Apps \mapsto [installed \mapsto \text{FALSE},\ version \mapsto 0,\ terminated \mapsto \text{FALSE}]],$
$\quad data \mapsto \{\},\ permissions \mapsto [p \in Permissions \mapsto [a \in Apps \mapsto Null]],$
$\quad permission\_groups \mapsto \{\}]$

$\land\ app\_vars = [a \in Apps \mapsto$
$\quad [manifest \mapsto [p \in Permissions \mapsto Null],$
$\quad signature \mapsto \{\},\ private\_keys \mapsto \{\},\ public\_key \mapsto \{\},$
$\quad certificate \mapsto \{\},\ package \mapsto \{\},\ services \mapsto \{\},$
$\quad receivers \mapsto \{\},\ activities \mapsto \{\},\ content\_providers \mapsto \{\}]]$

$\land\ aps\_vars = [permission\_history \mapsto \{\}]$

Procedure *installApp*

$\land\ app\_ = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *uninstallApp*

$\land\ app\_u = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *updateApp*

$\land\ app\_up = [self \in ProcSet \mapsto defaultInitValue]$

Procedure terminate

$\land\ app\_t = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *declarePermission*

$\land\ app\_d = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ perm = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *checkSelfPermission*

$\land\ app\_c = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *shouldShowRequestPermissionRationale*

$\land\ app\_s = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *requestPermission*

$\land\ app = [self \in ProcSet \mapsto defaultInitValue]$

Process *EnvNext*

$\land\ i\_ = defaultInitValue$

Process $AppNext$

$\land\ i\_A = [self \in Apps \mapsto defaultInitValue]$

Process $ApsNext$

$\land\ i = defaultInitValue$

$\land\ stack = [self \in ProcSet \mapsto \langle\rangle]$

$$\begin{array}{l}
\land\ pc = [self \in ProcSet \mapsto \text{CASE } self = EnvironmentId \rightarrow \text{“EnvBegin”} \\
\qquad\qquad\qquad\qquad\qquad\qquad \Box\ \ self \in Apps \rightarrow \text{“AppBegin”} \\
\qquad\qquad\qquad\qquad\qquad\qquad \Box\ \ self = ApsId \rightarrow \text{“ApsBegin”}]
\end{array}$$

$$INSTALL\_APP(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“INSTALL\_APP”} \\
\land\ env\_vars' = [env\_vars \text{ EXCEPT } !.applications[app\_[self]].installed = \text{TRUE}] \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
\land\ app\_' = [app\_ \text{ EXCEPT } ![self] = Head(stack[self]).app\_] \\
\land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
\land\ \text{UNCHANGED } \langle app\_vars, aps\_vars, app\_u, app\_up, app\_t, \\
\qquad\qquad\qquad app\_d, perm, app\_c, app\_s, app, i\_, i\_A, \\
\qquad\qquad\qquad i\rangle
\end{array}$$

$$installApp(self) \triangleq INSTALL\_APP(self)$$

$$UNINSTALL\_APP(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“UNINSTALL\_APP”} \\
\land\ env\_vars' = [env\_vars \text{ EXCEPT } !.applications[app\_u[self]].installed = \text{FALSE}] \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
\land\ app\_u' = [app\_u \text{ EXCEPT } ![self] = Head(stack[self]).app\_u] \\
\land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
\land\ \text{UNCHANGED } \langle app\_vars, aps\_vars, app\_, app\_up, app\_t, \\
\qquad\qquad\qquad app\_d, perm, app\_c, app\_s, app, i\_, i\_A, \\
\qquad\qquad\qquad i\rangle
\end{array}$$

$$uninstallApp(self) \triangleq UNINSTALL\_APP(self)$$

$$UPDATE\_APP(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“UPDATE\_APP”} \\
\land\ env\_vars' = [env\_vars \text{ EXCEPT } !.applications[app\_up[self]].version = env\_vars.ap \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
\land\ app\_up' = [app\_up \text{ EXCEPT } ![self] = Head(stack[self]).app\_up] \\
\land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
\land\ \text{UNCHANGED } \langle app\_vars, aps\_vars, app\_, app\_u, app\_t, \\
\qquad\qquad\qquad app\_d, perm, app\_c, app\_s, app, i\_, i\_A, i\rangle
\end{array}$$

