─────────────── MODULE *Config* ───────────────

INSTANCE *Naturals*

INSTANCE *FiniteSets*

INSTANCE *Sequences*

INSTANCE *TLC*

──

An empty constant
CONSTANT *Nil*

Transaction type constants
CONSTANTS
    TransactionChange,
    TransactionRollback

Transaction status constants
CONSTANTS
    TransactionPending,
    TransactionValidating,
    TransactionApplying,
    TransactionComplete,
    TransactionFailed

Configuration status constants
CONSTANTS
    ConfigurationPending,
    ConfigurationInitializing,
    ConfigurationUpdating,
    ConfigurationComplete,
    ConfigurationFailed

The set of all nodes
CONSTANT *Node*

Target is the possible targets, paths, and values

Example: $Target \triangleq$ [
    $target1 \mapsto$ [
      $path1 \mapsto$ {"value1", "value2"},
      $path2 \mapsto$ {"value2", "value3"}],
    $target2 \mapsto$ [
      $path2 \mapsto$ {"value3", "value4"},
      $path3 \mapsto$ {"value4", "value5"}]]

CONSTANT *Target*

1

ASSUME $Nil \in$ STRING

ASSUME $TransactionPending \in$ STRING
ASSUME $TransactionValidating \in$ STRING
ASSUME $TransactionApplying \in$ STRING
ASSUME $TransactionComplete \in$ STRING
ASSUME $TransactionFailed \in$ STRING

ASSUME $ConfigurationPending \in$ STRING
ASSUME $ConfigurationInitializing \in$ STRING
ASSUME $ConfigurationUpdating \in$ STRING
ASSUME $ConfigurationComplete \in$ STRING
ASSUME $ConfigurationFailed \in$ STRING

ASSUME $\wedge IsFiniteSet(Node)$
$\qquad \wedge \forall n \in Node :$
$\qquad\qquad \wedge n \notin$ DOMAIN $Target$
$\qquad\qquad \wedge n \in$ STRING

ASSUME $\wedge \forall t \in$ DOMAIN $Target :$
$\qquad\qquad \wedge t \notin Node$
$\qquad\qquad \wedge t \in$ STRING
$\qquad\qquad \wedge \forall p \in$ DOMAIN $Target[t] :$
$\qquad\qquad\qquad IsFiniteSet(Target[t][p])$

TYPE $TransactionType ::= type \in$
$\quad \{TransactionChange,$
$\quad\ TransactionRollback\}$

TYPE $TransactionStatus ::= status \in$
$\quad \{TransactionPending,$
$\quad\ TransactionValidating,$
$\quad\ TransactionApplying,$
$\quad\ TransactionComplete,$
$\quad\ TransactionFailed\}$

TYPE Transaction $\triangleq$ [
$\quad type \quad ::= type \in TransactionType,$
$\quad index ::= index \in Nat,$
$\quad revision ::=$ revision $\in Nat,$
$\quad atomic ::= atomic \in$ BOOLEAN ,
$\quad sync \quad ::= sync \in$ BOOLEAN ,
$\quad changes ::=$ [$target \in$ SUBSET (DOMAIN $Target$) $\mapsto$ [
$\qquad path \in$ SUBSET (DOMAIN $Target[target]$) $\mapsto$ [
$\qquad\quad value ::= value \in$ STRING,
$\qquad\quad delete ::= delete \in$ BOOLEAN]]],
$\quad rollback ::= index \in Nat,$
$\quad status ::= status \in TransactionStatus$]

TYPE $ConfigurationStatus ::= status \in$
 $\{ConfigurationPending,$
  $ConfigurationInitializing,$
  $ConfigurationUpdating,$
  $ConfigurationComplete,$
  $ConfigurationFailed\}$

TYPE Configuration $\triangleq$ [
  id $::= id \in$ STRING,
  revision $::=$ revision $\in Nat,$
  $target ::= target \in$ STRING,
  $paths ::=$ [$path \in$ SUBSET (DOMAIN $Target[target]$) $\mapsto$ [
      $value ::= value \in$ STRING,
      $index ::= index \in Nat,$
      $deleted ::= delete \in$ BOOLEAN]],
  $txIndex ::= txIndex \in Nat,$
  $syncIndex ::= syncIndex \in Nat,$
  $term$ $::= term \in Nat,$
  $status ::= status \in ConfigurationStatus]$

