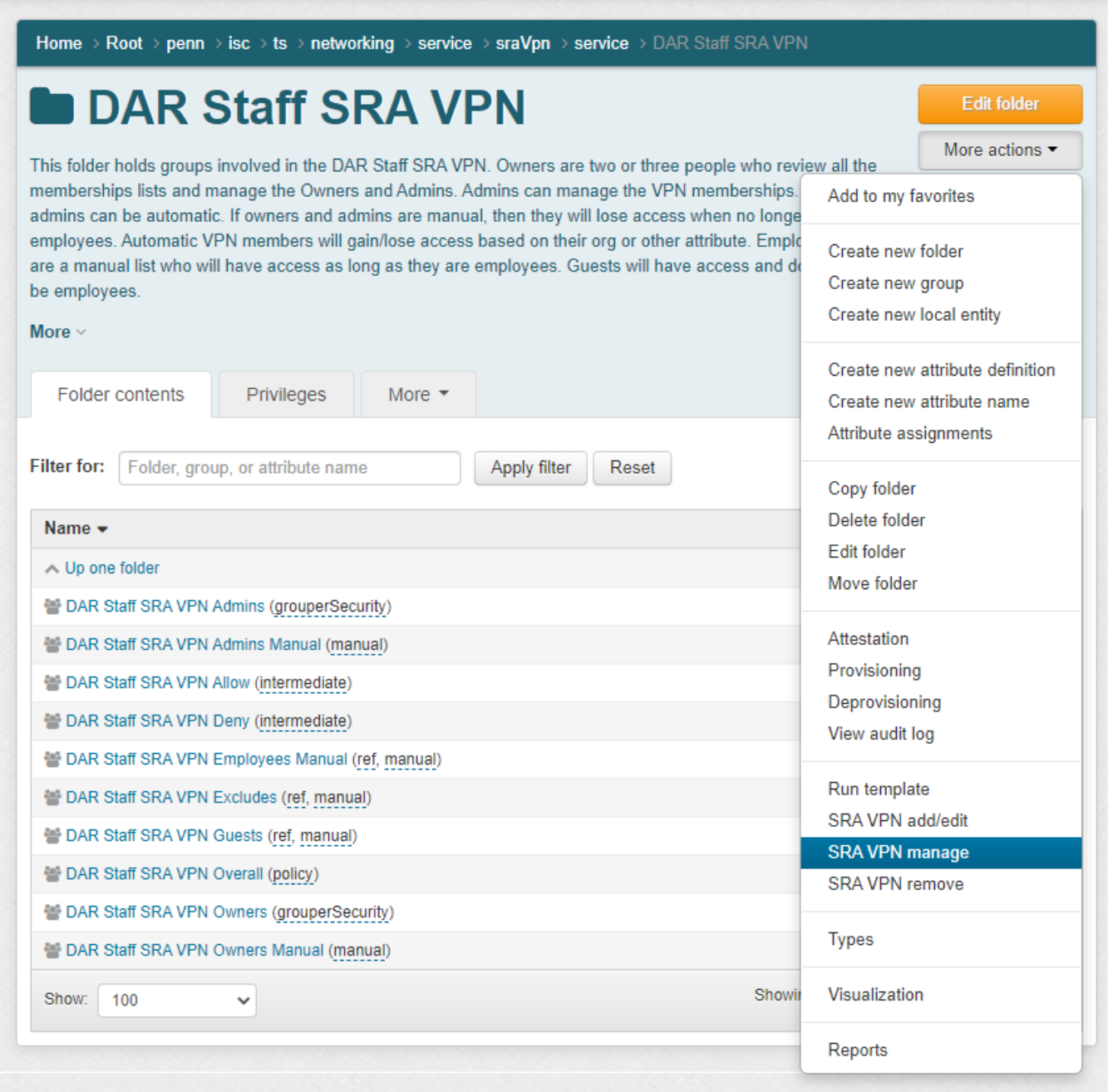
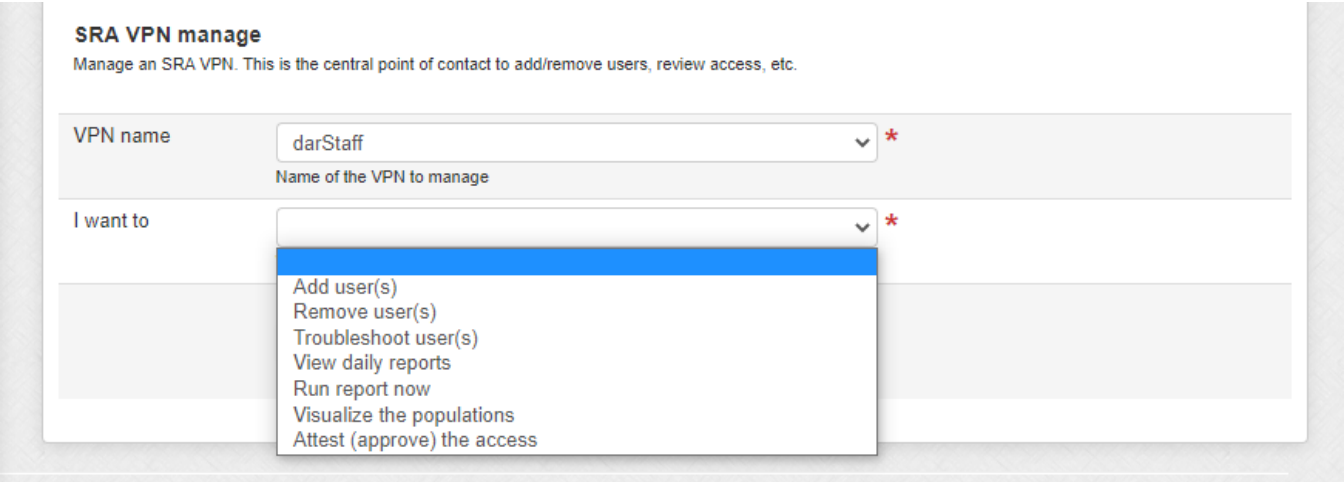
Grouper custom template via GSH delegated VPN management - manage

This GSH template allows school and center VPN admins to manage their instance of a dept VPN

Manage VPN

Central networking staff can see all VPNs and department admins can see only the VPNs they are the owner or admin of