$$updateApp(self) \triangleq UPDATE\_APP(self)$$

$$TERMINATED(self) \triangleq \begin{array}[t]{l}
\land\ pc[self] = \text{“TERMINATED”} \\
\land\ env\_vars' = [env\_vars \text{ EXCEPT } !.applications[self].terminated = \text{TRUE}] \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
\land\ app\_t' = [app\_t \text{ EXCEPT } ![self] = Head(stack[self]).app\_t] \\
\land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
\land\ \text{UNCHANGED } \langle app\_vars, aps\_vars, app\_, app\_u, app\_up,
\end{array}$$

```
                                  app_d, perm, app_c, app_s, app, i_, i_A, i⟩

terminate(self) ≜ TERMINATED(self)

DECLARE_PERMISSION(self) ≜ ∧ pc[self] = "DECLARE_PERMISSION"
                           ∧ app_vars′ = [app_vars EXCEPT ![app_d[self]].manifest[perm[self]] = T
                           ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                           ∧ app_d′ = [app_d EXCEPT ![self] = Head(stack[self]).app_d]
                           ∧ perm′ = [perm EXCEPT ![self] = Head(stack[self]).perm]
                           ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                           ∧ UNCHANGED ⟨env_vars, aps_vars, app_, app_u,
                                         app_up, app_t, app_c, app_s, app,
                                         i_, i_A, i⟩

declarePermission(self) ≜ DECLARE_PERMISSION(self)

CHECK_SELF_PERMISSION(self) ≜ ∧ pc[self] = "CHECK_SELF_PERMISSION"
                              ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                              ∧ app_c′ = [app_c EXCEPT ![self] = Head(stack[self]).app_c]
                              ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                              ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars,
                                            app_, app_u, app_up, app_t,
                                            app_d, perm, app_s, app, i_,
                                            i_A, i⟩

checkSelfPermission(self) ≜ CHECK_SELF_PERMISSION(self)

SHOULD_SHOW_REQUEST_PERMISSION_RATIONALE(self) ≜ ∧ pc[self] = "SHOULD_SHOW_REQU
                                                 ∧ pc′ = [pc EXCEPT ![self] = Head(st
                                                 ∧ app_s′ = [app_s EXCEPT ![self] = H
                                                 ∧ stack′ = [stack EXCEPT ![self] = T
                                                 ∧ UNCHANGED ⟨env_vars,
                                                               app_vars,
                                                               aps_vars,
                                                               app_, app_u,
                                                               app_up,
                                                               app_t, app_d,
                                                               perm, app_c,
                                                               app, i_, i_A,
                                                               i⟩

shouldShowRequestPermissionRationale(self) ≜ SHOULD_SHOW_REQUEST_PERMISSION_RATIONAL

REQUEST_PERMISSION(self) ≜ ∧ pc[self] = "REQUEST_PERMISSION"
                           ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                           ∧ app′ = [app EXCEPT ![self] = Head(stack[self]).app]
                           ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
```

```
                                        ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars, app_,
                                                     app_u, app_up, app_t, app_d, perm,
                                                     app_c, app_s, i_, i_A, i⟩

requestPermission(self) ≜ REQUEST_PERMISSION(self)

EnvBegin ≜ ∧ pc[EnvironmentId] = "EnvBegin"
           ∧ IF FALSE
                 THEN ∧ TRUE
                      ∧ pc′ = [pc EXCEPT ![EnvironmentId] = "EnvBegin"]
                 ELSE ∧ pc′ = [pc EXCEPT ![EnvironmentId] = "Done"]
           ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars, stack, app_, app_u,
                        app_up, app_t, app_d, perm, app_c, app_s, app, i_,
                        i_A, i⟩

EnvNext ≜ EnvBegin

AppBegin(self) ≜ ∧ pc[self] = "AppBegin"
                 ∧ IF env_vars.applications[self].terminated ≠ TRUE
                       THEN ∧ ∨ ∧ ∧ app_t′ = [app_t EXCEPT ![self] = self]
                                  ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦ "terminate",
                                                                       pc        ↦ "AppBegin",
                                                                       app_t     ↦ app_t[self]]⟩
                                                                      ∘ stack[self]]
                                ∧ pc′ = [pc EXCEPT ![self] = "TERMINATED"]
                                ∧ UNCHANGED ⟨app_u, app_up⟩
                            ∨ ∧ IF env_vars.applications[self].installed = TRUE
                                    THEN ∧ ∨ ∧ ∧ app_u′ = [app_u EXCEPT ![self] = self]
                                               ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦
                                                                                    pc        ↦
                                                                                    app_u     ↦
                                                                                   ∘ stack[self]]
                                             ∧ pc′ = [pc EXCEPT ![self] = "UNINSTALL_APP"]
                                             ∧ UNCHANGED app_up
                                          ∨ ∧ ∨ ∧ IF env_vars.applications[self].version < MaxAl
                                                        THEN ∧ ∧ app_up′ = [app_up EXCEPT ![se
                                                                  ∧ stack′ = [stack EXCEPT ![self] =



                                                             ∧ pc′ = [pc EXCEPT ![self] = "UPDA
                                                        ELSE ∧ pc′ = [pc EXCEPT ![self] = "AppBe
                                                             ∧ UNCHANGED ⟨stack,
                                                                          app_up⟩
                                                ∨ ∧ TRUE
                                                  ∧ pc′ = [pc EXCEPT ![self] = "AppBegin"]
                                          ∧ UNCHANGED ⟨stack, app_up⟩
                                        ∧ app_u′ = app_u
                                  ELSE  ∧ pc′ = [pc EXCEPT ![self] = "DECLARING_PERMISSION"]
                                        ∧ UNCHANGED ⟨stack, app_u,
                                                     app_up⟩
                              ∧ app_t′ = app_t
                      ELSE  ∧ pc′ = [pc EXCEPT ![self] = "Done"]
                            ∧ UNCHANGED ⟨stack, app_u, app_up, app_t⟩
                ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars, app_, app_d,
                             perm, app_c, app_s, app, i_, i_A, i⟩

DECLARING_PERMISSION(self) ≜ ∧ pc[self] = "DECLARING_PERMISSION"
                             ∧ ∃ p ∈ Permissions :
                                 ∧ ∧ app_d′ = [app_d EXCEPT ![self] = self]
                                   ∧ perm′ = [perm EXCEPT ![self] = p]
                                   ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦  "declarePerr
                                                                      pc        ↦  "INSTALLI
                                                                      app_d     ↦  app_d[self]
                                                                      perm      ↦  perm[self]]⟩
                                                                      ∘ stack[self]]
                                 ∧ pc′ = [pc EXCEPT ![self] = "DECLARE_PERMISSION"]
                             ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars,
                                          app_, app_u, app_up, app_t,
                                          app_c, app_s, app, i_, i_A, i⟩

INSTALLING_APP(self) ≜ ∧ pc[self] = "INSTALLING_APP"
                       ∧ ∧ app_′ = [app_ EXCEPT ![self] = self]
                         ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦  "installApp",
                                                              pc        ↦  "AppBegin",
                                                              app_      ↦  app_[self]]⟩
                                                              ∘ stack[self]]
                       ∧ pc′ = [pc EXCEPT ![self] = "INSTALL_APP"]
                       ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars, app_u,
                                    app_up, app_t, app_d, perm, app_c,
                                    app_s, app, i_, i_A, i⟩

AppNext(self) ≜ AppBegin(self) ∨ DECLARING_PERMISSION(self)
                   ∨ INSTALLING_APP(self)

ApsBegin ≜ ∧ pc[ApsId] = "ApsBegin"
           ∧ IF FALSE
                THEN ∧ TRUE
                     ∧ pc′ = [pc EXCEPT ![ApsId] = "ApsBegin"]
                ELSE ∧ pc′ = [pc EXCEPT ![ApsId] = "Done"]
           ∧ UNCHANGED ⟨env_vars, app_vars, aps_vars, stack, app_, app_u,
                        app_up, app_t, app_d, perm, app_c, app_s, app, i_,
```