A sequence of transactions
Each transactions contains a record of 'changes' for a set of targets
VARIABLE $transaction$

A record of target configurations
Each configuration represents the desired state of the target
VARIABLE $configuration$

A record of target states
VARIABLE $target$

A record of target masters
VARIABLE $master$

VARIABLE $history$

$vars \triangleq \langle transaction,\ configuration,\ master,\ target,\ history \rangle$

$SetMaster(n,\ t) \triangleq$
  $\wedge\ master[t].master \neq n$
  $\wedge\ master' = [master$ EXCEPT $![t].term\ \ \ = master[t].term + 1,$
                                   $![t].master = n]$
  $\wedge$ UNCHANGED $\langle transaction,\ configuration,\ target,\ history \rangle$

$UnsetMaster(t) \triangleq$
  $\wedge\ master[t].master \neq Nil$
  $\wedge\ master' = [master$ EXCEPT $![t].master = Nil]$
  $\wedge$ UNCHANGED $\langle transaction,\ configuration,\ target,\ history \rangle$

$ValidValues(t, p) \triangleq$
  UNION $\{\{[value \mapsto v, \ delete \mapsto \text{FALSE}] : v \in Target[t][p]\}, \{[value \mapsto Nil, \ delete \mapsto \text{TRUE}]\}\}$

$ValidPaths(t) \triangleq$
  UNION $\{\{v @@ [path \mapsto p] : v \in ValidValues(t, p)\} : p \in \text{DOMAIN } Target[t]\}$

$ValidTargets \triangleq$
  UNION $\{\{p @@ [target \mapsto t] : p \in ValidPaths(t)\} : t \in \text{DOMAIN } Target\}$

$ValidPath(s, t, p) \triangleq$
  LET $value \triangleq \text{CHOOSE } v \in s : v.target = t \land v.path = p$
  IN
    $[value \ \mapsto value.value,$
     $delete \ \mapsto value.delete]$

$ValidTarget(s, t) \triangleq$
  $[p \in \{v.path : v \in \{v \in s : v.target = t\}\} \mapsto ValidPath(s, t, p)]$

$ValidChange(s) \triangleq$
  $[t \in \{v.target : v \in s\} \mapsto ValidTarget(s, t)]$

$ValidChanges \triangleq$
  LET $changeSets \triangleq \{s \in \text{SUBSET } ValidTargets :$
                           $\forall t \in \text{DOMAIN } Target \quad :$
                             $\forall p \in \text{DOMAIN } Target[t] :$
                               $Cardinality(\{v \in s : v.target = t \land v.path = p\}) \leq 1\}$
  IN
    $\{ValidChange(s) : s \in changeSets\}$

$NextIndex \triangleq$
  IF DOMAIN $transaction = \{\}$ THEN
    1
  ELSE
    LET $i \triangleq \text{CHOOSE } i \in \text{DOMAIN } transaction :$
        $\forall j \in \text{DOMAIN } transaction :$
          $i \geq j$
    IN $\quad i + 1$

$Change(c) \triangleq$
  $\land \quad transaction' = transaction @@ (NextIndex :> [type \qquad \mapsto TransactionChange,$
                                              $index \quad \mapsto NextIndex,$
                                              $atomic \ \mapsto \text{FALSE},$
                                              $sync \qquad \mapsto \text{FALSE},$
                                              $changes \mapsto c,$

4

$$sources \mapsto \langle\rangle,$$
$$status \quad \mapsto TransactionPending])$$
$$\wedge \quad \text{UNCHANGED} \ \langle configuration, \ master, \ target, \ history\rangle$$

Add a rollback to the transaction log
$Rollback(t) \ \triangleq$
    $\wedge \ transaction[t].type = TransactionChange$
    $\wedge \ transaction' = transaction \ @@ \ (NextIndex :> [type \quad \mapsto TransactionRollback,$
                                                $index \quad \mapsto NextIndex,$
                                                $atomic \quad \mapsto \text{FALSE},$
                                                $sync \quad \mapsto \text{FALSE},$
                                                $rollback \ \mapsto t,$
                                                $status \quad \mapsto TransactionPending])$
    $\wedge \ \text{UNCHANGED} \ \langle configuration, \ master, \ target, \ history\rangle$

This section models the Transaction log reconciler.