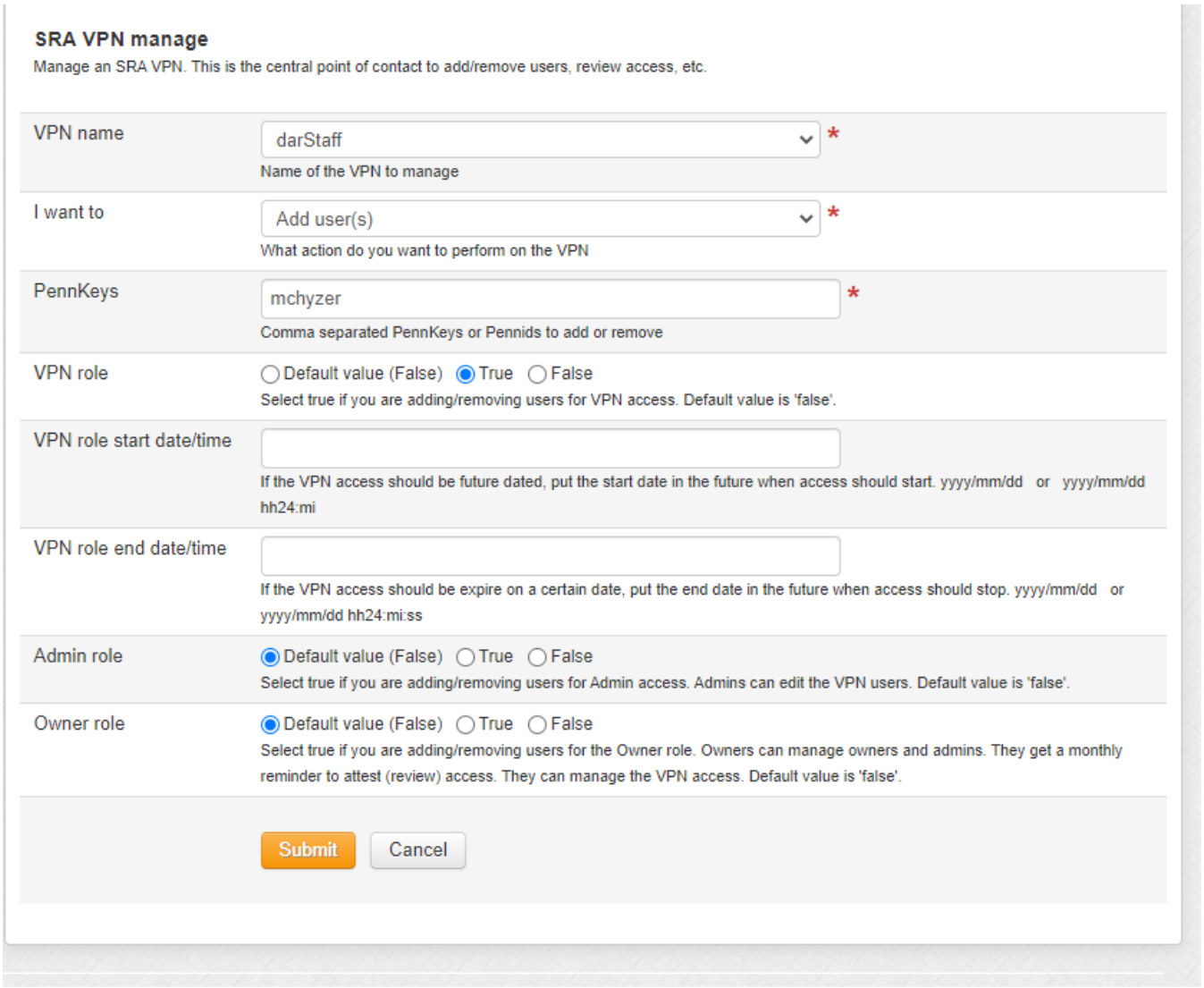




Add a user to VPN role

This will:

1. Remove from Excludes if adding
2. Remove from Guests if employee
3. Add users to vpn role as employee if employee, or edit if already a manual employee
4. Add users to vpn role as guest if not employee
5. Lets see if things are ok with no enabled date
6. Lets see if things are ok with no enabled date