$i\_A,\ i\rangle$

$ApsNext \triangleq ApsBegin$

Allow infinite stuttering to prevent deadlock on termination.

$Terminating \triangleq \land \forall self \in ProcSet : pc[self] = \text{"Done"}$
$\qquad\qquad\qquad \land \text{UNCHANGED } vars$

```
Next ≜ EnvNext ∨ ApsNext
          ∨ (∃ self ∈ ProcSet :  ∨ installApp(self) ∨ uninstallApp(self)
                                 ∨ updateApp(self) ∨ terminate(self)
                                 ∨ declarePermission(self)
                                 ∨ checkSelfPermission(self)
                                 ∨ shouldShowRequestPermissionRationale(self)
                                 ∨ requestPermission(self))
          ∨ (∃ self ∈ Apps : AppNext(self))
          ∨ Terminating
```

```
Spec ≜ ∧ Init ∧ □[Next]_vars
       ∧ WF_vars((pc[EnvironmentId] ≠ "EnvBegin") ∧ EnvNext)
       ∧ ∀ self ∈ Apps : ∧ WF_vars((pc[self] ≠ "AppBegin") ∧ AppNext(self))
                         ∧ WF_vars(terminate(self))
                         ∧ WF_vars(uninstallApp(self))
                         ∧ WF_vars(updateApp(self))
                         ∧ WF_vars(declarePermission(self))
                         ∧ WF_vars(installApp(self))
       ∧ WF_vars((pc[ApsId] ≠ "ApsBegin") ∧ ApsNext)
```

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

END TRANSLATION

\* Modification History
\* Last modified *Thu* May 18 19:35:53 *GMT* + 03:30 2023 by *Amirhosein*
\* Created *Fri Apr* 28 08:40:56 *GMT* + 03:30 2023 by *Amirhosein*