$ReconcileChange(n, i) \ \triangleq$
    If the transaction is Pending, begin validation if the prior transaction
    has already been applied. This simplifies concurrency control in the controller
    and guarantees transactions are applied to the configurations in sequential order.
    $\vee \ \wedge \ transaction[i].status = TransactionPending$
        $\wedge \ \vee \ \wedge \ i - 1 \in \text{DOMAIN} \ transaction$
                $\wedge \ transaction[i - 1].status \in \{TransactionComplete, \ TransactionFailed\}$
            $\vee \ i - 1 \notin \text{DOMAIN} \ transaction$
        $\wedge \ transaction' = [transaction \ \text{EXCEPT} \ ![i].status = TransactionValidating]$
        $\wedge \ \text{UNCHANGED} \ \langle configuration, \ history\rangle$
    If the transaction is in the Validating state, compute and validate the
    Configuration for each target.
    $\vee \ \wedge \ transaction[i].status = TransactionValidating$
        If validation fails any target, mark the transaction Failed.
        If validation is successful, proceed to Applying.
        $\wedge \ \exists \ valid \in \text{BOOLEAN} \ :$
            $\text{IF} \ valid \ \text{THEN}$
                $\wedge \ transaction' = [transaction \ \text{EXCEPT} \ ![i].status = TransactionApplying]$
            ELSE
                $\wedge \ transaction' = [transaction \ \text{EXCEPT} \ ![i].status = TransactionFailed]$
        $\wedge \ \text{UNCHANGED} \ \langle configuration, \ history\rangle$
    If the transaction is in the Applying state, update the Configuration for each
    target and Complete the transaction.
    $\vee \ \wedge \ transaction[i].status = TransactionApplying$
        Update the target configurations, adding the transaction index to each updated path
        $\wedge \ configuration' = [$
            $t \in \text{DOMAIN} \ Target \mapsto$
                $\text{IF} \ t \in \text{DOMAIN} \ transaction[i].changes \ \text{THEN}$

$$[configuration[t] \text{ EXCEPT}$$
$$!.paths = [p \in \text{DOMAIN } transaction[i].changes[t] \mapsto$$
$$[index \quad \mapsto transaction[i].index,$$
$$value \quad \mapsto transaction[i].changes[t][p].value,$$
$$deleted \mapsto transaction[i].changes[t][p].delete]]$$
$$@@ \ configuration[t].paths,$$
$$!.txIndex = transaction[i].index,$$
$$!.status \quad = ConfigurationPending]$$
$$\text{ELSE}$$
$$configuration[t]]$$
$$\wedge \ history' = [r \in \text{DOMAIN } Target \mapsto Append(history[r], \ configuration'[r])]$$
$$\wedge \ transaction' = [transaction \text{ EXCEPT}$$
$$![i].status \quad = TransactionComplete,$$
$$![i].sources = [t \in \text{DOMAIN } transaction[i].changes \mapsto$$
$$\text{LET } updatePaths \ \triangleq \ \{p \in \text{DOMAIN } transaction[i].changes[t] :$$
$$\neg transaction[i].changes[t][p].delete\}$$
$$\text{IN} \quad [p \in updatePaths \cap \text{DOMAIN } configuration[t].paths \mapsto configuration[t].paths[p]]]]$$

$ReconcileRollback(n, \ i) \ \triangleq$

$$\vee \ \wedge \ transaction[i].status = TransactionPending$$
$$\wedge \ \vee \ \wedge \ i - 1 \in \text{DOMAIN } transaction$$
$$\wedge \ transaction[i - 1].status \in \{TransactionComplete, \ TransactionFailed\}$$
$$\vee \ i - 1 \notin \text{DOMAIN } transaction$$
$$\wedge \ transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionValidating]$$
$$\wedge \ \text{UNCHANGED } \langle configuration, \ history \rangle$$

$$\vee \ \wedge \ transaction[i].status = TransactionValidating$$
$$\wedge \ \vee \ \wedge \ transaction[transaction[i].rollback].status = TransactionComplete$$