GSH script

```
boolean roleIsVpnUsers = StringUtils.equals(gsh_input_role, "vpn");
   boolean roleIsAdmins = StringUtils.equals(gsh_input_role, "admin");
   boolean roleIsOwners = StringUtils.equals(gsh_input_role, "owner");

   // 1. vpn is required (this should never happen)
   if (StringUtils.isBlank(gsh_input_projectSystemName)) {

     gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_projectSystemName",
        "Error: VPN is required!");

   }
```

```
    // 2. vpn needs to exist (this should never happen)
    String projectStemName = "penn:isc:ts:networking:service:sraVpn:service:" + gsh_input_projectSystemName;
    Stem projectFolder = StemFinder.findByName(gsh_builtin_grouperSession, projectStemName, false);
    if (projectFolder == null) {
      gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_projectSystemName",
          "Error: Cannot find VPN configuration! '" + gsh_input_projectSystemName + "'");
    }

    // 3. check valid action (this should never happen)
    boolean actionAddUser = StringUtils.equals(gsh_input_action, "addUser");
    boolean actionRemoveUser = StringUtils.equals(gsh_input_action, "removeUser");
    boolean actionTroubleShoot = StringUtils.equals(gsh_input_action, "troubleshoot");
    boolean actionViewReports = StringUtils.equals(gsh_input_action, "viewReports");
    boolean actionRunReportNow = StringUtils.equals(gsh_input_action, "runReportNow");
    boolean actionVisualize = StringUtils.equals(gsh_input_action, "visualize");
    boolean actionAttest = StringUtils.equals(gsh_input_action, "attest");
    if (!actionAddUser && !actionRemoveUser && !actionTroubleShoot && !actionViewReports && !actionRunReportNow && !actionVisualize && !
actionAttest) {

      gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_action",
          "Error: Action cannot be found '" + gsh_input_action + "'!");

    }

    // Do not proceed is there is an error
    if (GrouperUtil.length(gsh_builtin_gshTemplateOutput.getValidationLines()) > 0) {
      gsh_builtin_gshTemplateOutput.assignIsError(true);
      GrouperUtil.gshReturn();
    }

    // 4. make sure the user is an Admin (or wheel or networking admin) (note: Owners are automatically Admins) (this should never happen)
    Group adminsGroup = GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnAdmins", true);
    if (!adminsGroup.hasMember(gsh_builtin_subject) && !adminsGroup.canHavePrivilege(gsh_builtin_subject, "updaters", false)) {
      gsh_builtin_gshTemplateOutput.addValidationLine(
          "Error: User is not an Admin for this VPN!");
    }

    // 5. need to pick a role (not sure how this could happen)
    if ((actionAddUser || actionRemoveUser) && (!roleIsVpnUsers && !roleIsAdmins && !roleIsOwners)) {
      gsh_builtin_gshTemplateOutput.addValidationLine(
          "Error: Select a role to manage");
    }

    // 6. see if owner
    Group ownersGroup = GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnOwners", true);
    boolean isOwner = ownersGroup.hasMember(gsh_builtin_subject);

    // 7. ony owner can manage Admins and Owners
    if (!isOwner && (roleIsAdmins || roleIsOwners) && !adminsGroup.canHavePrivilege(gsh_builtin_subject, "updaters", false)) {
      gsh_builtin_gshTemplateOutput.addValidationLine(
          "Error: Only Owners can manage Admins and Owners");
    }

    Group employeeGroup = GroupFinder.findByName(gsh_builtin_grouperSession, "penn:community:employeeOrContractorIncludingUphs", true);

    // 8. User must be an employee (this should never happen)
    if (!employeeGroup.hasMember(gsh_builtin_subject)) {
      gsh_builtin_gshTemplateOutput.addValidationLine(
          "Error: User is not an employee!");
    }

    // 9. Check Pennkeys
    String[] pennkeys = null;
```

```
    // convert owner pennkeys into subjects
    Subject[] subjects = null;
    boolean[] subjectIsEmployee = null;
    Group overallGroup =         GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnOverall", true);
    boolean[] subjectIsOverall = null;
    Group allowGroup =           GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnAllow", true);
    boolean[] subjectIsAllow = null;
    Group excludesGroup =        GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnExcludes", true);
    Group employeesManualGroup = GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnEmployeesManual", true);
    Group guestsGroup =          GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnGuests", true);
    Group adminsManualGroup =    GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnAdminsManual", true);
    Group ownersManualGroup =    GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName +
"SraVpnOwnersManual", true);

    boolean[] subjectIsExclude = null;
    boolean[] subjectIsEmployeesManual = null;
    boolean[] subjectIsAdmin = null;
    boolean[] subjectIsOwner = null;
    boolean[] subjectIsAdminManual = null;
    boolean[] subjectIsOwnerManual = null;
    boolean[] subjectIsGuest = null;
    List<String>[] otherVpnProjectSystemNamesForUser = null;
    if (!StringUtils.isBlank(gsh_input_pennkeys)) {
      pennkeys = GrouperUtil.splitTrim(gsh_input_pennkeys, ",");
      subjects = new Subject[pennkeys.length];
      subjectIsEmployee = new boolean[pennkeys.length];
      subjectIsOverall = new boolean[pennkeys.length];
      subjectIsAllow = new boolean[pennkeys.length];
      subjectIsExclude = new boolean[pennkeys.length];
      subjectIsAdmin = new boolean[pennkeys.length];
      subjectIsOwner = new boolean[pennkeys.length];
      subjectIsGuest = new boolean[pennkeys.length];
      subjectIsAdminManual = new boolean[pennkeys.length];
      subjectIsOwnerManual = new boolean[pennkeys.length];
      subjectIsEmployeesManual = new boolean[pennkeys.length];
      otherVpnProjectSystemNamesForUser = new List[pennkeys.length];
      for (int i=0;i<pennkeys.length;i++) {
        Subject subject = SubjectFinder.findByIdOrIdentifierAndSource(pennkeys[i], "pennperson", false);
        if (subject == null) {
          gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_pennkeys",
            "Error: PennKey cannot be found '" + pennkeys[i] + "'!");
        }
        subjects[i] = subject;

        // in all cases we need to know if the person is an employee
        subjectIsEmployee[i] = employeeGroup.hasMember(subject);

        // we only need to know overall if doing VPN role
        subjectIsOverall[i] = (roleIsVpnUsers || actionTroubleShoot) && overallGroup.hasMember(subjects[i]);
        subjectIsAllow[i] = (roleIsVpnUsers || actionTroubleShoot) && allowGroup.hasMember(subjects[i]);
        subjectIsExclude[i] = (roleIsVpnUsers || actionTroubleShoot) && excludesGroup.hasMember(subjects[i]);
        subjectIsGuest[i] = (roleIsVpnUsers || actionTroubleShoot) && guestsGroup.hasMember(subjects[i]);
        subjectIsEmployeesManual[i] = (roleIsVpnUsers || actionTroubleShoot) && employeesManualGroup.hasMember(subjects[i]);

        // we only need to know admin if doing admin role
        subjectIsAdmin[i] = (roleIsAdmins || actionTroubleShoot) && adminsGroup.hasMember(subject);
        subjectIsAdminManual[i] = (roleIsAdmins || actionTroubleShoot) && adminsManualGroup.hasMember(subjects[i]);

        // we only need to know owner if doing admin role
        subjectIsOwner[i] = (roleIsOwners || actionTroubleShoot) && ownersGroup.hasMember(subject);
        subjectIsOwnerManual[i] = (roleIsOwners || actionTroubleShoot) && ownersManualGroup.hasMember(subjects[i]);
```

```
      // 10. Admins / Owners must be employees
      if ((roleIsAdmins || roleIsOwners) && !subjectIsEmployee[i]) {
        gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_pennkeys",
          "Error: PennKey '" + pennkeys[i] + "' must be an employee to be an Admin/Owner: " + employeeGroup.getName() + "!");
      }

      if ((actionAddUser && roleIsVpnUsers) || actionTroubleShoot) {
        // see what other vpns
        // 11.5 make sure not in other vpns, can only be in one
        otherVpnProjectSystemNamesForUser[i] = new GcDbAccess().sql("select distinct gs.extension as stem_extension from grouper_stems gs,
grouper_memberships_lw_v gmlv "
            + "where gmlv.subject_source = 'pennperson' and gs.name like 'penn:isc:ts:networking:service:sraVpn:service:%' "
            + "and gmlv.group_name like 'penn:isc:ts:networking:service:sraVpn:service:%' "
            + "and gmlv.group_name = 'penn:isc:ts:networking:service:sraVpn:service:' || gs.extension || ':' || gs.extension || 'SraVpnOverall' "
            + "and gmlv.list_name = 'members' and gmlv.subject_id = ?").addBindVar(subjects[i].getId()).selectList(String.class);
        //remove this one
        otherVpnProjectSystemNamesForUser[i].remove(gsh_input_projectSystemName);
      }
    }
  }

  // 11. Check Dates
  java.sql.Timestamp vpnStartTime = null;
  java.sql.Timestamp vpnEndTime = null;
  if (actionAddUser && roleIsVpnUsers) {

    if (!StringUtils.isBlank(gsh_input_vpnStartDate)) {
      try {
        vpnStartTime = GrouperUtil.stringToTimestamp(gsh_input_vpnStartDate);
        if (vpnStartTime.getTime() < System.currentTimeMillis()) {
          gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_vpnStartDate", "Error: VPN role start date/time must be in the future");
        }
      } catch (Exception e) {
        gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_vpnStartDate", "Error: VPN role start date/time must be 'yyyy/mm/dd' or 'yyyy/mm/dd
hh24:mi:ss'");
      }
    }

    if (!StringUtils.isBlank(gsh_input_vpnEndDate)) {
      try {
        vpnEndTime = GrouperUtil.stringToTimestamp(gsh_input_vpnEndDate);
        if (vpnEndTime.getTime() < System.currentTimeMillis()) {
          gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_vpnEndDate", "Error: VPN role end date/time must be in the future");
        }
      } catch (Exception e) {
        gsh_builtin_gshTemplateOutput.addValidationLine("gsh_input_vpnEndDate", "Error: VPN role end date/time must be 'yyyy/mm/dd' or 'yyyy/mm/dd
hh24:mi:ss'");
      }
    }
  }
  Long vpnStartTimeLong = vpnStartTime == null ? null : vpnStartTime.getTime();
  Long vpnEndTimeLong = vpnEndTime == null ? null : vpnEndTime.getTime();

  // Do not proceed is there is an error
  if (GrouperUtil.length(gsh_builtin_gshTemplateOutput.getValidationLines()) > 0) {
    gsh_builtin_gshTemplateOutput.assignIsError(true);
    GrouperUtil.gshReturn();
  }

  // dont navigate away from manage page for these actions
  if (actionAddUser || actionRemoveUser || actionTroubleShoot || actionAttest) {
    gsh_builtin_gshTemplateOutput.assignRedirectToGrouperOperation("NONE");
  }

  if (actionAddUser && roleIsVpnUsers) {
    grouperGroovyRuntime.setPercentDone(10);
    for (int i=0;i<subjects.length;i++) {

      boolean performedAction = false;

      if (otherVpnProjectSystemNamesForUser[i].size() > 0) {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "Cannot add user to VPN role: " + subjects[i].getDescription());
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "User: " + subjects[i].getName() + " already has access to other VPN(s) and cannot be in two
VPN(s) at once! " + StringUtils.join(otherVpnProjectSystemNamesForUser[i].iterator(), ", "));
        continue;
      }

      gsh_builtin_gshTemplateOutput.addOutputLine("Add user to VPN role: " + subjects[i].getDescription());
```

```
      // 12. Remove from Excludes if adding
      if (subjectIsExclude[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(excludesGroup).assignSubject(subjects[i]).assignSaveMode(SaveMode.
DELETE);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Removing: " + subjects[i].getName() + " from Excludes group: " + membershipSave.
getSaveResultType());
        subjectIsExclude[i] = false;
        performedAction = true;
      }

      // 13. Remove from Guests if employee
      if (subjectIsEmployee[i] && subjectIsGuest[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(guestsGroup).assignSubject(subjects[i]).assignSaveMode(SaveMode.
DELETE);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Removing: " + subjects[i].getName() + " from Guests group: " + membershipSave.
getSaveResultType());
        subjectIsAllow[i] = allowGroup.hasMember(subjects[i]);
        subjectIsGuest[i] = false;
        performedAction = true;
      }

      // 14. Add users to vpn role as employee, or edit if already a manual employee
      if ((!subjectIsAllow[i] && subjectIsEmployee[i]) || subjectIsEmployeesManual[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(employeesManualGroup).assignSubject(subjects[i]).
assignImmediateMshipEnabledTime(vpnStartTimeLong).assignImmediateMshipDisabledTime(vpnEndTimeLong);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Adding: " + subjects[i].getName() + " to Employee Manual group: " + membershipSave.
getSaveResultType());
        performedAction = true;
      }

      // 15. Add users to vpn role as guest
      if ((!subjectIsAllow[i] && !subjectIsEmployee[i]) || subjectIsGuest[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(guestsGroup).assignSubject(subjects[i]).
assignImmediateMshipEnabledTime(vpnStartTimeLong).assignImmediateMshipDisabledTime(vpnEndTimeLong);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Adding: " + subjects[i].getName() + " to Guest group: " + membershipSave.getSaveResultType());
        performedAction = true;
      }

      // 16. Lets see if things are ok with no enabled date
      subjectIsOverall[i] = overallGroup.hasMember(subjects[i]);
      if (vpnStartTime == null && !subjectIsOverall[i]) {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "Error: User still does not have access to VPN (maybe they are in a global lockout group): "
+ subjects[i].getName());
        performedAction = true;
      }

      // 17. Lets see if things are ok with no enabled date
      if (vpnStartTime != null && subjectIsOverall[i]) {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "Error: User is supposed to have a start date in the future but still have access to VPN
(maybe they are in a VPN ref group): " + subjects[i].getName());
        performedAction = true;
      }

      // 18. Do we need an output
      if (!performedAction && vpnStartTime == null && subjectIsOverall[i]) {
        gsh_builtin_gshTemplateOutput.addOutputLine("User: " + subjects[i].getName() + " already has access to the VPN");
        performedAction = true;
      }
      if (!performedAction && vpnStartTime != null) {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "User: " + subjects[i].getName() + " already has access to VPN and cannot have a future
start date (perhaps from a VPN ref group)");
      }
      grouperGroovyRuntime.setPercentDone(Math.max(10, (int)(((double)i)/subjects.length)));
    }

  }

  if (actionAddUser && roleIsAdmins) {
    for (int i=0;i<subjects.length;i++) {

      boolean performedAction = false;
```

```
      // 19. Add users to Admins manual
      if (!subjectIsAdmin[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(adminsManualGroup).assignSubject(subjects[i]);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Adding: (" + membershipSave.getSaveResultType() + ") to Admins Manual group: " + subjects[i].
getDescription());
        performedAction = true;
      }

      // 20. Lets see if things were already ok
      if (!performedAction) {
        gsh_builtin_gshTemplateOutput.addOutputLine("User was already in the Admins role: " + subjects[i].getDescription());
        performedAction = true;
      }
    }
  }

  if (actionAddUser && roleIsOwners) {
    for (int i=0;i<subjects.length;i++) {

      boolean performedAction = false;

      // 21. Add users to Owners manual
      if (!subjectIsOwner[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(ownersManualGroup).assignSubject(subjects[i]);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Adding: (" + membershipSave.getSaveResultType() + ") to Owners Manual group: " + subjects[i].
getDescription());
        performedAction = true;
      }

      // 22. Lets see if things are ok with no enabled date
      if (!performedAction) {
        gsh_builtin_gshTemplateOutput.addOutputLine("User was already in the Owners role: " + subjects[i].getDescription());
        performedAction = true;
      }
    }
  }


  if (actionRemoveUser && roleIsVpnUsers) {
    for (int i=0;i<subjects.length;i++) {

      boolean performedAction = false;

      // 23. Remove from Employees Manual
      if (subjectIsEmployeesManual[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(employeesManualGroup).assignSubject(subjects[i]).assignSaveMode
(SaveMode.DELETE);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Removing: (" + membershipSave.getSaveResultType() + ") from Employees Manual group: " +
subjects[i].getDescription());
        subjectIsOverall[i] = allowGroup.hasMember(subjects[i]);
        subjectIsExclude[i] = false;
        performedAction = true;
      }

      // 24. Remove from Guests
      if (subjectIsGuest[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(guestsGroup).assignSubject(subjects[i]).assignSaveMode(SaveMode.
DELETE);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Removing: (" + membershipSave.getSaveResultType() + ") from Guests Manual group: " + subjects
[i].getDescription());
        subjectIsOverall[i] = allowGroup.hasMember(subjects[i]);
        performedAction = true;
      }

      // 25. Add to excludes
      if (subjectIsOverall[i]) {
        MembershipSave membershipSave = new MembershipSave().assignGroup(excludesGroup).assignSubject(subjects[i]);
        membershipSave.save();
        gsh_builtin_gshTemplateOutput.addOutputLine("Adding: (" + membershipSave.getSaveResultType() + ") to Excludes group: " + subjects[i].
getDescription());
        performedAction = true;
      }
```

```
        // 26. Do we need an output
        if (!performedAction) {
          gsh_builtin_gshTemplateOutput.addOutputLine("User did not have access to the VPN: " + subjects[i].getDescription());
          performedAction = true;
        }
      }
    }

    if ((actionAddUser || actionRemoveUser) && roleIsVpnUsers) {
      //26.5 lets see how long it will take
      long changeLogTempCount = new GcDbAccess().sql("select count(1) from grouper_change_log_entry_temp").select(long.class);
      long currentAdClcPointer = new GcDbAccess().sql("select last_sequence_processed from grouper_change_log_consumer where name =
'pspng_activedirectoryFull'").select(long.class);
      long changeLogMaxPointer = new GcDbAccess().sql("select max(sequence_number) from grouper_change_log_entry").select(long.class);
      long recordsToGo = changeLogTempCount + (changeLogMaxPointer - currentAdClcPointer);
      String timeToProcess = null;
