```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; register-user.lisp
;
; Created on March 17, 2013 by Matthew A. Crist.
;
; This file will invoke the actions required to verify that a user has the
; ability to perform actions on this server.  This is used to test the
; facility of the server and does not provide direct verification for use
; of the services of the server.  (SEE isInAddressBook IN
; ADDRESS-BOOK.LISP)
;
; XML Document Format:
; ------------------------------------------------------------------------
; <?xml version="1.0"?>
; <!DOCTYPE verify SYSTEM "dtd/verify-user.dtd">
; <verify>
;    <domain>localhost</domain>
;    <name>matthew.crist</name>
;    <password>simulation</password>
;     <location>128.0.0.2</location>
; </verify>
;
; CHANGE LOG:
; ------------------------------------------------------------------------
; 2013-03-17    -   Initial conception of this file.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(in-package "ACL2")

(include-book "../../../include/io-utilities" :uncertified-okp t)
(include-book "../../../include/remote-actions" :uncertified-okp t)
(include-book "../../../include/xml-scanner" :uncertified-okp t)
(include-book "../address-book" :uncertified-okp t)

; (getDomain tokens)
; Acquires the domain of the verification request from the token list.
; tokens    -   the scanned tokens from the incoming XML verify request.
(defun getDomain (tokens)
    (if (endp tokens)
        nil
        (if (equal "<domain>" (caar tokens))
            (caadr tokens)
            (getDomain (cdr tokens)))))

; (getName tokens)
; Acquires the name of the verification request from the token list.
; tokens    -   the scanned tokens from the incoming XML verify request.
(defun getName (tokens)
    (if (endp tokens)
        nil
        (if (equal "<name>" (caar tokens))
            (caadr tokens)
            (getName (cdr tokens)))))

; (getPassword tokens)
; Acquires the password of the verification request from the token list.
; tokens    -   the scanned tokens from the incoming XML verify request.
(defun getPassword (tokens)
    (if (endp tokens)
        nil
        (if (equal "<password>" (caar tokens))
            (caadr tokens)
            (getPassword (cdr tokens)))))

; (getLocation tokens)
; Acquires the physical location of the user that is being verified for
; network connectivity on the verification process.
```

```
; tokens - the scanned tokens from the incoming XML verify request.
(defun getLocation (tokens)
    (if (endp tokens)
        nil
        (if (equal "<location>" (caar tokens))
            (caadr tokens)
            (getLocation (cdr tokens)))))

; (verifyUser user addressBook)
; Verifies that a user is present in the address-book.  This function is
; essentially the same as the isInAdressBook predicate in the address-book
; file, but is placed in a "wrapper" here for obfuscation.
(defun verifyUser (user addressBook)
    (if (endp addressBook)
        nil
        (if (equal (car addressBook) user)
            t
            (verifyUser user (cdr addressBook)))))

(set-state-ok t)
(set-ignore-ok t)

; (testUser userXML addressBookXML)
; Writes a server action file based on the result of a user being verified
; If the user is accepted, it will return a response to the client that
; verification was successful.  If not, it will write back a failure.
(defun testUser (userXML addressBookXML state)
    (let* ((userTokens (tokenizeXML userXML))
             (domain (getDomain   userTokens))
           (name   (getName     userTokens))
              (pass   (getPassword userTokens))
              (loc    (getLocation userTokens))
              (user   (list domain name pass))
              (abook  (getAddressBook (tokenizeXML addressBookXML))))
        (if (verifyUser user abook)
            (mv-let (error state)
                (string-list->file "incoming/user/verify/server-action.xml"
                      (list "<?xml version='1.0'?>"
                       "<!DOCTYPE message SYSTEM 'dtd/system-message.dtd'>"
                       "<message>"
                                 (concatenate 'string "   <action>" "ACCEPT" "</action>")
                                 (concatenate 'string "   <location>" loc "</location>")
                             "</message>")
                    state)
                (if error
                    (mv error state)
                    (mv "Action written successfully!" state)))
            (mv-let (error state)
                (string-list->file "incoming/user/verify/server-action.xml"
                      (list "<?xml version='1.0'?>"
                       "<!DOCTYPE message SYSTEM 'dtd/system-message.dtd'>"
                       "<message>"
                                 (concatenate 'string "   <action>" "REJECT" "</action>")
                                 (concatenate 'string "   <location>" loc "</location>")
                             "</message>")
                    state)
                (if error
                    (mv error state)
                    (mv "Action written successfully!" state))))))
```