$$\wedge \ \vee \ \wedge \ transaction[i].rollback \ \in \text{DOMAIN } transaction$$
$$\wedge \ \text{LET } canRollback \ \triangleq \ \forall \ t \in \text{DOMAIN } transaction[transaction[i].rollback].changes :$$
$$\forall \ p \in \text{DOMAIN } transaction[transaction[i].rollback].changes[t] :$$
$$configuration[t].paths[p].index = transaction[i].rollback$$
$$\text{IN}$$
$$\text{IF } canRollback \text{ THEN}$$
$$\wedge \ transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionApplying]$$
$$\text{ELSE}$$
$$\wedge \ transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionFailed]$$

$$\lor \land transaction[i].rollback \notin \text{DOMAIN } transaction$$
$$\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionFailed]$$
$$\lor \land transaction[transaction[i].rollback].status = TransactionFailed$$
$$\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionFailed]$$
$$\land \text{UNCHANGED } \langle configuration, \ history \rangle$$

target and Complete the transaction.
$$\lor \land transaction[i].status = TransactionApplying$$

Revert the target configurations
$$\land configuration' = [$$
$$t \in \text{DOMAIN } Target \mapsto$$
$$\text{IF } t \in \text{DOMAIN } transaction[transaction[i].rollback].changes \text{ THEN}$$
$$\text{LET } adds \quad \triangleq \{p \in \text{DOMAIN } transaction[transaction[i].rollback].changes[t] :$$
$$\land p \notin \text{DOMAIN } transaction[transaction[i].rollback].sources[t]$$
$$\land \neg transaction[transaction[i].rollback].changes[t][p].delete\}$$
$$updates \quad \triangleq \{p \in \text{DOMAIN } transaction[transaction[i].rollback].changes[t] :$$
$$\land p \in \text{DOMAIN } transaction[transaction[i].rollback].sources[t]$$
$$\land \neg transaction[transaction[i].rollback].changes[t][p].delete\}$$
$$removes \quad \triangleq \{p \in \text{DOMAIN } transaction[transaction[i].rollback].changes[t] :$$
$$\land p \in \text{DOMAIN } transaction[transaction[i].rollback].sources[t]$$
$$\land transaction[transaction[i].rollback].changes[t][p].delete\}$$
$$changes \quad \triangleq adds \cup updates \cup removes$$
$$unchanges \triangleq \text{DOMAIN } configuration[t].paths \setminus changes$$
$$\text{IN}$$
$$[configuration[t] \text{ EXCEPT}$$
$$!.paths = [p \in unchanges \mapsto configuration[t].paths[p]]$$
$$@@ [p \in updates \cup removes \mapsto$$
$$transaction[transaction[i].rollback].sources[t][p]],$$
$$!.txIndex = transaction[i].index,$$
$$!.status \quad = ConfigurationPending]$$
$$\text{ELSE}$$
$$configuration[t]]$$
$$\land history' = [r \in \text{DOMAIN } Target \mapsto Append(history[r], \ configuration'[r])]$$
$$\land transaction' = [transaction \text{ EXCEPT } ![i].status = TransactionComplete]$$

Reconcile the transaction log
$$ReconcileTransaction(n, \ i) \quad \triangleq$$
$$\land \lor \land transaction[i].type = TransactionChange$$
$$\land ReconcileChange(n, \ i)$$
$$\lor \land transaction[i].type = TransactionRollback$$
$$\land ReconcileRollback(n, \ i)$$
$$\land \text{UNCHANGED } \langle master, \ target \rangle$$

$ReconcileConfiguration(n,\ c)\ \triangleq$
$\quad \wedge\ \vee\ \wedge\ configuration[c].status = ConfigurationPending$
$\qquad\quad \wedge\ master[configuration[c].target].master \neq Nil$

                If the configuration is marked *ConfigurationPending* and mastership
                has changed (indicated by an increased mastership term), mark the
                configuration *ConfigurationInitializing* to force full re-synchronization.