      if (recordsToGo < 2000) {
        timeToProcess = "should be granted in a few minutes";
      } else if (recordsToGo < 10000) {
        timeToProcess = "should be granted in 10-15 minutes";
      } else if (recordsToGo < 100000) {
        timeToProcess = "should be granted in an hour";
      } else {
        timeToProcess = "is delayed and could take a few hours";
      }
      gsh_builtin_gshTemplateOutput.addOutputLine("Note: there are " + recordsToGo + " items in queue which means access " + timeToProcess +
".  'Troubleshoot' the user to confirm access.");
    }

    if (actionRemoveUser && roleIsAdmins) {
      for (int i=0;i<subjects.length;i++) {

        boolean performedAction = false;

        // 27. Remove users from Admins manual
        if (subjectIsAdminManual[i]) {
          MembershipSave membershipSave = new MembershipSave().assignGroup(adminsManualGroup).assignSubject(subjects[i]).assignSaveMode
(SaveMode.DELETE);
          membershipSave.save();
          gsh_builtin_gshTemplateOutput.addOutputLine("Removing: (" + membershipSave.getSaveResultType() + ") from Admins Manual group: " + subjects
[i].getDescription());
          performedAction = true;
          subjectIsAdmin[i] = adminsGroup.hasMember(subjects[i]);
        }

        // 28. Lets see if things were already ok
        if (!subjectIsAdmin[i]) {
          gsh_builtin_gshTemplateOutput.addOutputLine("User: did not have access to Admin role: " + subjects[i].getDescription());
          performedAction = true;
        }

        // 29. Lets see if things were already ok
        if (!performedAction) {
          gsh_builtin_gshTemplateOutput.addOutputLine("error", "Cannot remove user from Admin role, they are probably in an Admin ref group: " + subjects
[i].getDescription());
          performedAction = true;
        }
      }
    }

    if (actionRemoveUser && roleIsOwners) {
      for (int i=0;i<subjects.length;i++) {

        boolean performedAction = false;

        // 30. Remove users from Admins manual
        if (subjectIsOwnerManual[i]) {
          MembershipSave membershipSave = new MembershipSave().assignGroup(ownersManualGroup).assignSubject(subjects[i]).assignSaveMode
(SaveMode.DELETE);
          membershipSave.save();
          gsh_builtin_gshTemplateOutput.addOutputLine("Removing: (" + membershipSave.getSaveResultType() + ") from Owners Manual group: " + subjects
[i].getDescription());
          performedAction = true;
          subjectIsAdmin[i] = ownersGroup.hasMember(subjects[i]);
        }
```

```
      // 31. Lets see if things were already ok
      if (!subjectIsOwner[i]) {
        gsh_builtin_gshTemplateOutput.addOutputLine("User did not have the Owner role: " + subjects[i].getDescription());
        performedAction = true;
      }

      // 32. Lets see if things were already ok
      if (!performedAction) {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "Cannot remove user from Owner role, they are probably in an Owner ref group: " + subjects
[i].getDescription());
        performedAction = true;
      }
    }
  }

  // 32.5 Run membership job if editing Admins or Owners
  if (roleIsAdmins || roleIsOwners) {
    GrouperLoader.runOnceByJobName(gsh_builtin_grouperSession, "OTHER_JOB_sraVpnMships", true);
  }

  if (actionTroubleShoot) {

    Group denyGroup = GroupFinder.findByName(gsh_builtin_grouperSession, projectStemName + ":" + gsh_input_projectSystemName + "SraVpnDeny",
true);

    List<Group> vpnRefGroups = new ArrayList<Group>();

    for (Member member : allowGroup.getImmediateMembers()) {

      // not a group we know about
      if (StringUtils.equals("g:gsa", member.getSubjectSourceId())
          && !StringUtils.equals(employeesManualGroup.getId(), member.getSubjectId())
          && !StringUtils.equals(guestsGroup.getId(), member.getSubjectId())) {

        vpnRefGroups.add(GroupFinder.findByUuid(gsh_builtin_grouperSession, member.getSubjectId(), true));

      }
    }

    List<Group> adminRefGroups = new ArrayList<Group>();

    for (Member member : allowGroup.getImmediateMembers()) {

      // not a group we know about
      if (StringUtils.equals("g:gsa", member.getSubjectSourceId())
          && !StringUtils.equals(adminsManualGroup.getId(), member.getSubjectId())) {

        adminRefGroups.add(GroupFinder.findByUuid(gsh_builtin_grouperSession, member.getSubjectId(), true));

      }
    }

    List<Group> ownerRefGroups = new ArrayList<Group>();

    for (Member member : allowGroup.getImmediateMembers()) {

      // not a group we know about
      if (StringUtils.equals("g:gsa", member.getSubjectSourceId())
          && !StringUtils.equals(ownersManualGroup.getId(), member.getSubjectId())) {

        ownerRefGroups.add(GroupFinder.findByUuid(gsh_builtin_grouperSession, member.getSubjectId(), true));

      }
    }


    for (int i=0;i<subjects.length;i++) {

      gsh_builtin_gshTemplateOutput.addOutputLine("info", "Troubleshooting user: " + subjects[i].getDescription());
```

```
      // 33. Check LDAP
      List<String> subjectIdsFromLdap = LdapSessionUtils.ldapSession().list(String.class, "pennKiteAd", "DC=kite,DC=upenn,DC=edu", LdapSearchScope.
SUBTREE_SCOPE,
        "(&(employeeID=" + subjects[i].getId() + ")(memberOf=CN=" + gsh_input_projectSystemName + "SraVpnOverall,OU=" +
gsh_input_projectSystemName
        + ",OU=service,OU=sraVpn,OU=service,OU=networking,OU=ts,OU=isc,OU=penn,OU=GrouperFull,OU=LocalAuth,DC=kite,DC=upenn,DC=edu))",
        "employeeId");

      if (GrouperUtil.length(subjectIdsFromLdap) > 0 && StringUtils.equals(subjectIdsFromLdap.get(0), subjects[i].getId())) {

        // 34. See if the user can use the VPN from LDAP
        gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is in LDAP and can use the VPN: " + subjects[i].getName());
        if (!subjectIsOverall[i]) {
          // 35. See if LDAP matches Grouper
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is not in the Overall group and is waiting for deprovisioning: " + subjects[i].
getName());
        }

      } else {

        // 36. See if the user cannot use the VPN from LDAP
        gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is not in LDAP and cannot use the VPN: " + subjects[i].getName());

        if (subjectIsOverall[i]) {
          // 37. See if LDAP matches Grouper
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is in the Overall group and is waiting for provisioning: " + subjects[i].getName());
        }

      }

      // 38. Last start date
      if (subjectIsOverall[i]) {
        Long lastStartTime = new GcDbAccess().sql("select max(gpmglv.the_start_time) from grouper_pit_mship_group_lw_v gpmglv where gpmglv.
subject_id = ? "
          + "and gpmglv.subject_source = 'pennperson' and gpmglv.field_name = 'members' and gpmglv.group_name = ? and the_start_time < ?")
          .addBindVar(subjects[i].getId())
          .addBindVar("penn:isc:ts:networking:service:sraVpn:service:" + gsh_input_projectSystemName + ":" + gsh_input_projectSystemName +
"SraVpnOverall")
          .addBindVar(System.currentTimeMillis() * 1000).select(Long.class);
        if (lastStartTime != null) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- Last start time for VPN: " + new java.sql.Timestamp(GrouperUtil.longValue(lastStartTime
/1000)));
        }
      }

      // 39. Next start date
      if (!subjectIsOverall[i]) {
        Long lastStartTime = new GcDbAccess().sql("select min(gpmglv.the_start_time) from grouper_pit_mship_group_lw_v gpmglv where gpmglv.
subject_id = ? "
          + " and gpmglv.subject_source = 'pennperson' and gpmglv.field_name = 'members' and gpmglv.group_name = ? and the_start_time > ?")
          .addBindVar(subjects[i].getId())
          .addBindVar("penn:isc:ts:networking:service:sraVpn:service:" + gsh_input_projectSystemName + ":" + gsh_input_projectSystemName +
"SraVpnOverall")
          .addBindVar(System.currentTimeMillis() * 1000).select(Long.class);
        if (lastStartTime != null) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- Next start time for VPN: " + new java.sql.Timestamp(GrouperUtil.longValue(lastStartTime
/1000)));
        }
      }

      // 40. Last end date
      if (!subjectIsOverall[i]) {
        Long lastEndTime = new GcDbAccess().sql("select max(gpmglv.the_end_time) from grouper_pit_mship_group_lw_v gpmglv where gpmglv.
subject_id = ? "
          + "and gpmglv.subject_source = 'pennperson' and gpmglv.field_name = 'members' and gpmglv.group_name = ? and the_end_time < ?")
          .addBindVar(subjects[i].getId())
          .addBindVar("penn:isc:ts:networking:service:sraVpn:service:" + gsh_input_projectSystemName + ":" + gsh_input_projectSystemName +
"SraVpnOverall")
          .addBindVar(System.currentTimeMillis() * 1000).select(Long.class);
        if (lastEndTime != null) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- Last end time for VPN: " + new java.sql.Timestamp(GrouperUtil.longValue(lastEndTime
/1000)));
        }
      }
```

```
        // 41. Next end date
        if (subjectIsOverall[i]) {
          Long lastStartTime = new GcDbAccess().sql("select min(gpmglv.the_end_time) from grouper_pit_mship_group_lw_v gpmglv where gpmglv.
subject_id = ? "
              + "and gpmglv.subject_source = 'pennperson' and gpmglv.field_name = 'members' and gpmglv.group_name = ? and the_end_time > ?")
            .addBindVar(subjects[i].getId())
            .addBindVar("penn:isc:ts:networking:service:sraVpn:service:" + gsh_input_projectSystemName + ":" + gsh_input_projectSystemName +
"SraVpnOverall")
            .addBindVar(System.currentTimeMillis() * 1000).select(Long.class);
          if (lastStartTime != null) {
            gsh_builtin_gshTemplateOutput.addOutputLine("info", "- Next end time for VPN: " + new java.sql.Timestamp(GrouperUtil.longValue(lastStartTime
/1000)));
          }
        }

        // 42. Employee group
        if (subjectIsEmployee[i]) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is an employee");

        } else {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is not an employee");

        }

        // 43. Deny group
        if (!subjectIsOverall[i] && denyGroup.hasMember(subjects[i])) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is in the deny group");
        }

        // 44. Allow group
        if (!subjectIsOverall[i] && subjectIsAllow[i]) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is in the allow group");
        }

        // 45. Which ref groups
        for (Group vpnRefGroup : vpnRefGroups) {
          if (vpnRefGroup.hasMember(subjects[i])) {
            gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is in VPN ref group: " + vpnRefGroup.getName());
          } else {
            gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is not in VPN ref group: " + vpnRefGroup.getName());
          }
        }

        // see if in multiple or other VPNs
        if (otherVpnProjectSystemNamesForUser[i].size() > 0) {
          gsh_builtin_gshTemplateOutput.addOutputLine("info", "- User is in other VPNs and cannot be in this one until removed from others: " + StringUtils.
join(otherVpnProjectSystemNamesForUser[i].iterator(), ", "));
        }

        //     ( SELECT to_timestamp((max(gpmglv.the_start_time) / 1000000)::double precision) AS to_timestamp
        //           FROM grouper_pit_mship_group_lw_v gpmglv
        //          WHERE gpmglv.member_id::text = gm.id::text AND gpmglv.the_active = 'T'::text AND gpmglv.field_name::text = 'members'::text AND gpmglv.
group_name::text ~~ 'penn:isc:ts:networking:service:sraVpn:service:%SraVpnOverall'::text AND gpmglv.group_name::text = ((((('penn:isc:ts:networking:
service:sraVpn:service:'::text || gs.extension::text) || ':'::text) || gs.extension::text) || 'SraVpnOverall'::text)) AS can_use_vpn_since,
        //     ( SELECT to_timestamp((min(gmav.immediate_mship_enabled_time) / 1000)::double precision) AS to_timestamp
        //           FROM grouper_memberships_all_v gmav,
        //            grouper_groups gg,
        //            grouper_fields gf
        //          WHERE gmav.owner_group_id::text = gg.id::text AND gf.name::text = 'readers'::text AND gf.id::text = gmav.field_id::text AND gmav.
member_id::text = gm.id::text AND gg.name::text ~~ 'penn:isc:ts:networking:service:sraVpn:service:%SraVpnOverall'::text AND gg.name::text = ((((('penn:
isc:ts:networking:service:sraVpn:service:'::text || gs.extension::text) || ':'::text) || gs.extension::text) || 'SraVpnOverall'::text) AND gmav.
immediate_mship_enabled_time::double precision > (1000::double precision * (date_part('epoch'::text, now()) - 10000::double precision))) AS
vpn_to_start_timestamp,
        //     ( SELECT to_timestamp((max(gmav.immediate_mship_disabled_time) / 1000)::double precision) AS to_timestamp
        //           FROM grouper_memberships_all_v gmav,
        //            grouper_groups gg,
        //            grouper_fields gf
        //          WHERE gmav.owner_group_id::text = gg.id::text AND gf.name::text = 'readers'::text AND gf.id::text = gmav.field_id::text AND gmav.
member_id::text = gm.id::text AND gmav.member_id::text = gm.id::text AND gg.name::text ~~ 'penn:isc:ts:networking:service:sraVpn:service:%
SraVpnOverall'::text AND gg.name::text = ((((('penn:isc:ts:networking:service:sraVpn:service:'::text || gs.extension::text) || ':'::text) || gs.extension::text) ||
'SraVpnOverall'::text) AND gmav.immediate_mship_disabled_time::double precision > (1000::double precision * (date_part('epoch'::text, now()) - 10000::
double precision))) AS vpn_to_end_timestamp
```

```
      // lets look at all groups
    }

     gsh_builtin_gshTemplateOutput.addOutputLine("info", "LDAP group DN: CN=" + gsh_input_projectSystemName + "SraVpnOverall,OU=" +
gsh_input_projectSystemName + ",OU=service,OU=sraVpn,OU=service,OU=networking,OU=ts,OU=isc,OU=penn,OU=GrouperFull,OU=LocalAuth,
DC=kite,DC=upenn,DC=edu");

  }

  if (actionRunReportNow) {

    java.sql.Timestamp now = new java.sql.Timestamp(System.currentTimeMillis());

    grouperGroovyRuntime.setPercentDone(10);

    GrouperUtil.sleep(3000);

    grouperGroovyRuntime.setPercentDone(20);

    // 46. kick off the dependent jobs on daemon
    GrouperLoader.runOnceByJobName(gsh_builtin_grouperSession, "OTHER_JOB_sraVpnMships", true);
    GrouperLoader.runOnceByJobName(gsh_builtin_grouperSession, "OTHER_JOB_sraVpnTimes", true);
    GrouperLoader.runOnceByJobName(gsh_builtin_grouperSession, "OTHER_JOB_sraVpnPit", true);

    boolean mshipJobGood = false;
    boolean timesJobGood = false;
    boolean pitJobGood = false;

    // 47. wait for jobs to run on daemon
    for (int i=0;i<12;i++) {
      GrouperUtil.sleep(5000);
      mshipJobGood = mshipJobGood || 1 <= new GcDbAccess().sql("select count(1) from grouper_loader_log where job_name =
'OTHER_JOB_sraVpnMships' and status = 'SUCCESS' and started_time > ?").addBindVar(now).select(int.class);
      timesJobGood = timesJobGood || 1 <= new GcDbAccess().sql("select count(1) from grouper_loader_log where job_name =
'OTHER_JOB_sraVpnTimes' and status = 'SUCCESS' and started_time > ?").addBindVar(now).select(int.class);
      pitJobGood = timesJobGood || 1 <= new GcDbAccess().sql("select count(1) from grouper_loader_log where job_name = 'OTHER_JOB_sraVpnPit'
and status = 'SUCCESS' and started_time > ?").addBindVar(now).select(int.class);
      if (timesJobGood && mshipJobGood && pitJobGood) {
        break;
      }
    }

    grouperGroovyRuntime.setPercentDone(60);

    String reportConfigName = gsh_input_projectSystemName + "SraVpnReport";
    GrouperReportConfigurationBean reportConfigBean = GrouperReportConfigService.getGrouperReportConfigBean(projectFolder, reportConfigName);

    String reportJobName = "grouper_report_" + projectFolder.getId() + "_" + reportConfigBean.getAttributeAssignmentMarkerId();

    now = new java.sql.Timestamp(System.currentTimeMillis());
    GrouperUtil.sleep(3000);

    grouperGroovyRuntime.setPercentDone(70);

    // 48. run report
    GrouperLoader.runOnceByJobName(gsh_builtin_grouperSession, reportJobName, true);
    boolean reportJobGood = false;

    // 49. wait for report
    for (int i=0;i<12;i++) {
      GrouperUtil.sleep(5000);
      reportJobGood = reportJobGood || (1 <= new GcDbAccess().sql("select count(1) from grouper_loader_log where job_name = '" + reportJobName + "'
and status = 'SUCCESS' and started_time > ?").addBindVar(now).select(int.class));
      if (reportJobGood) {
        break;
      }
    }
```

```
    if (mshipJobGood && timesJobGood && reportJobGood) {
      gsh_builtin_gshTemplateOutput.addOutputLine("success", "A new report (including dependent jobs) was generated successfully.");
    } else {
      if (System.currentTimeMillis() - projectFolder.getCreateTimeLong() < 1000*60*60) {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "This folder is less than an hour old, please try again in 15-60 minutes");
      } else {
        gsh_builtin_gshTemplateOutput.addOutputLine("error", "There was a problem generating a report, please contact help@isc.upenn.edu");
        gsh_builtin_gshTemplateOutput.assignIsError(true);
      }
      gsh_builtin_gshTemplateOutput.assignRedirectToGrouperOperation("NONE");
      GrouperUtil.gshReturn();
    }
    grouperGroovyRuntime.setPercentDone(90);

  }

  if (actionViewReports || actionRunReportNow) {

    // 50. redirect to reports list
    String reportConfigName = gsh_input_projectSystemName + "SraVpnReport";
    GrouperReportConfigurationBean reportConfigBean = GrouperReportConfigService.getGrouperReportConfigBean(projectFolder, reportConfigName);

    if (reportConfigBean == null) {
      gsh_builtin_gshTemplateOutput.addOutputLine("error", "Reports are not configured for this VPN, contact help@isc.upenn.edu for help");
    } else {
      gsh_builtin_gshTemplateOutput.addOutputLine("success", "Please wait while reports are retrieved");
      gsh_builtin_gshTemplateOutput.assignRedirectToGrouperOperation("operation=UiV2GrouperReport.
viewAllReportInstancesForFolder&attributeAssignmentMarkerId=" + reportConfigBean.getAttributeAssignmentMarkerId()  + "&stemId=" + projectFolder.
getId());
    }
  }

  if (actionVisualize) {

    // 51. redirect to visualization
    gsh_builtin_gshTemplateOutput.assignRedirectToGrouperOperation("operation=UiV2Visualization.stemView&stemId=" + projectFolder.getId());
  }

  if (actionAttest) {

    // 52. mark report as attestated
    AttestationStemSave attestationStemSave = new AttestationStemSave().assignStem(projectFolder).assignReplaceAllSettings(false).
assignMarkAsAttested(true);
    attestationStemSave.save();
    gsh_builtin_gshTemplateOutput.addOutputLine("Attested: VPN access: " + attestationStemSave.getSaveResultType());
  }
```