$\qquad\quad \wedge\ \vee\ \wedge\ master[configuration[c].target].term > configuration[c].term$
$\qquad\qquad\quad \wedge\ configuration' = [configuration\ \text{EXCEPT}\ ![c].status = ConfigurationInitializing,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\ ![c].term\ \ = master[configuration[c].target].term]$
$\qquad\qquad\quad \wedge\ history' = [history\ \text{EXCEPT}\ ![c] = Append(history[c],\ configuration'[c])]$

                If the configuration is marked *ConfigurationPending* and the values have
                changed (determined by comparing the transaction index to the last *sync*
                index), mark the configuration *ConfigurationUpdating* to push the changes
                to the target.

$\qquad\qquad\ \vee\ \wedge\ master[configuration[c].target].term = configuration[c].term$
$\qquad\qquad\qquad \wedge\ configuration[c].syncIndex < configuration[c].txIndex$
$\qquad\qquad\qquad \wedge\ configuration' = [configuration\ \text{EXCEPT}\ ![c].status = ConfigurationUpdating]$
$\qquad\qquad\qquad \wedge\ history' = [history\ \text{EXCEPT}\ ![c] = Append(history[c],\ configuration'[c])]$
$\qquad\quad \wedge\ \text{UNCHANGED}\ \langle target \rangle$
$\quad\ \vee\ \wedge\ configuration[c].status = ConfigurationInitializing$
$\qquad \wedge\ master[configuration[c].target].master = n$

           Merge the configuration paths with the target paths, removing paths
           that have been marked deleted

$\qquad \wedge\ \text{LET}\ deletePaths\ \triangleq\ \{p \in \text{DOMAIN}\ configuration[c].paths : configuration[c].paths[p].deleted\}$
$\qquad\qquad\qquad configPaths\ \triangleq\ \text{DOMAIN}\ configuration[c].paths \setminus deletePaths$
$\qquad\qquad\qquad targetPaths\ \triangleq\ \text{DOMAIN}\ target[configuration[c].target] \setminus deletePaths$
$\qquad\quad \text{IN}$
$\qquad\qquad \wedge\ target' = [target\ \text{EXCEPT}\ ![configuration[c].target] =$
$\qquad\qquad\qquad [p \in configPaths \mapsto [value \mapsto configuration[c].paths[p]]]$
$\qquad\qquad\qquad\quad @@\ [p \in targetPaths \mapsto target[configuration[c].target][p]]]$

                Set the configuration's status to Complete

$\qquad \wedge\ configuration' = [configuration\ \text{EXCEPT}\ ![c].status\qquad = ConfigurationComplete,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![c].syncIndex = configuration[c].txIndex]$
$\qquad \wedge\ history' = [history\ \text{EXCEPT}\ ![c] = Append(history[c],\ configuration'[c])]$

If the configuration is marked *ConfigurationUpdating*, we only need to
push paths that have changed since the target was initialized or last
updated by the controller. The set of changes made since the last
synchronization are identified by comparing the index of each path-value
to the last synchronization index, *syncIndex*

$\quad\ \vee\ \wedge\ configuration[c].status = ConfigurationUpdating$
$\qquad \wedge\ master[configuration[c].target].master = n$

        Compute the set of updated and deleted paths by comparing
        their indexes to the target s last sync index.

$\qquad \wedge\ \text{LET}\ updatePaths\ \triangleq\ \{p \in \text{DOMAIN}\ configuration[c].paths :$
$\qquad\qquad\qquad\qquad\qquad\qquad\quad configuration[c].paths[p].index > configuration[c].syncIndex\}$

8

$$
\begin{aligned}
deletePaths &\triangleq \{p \in updatePaths : configuration[c].paths[p].deleted\} \\
configPaths &\triangleq updatePaths \setminus deletePaths \\
targetPaths &\triangleq \text{DOMAIN } target[configuration[c].target] \setminus deletePaths
\end{aligned}
$$

IN

Update the target paths by adding/updating paths that have changed and
removing paths that have been deleted since the last *sync*.

$\land\ target' = [target \text{ EXCEPT } ![configuration[c].target] =$
$\quad [p \in configPaths \mapsto configuration[c].paths[p]]$
$\quad\quad @@ [p \in targetPaths \mapsto target[configuration[c].target][p]]]$
$\land\ configuration' = [configuration \text{ EXCEPT } ![c].status \quad = ConfigurationComplete,$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad ![c].syncIndex = configuration[c].txIndex]$
$\land\ history' = [history \text{ EXCEPT } ![c] = Append(history[c], configuration'[c])]$

If the configuration is not already *ConfigurationPending* and mastership
has been lost revert it. This can occur when the connection to the
target has been lost and the mastership is no longer valid.