Configuration

Search

Logged in as Chris Hyzer · Log out · Help

Home > Miscellaneous > GSH templates > Edit GSH template

GSH templates

Actions

Config id		sraVpnManage
Enabled	☐ EL?	◉ Default value (True) ○ True ○ False if this template is enabled. Default value is 'true'.
Show on folders	☐ EL?	○ Default value (False) ◉ True ○ False if this template option is available on folders. Default value is 'false'.
Folder show type	☐ EL?	Certain folder * Where should this template be available
Folder uuid to show	☐ EL?	6c82886503674a108d3fc7f5ec5d855f * UUID of folder to show this template
Folder show on descendants	☐ EL?	Certain folder and all descendants * certainFolder: just show on one folder. oneChildLevel: only show on child level under folder. certainFolderAndOneChildLevel: show the folder and one level of children. descendants: show all folders under the folder. certainFolderAndDescendants: show folder and all descendants.
Security run type	☐ EL?	Specified group * Who can run this template. Only GrouperSystem / wheel group, or specify a group
Group uuid	☐ EL?	2d4bd7bd843b4828b7fc8bf7fe323527 * UUID of group that can run this template
Run as type	☐ EL?	GrouperSystem * Select the type of user to run as. "currentUser" means run as the user using the UI. "GrouperSystem" means run as root user, "specifiedSubject" means you can pick a subject to run as (not common).
Group can WS act as	☐ EL?	 Group UUID where members of the group, when they execute a template via WS can act as other users
Externalized text?	☐ EL?	◉ Default value (False) ○ True ○ False Selec 'True' if you would like to use externalized text for template name, description, and input labels & descriptions. Default value is 'false'.
Template name	☐ EL?	SRA VPN manage * Template name
Template description	☐ EL?	Manage an SRA VPN. This is the central point of contact to add/remove users, review access, etc. * Template description
Show in more actions	☐ EL?	○ Default value (False) ◉ True ○ False If you want this template to show under folders and groups 'More actions' button, select 'True'. Default value is 'false'.
Template label in more actions	☐ EL?	SRA VPN manage * Label you want for this template to show under folders and groups 'More actions' button.
Display error messages and output on error	☐ EL?	○ Default value (False) ◉ True ○ False If you want to see error messages and output when you run gsh template and experience problems, Select 'True'. Default value is 'false'.