TODO: We still need to model mastership changes

$\lor\ \land\ configuration[c].status \neq ConfigurationPending$
$\quad \land\ master[configuration[c].target].master = Nil$
$\quad \land\ configuration' = [configuration \text{ EXCEPT } ![c].status = ConfigurationPending]$
$\quad \land\ history' = [history \text{ EXCEPT } ![c] = Append(history[c], configuration'[c])]$
$\quad \land\ \text{UNCHANGED } \langle target \rangle$
$\land\ \text{UNCHANGED } \langle transaction,\ master \rangle$

Init and next state predicates

$Init \triangleq$
$\quad \land\ transaction = \langle \rangle$
$\quad \land\ configuration = [t \in \text{DOMAIN } Target \mapsto$
$\quad\quad\quad\quad\quad\quad [target \mapsto t,$
$\quad\quad\quad\quad\quad\quad paths \mapsto$
$\quad\quad\quad\quad\quad\quad\quad [path \in \{\} \mapsto$
$\quad\quad\quad\quad\quad\quad\quad\quad [path \quad \mapsto path,$
$\quad\quad\quad\quad\quad\quad\quad\quad value \quad \mapsto Nil,$
$\quad\quad\quad\quad\quad\quad\quad\quad index \quad \mapsto 0,$
$\quad\quad\quad\quad\quad\quad\quad\quad deleted \mapsto \text{FALSE}]],$
$\quad\quad\quad\quad\quad\quad txIndex \quad \mapsto 0,$
$\quad\quad\quad\quad\quad\quad syncIndex \mapsto 0,$
$\quad\quad\quad\quad\quad\quad term \quad\quad \mapsto 0,$
$\quad\quad\quad\quad\quad\quad status \quad\quad \mapsto ConfigurationPending]]$
$\quad \land\ target\ = [t \in \text{DOMAIN } Target \mapsto$
$\quad\quad\quad\quad\quad [path \in \{\} \mapsto$
$\quad\quad\quad\quad\quad\quad [value \mapsto Nil]]]$
$\quad \land\ master = [t \in \text{DOMAIN } Target \mapsto [master \mapsto Nil,\ term \mapsto 0]]$
$\quad \land\ history = [t \in \text{DOMAIN } Target \mapsto \langle \rangle]$

$Next \triangleq$
  $\lor \exists\, c \in ValidChanges :$
   $Change(c)$
  $\lor \exists\, t \in \text{DOMAIN } transaction :$
   $Rollback(t)$
  $\lor \exists\, n \in Node :$
   $\exists\, t \in \text{DOMAIN } Target :$
    $SetMaster(n, t)$
  $\lor \exists\, t \in \text{DOMAIN } Target :$
   $UnsetMaster(t)$
  $\lor \exists\, n \in Node :$
   $\exists\, t \in \text{DOMAIN } transaction :$
    $ReconcileTransaction(n, t)$
  $\lor \exists\, n \in Node :$
   $\exists\, c \in \text{DOMAIN } configuration :$
    $ReconcileConfiguration(n, c)$

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

$Inv \triangleq$
  $\land\; \forall\, a, b \in \text{DOMAIN } transaction :$
   $transaction[a].index > transaction[b].index \Rightarrow$
    $(transaction[a].status \in \{TransactionComplete,\ TransactionFailed\} \Rightarrow$
     $transaction[b].status \in \{TransactionComplete,\ TransactionFailed\})$
  $\land\; \forall\, t \in \text{DOMAIN } Target :$
   $\forall\, c \in \text{DOMAIN } history[t] :$
    $\land\; configuration[t].txIndex \geq history[t][c].txIndex$
    $\land\; configuration[t].syncIndex \geq history[t][c].syncIndex$

\ * Modification History
\ * Last modified *Tue Jan* 18 16:25:57 *PST* 2022 by *jordanhalterman*
\ * Created *Wed Sep* 22 13:22:32 *PDT* 2021 by *jordanhalterman*