Lightweight GSH	☐ EL?	◉ Default value (False) ○ True ○ False If you are managing your own imports and don't want built-ins select true. This reduces the GSH execution time by a few seconds. Default value is 'false'.
Run gsh script in transaction	☐ EL?	○ Default value (True) ○ True ◉ False Run gsh script in a database transaction so that if any statement fails, then all the database changes are rolled back. Default value is 'true'.
Use individual audits	☐ EL?	◉ Default value (True) ○ True ○ False If each Grouper action that is run should have an individual audit record. If false, just have an audit record that the user ran the template in general. Default value is 'true'.
Consolidate output	☐ EL?	◉ Default value (True) ○ True ○ False When printing multiple output statements to the UI, should the successes be merged into one green block (and errors, and info into their blocks). Default value is 'true'.
GSH script	☐ EL?	`boolean roleIsVpnUsers = StringUtils.equals(gsh_input_role, "vpn");` `boolean roleIsAdmins = StringUtils.equals(gsh_input_role,` * This is the GSH template to run
Number of inputs	☐ EL?	6 Number of inputs (form elements on ui or in ws). Default value is '0'.

Input gsh_input_projectSystemName

Input item configuration and validation. An input item is provided by the user using the UI or WS.

Input gsh_input_projectSystemName: name	☐ EL?	gsh_input_projectSystemName * Name of the ui form element, ws param, template variable. Must start with gsh_input_
Input gsh_input_projectSystemName: label	☐ EL?	VPN name * Label on the UI for this input
Input gsh_input_projectSystemName: description	☐ EL?	Name of the VPN to manage * Description on the UI for this input
Input gsh_input_projectSystemName: input type	☐ EL?	 Type of the data. Default value is 'string'.
Input gsh_input_projectSystemName: form element type	☐ EL?	Dropdown Form element type on UI. Default value is 'text'.
Input gsh_input_projectSystemName: dropdown value format	☐ EL?	SQL Dropdown value format. Default value is 'csv'.
Input gsh_input_projectSystemName: dropdown database	☐ EL?	grouper * Database that this SQL runs against
Input gsh_input_projectSystemName: dropdown SQL	☐ EL?	`select distinct stem_extension from penn_sra_admins_v` `where penn_id = $$subjectId$$ order by 1` * Enter a SQL that has two columns (ordered): first column is the key, second column is the label. Can use $$subjectId$$ once in the SQL and will be substituted by a bind variable for the user using the application subjectId.
Input gsh_input_projectSystemName: dropdown SQL cache minutes	☐ EL?	 Number of minutes to cache this SQL call result. Default value is '2'.
Input gsh_input_projectSystemName: form element index	☐ EL?	 Form elements will show on the form in order of "index". DefaultValue is 0. If two elements have the same index then they will show in order of configuration. Default value is '0'.
Input gsh_input_projectSystemName: required	☐ EL?	○ Default value (False) ◉ True ○ False If this input is required for template to run. Default value is 'false'.
Input gsh_input_projectSystemName: Jexl for showEl	☐ EL?	 Jexl for if this field should show, note all inputs are available to use as variables

Input gsh_input_action

Input item configuration and validation. An input item is provided by the user using the UI or WS.

Input gsh_input_action: name	☐ EL?	gsh_input_action * Name of the ui form element, ws param, template variable. Must start with gsh_input_
Input gsh_input_action: label	☐ EL?	I want to * Label on the UI for this input
Input gsh_input_action: description	☐ EL?	What action do you want to perform on the VPN * Description on the UI for this input
Input gsh_input_action: input type	☐ EL?	 Type of the data. Default value is 'string'.
Input gsh_input_action: form element type	☐ EL?	Dropdown Form element type on UI. Default value is 'text'.
Input gsh_input_action: dropdown value format	☐ EL?	Json array of objects with "key" and "label" fields Dropdown value format. Default value is 'csv'.
Input gsh_input_action: json values	☐ EL?	[{"key": "addUser", "label": "Add user(s)"}, {"key": "removeUser", "label": "Remove user(s)"}, {"key": "troubleshoot", "label": "Troubleshoot user(s)"}, * Json values e.g [{"key": "1234", "label": "Business school"},{"key": "2345", "label": "Engineering school"}]
Input gsh_input_action: form element index	☐ EL?	 Form elements will show on the form in order of "index". DefaultValue is 0. If two elements have the same index then they will show in order of configuration. Default value is '0'.
Input gsh_input_action: required	☐ EL?	◯ Default value (False) ◉ True ◯ False If this input is required for template to run. Default value is 'false'.
Input gsh_input_action: Jexl for showEl	☐ EL?	${gsh_input_projectSystemName != null} Jexl for if this field should show, note all inputs are available to use as variables

Input gsh_input_pennkeys

Input item configuration and validation. An input item is provided by the user using the UI or WS.

Input gsh_input_pennkeys: name	☐ EL?	gsh_input_pennkeys * Name of the ui form element, ws param, template variable. Must start with gsh_input_
Input gsh_input_pennkeys: label	☐ EL?	PennKeys * Label on the UI for this input
Input gsh_input_pennkeys: description	☐ EL?	Comma separated PennKeys or Pennids to add or remove * Description on the UI for this input
Input gsh_input_pennkeys: input type	☐ EL?	 Type of the data. Default value is 'string'.
Input gsh_input_pennkeys: form element type	☐ EL?	 Form element type on UI. Default value is 'text'.
Input gsh_input_pennkeys: max length	☐ EL?	4000 Max length. Default value is '500'.
Input gsh_input_pennkeys: form element index	☐ EL?	 Form elements will show on the form in order of "index". DefaultValue is 0. If two elements have the same index then they will show in order of configuration. Default value is '0'.
Input gsh_input_pennkeys: validation type	☐ EL?	Regex * Type of validation on the input

Type of validation on the input

Input gsh_input_pennkeys: validation regex	☐ EL?	^[a-z0-9,]+$ * Regex to check the input and if it doesnt match then fail. For example, this is alphanumeric or underscore length between 1 and 50
Input gsh_input_pennkeys: validation message	☐ EL?	Enter lower case comma separated PennKeys or PennIds Validation message that should be shown for builtin validations
Input gsh_input_pennkeys: required	☐ EL?	○ Default value (False) ◉ True ○ False If this input is required for template to run. Default value is 'false'.
Input gsh_input_pennkeys: Jexl for showEl	☐ EL?	${gsh_input_action == 'troubleshoot' || gsh_input_action == 'addUse Jexl for if this field should show, note all inputs are available to use as variables
Input gsh_input_pennkeys: trim whitespace	☐ EL?	◉ Default value (True) ○ True ○ False If whitespace should be trimmed from the value for this input. Default value is 'true'.

Input gsh_input_role

Input item configuration and validation. An input item is provided by the user using the UI or WS.

Input gsh_input_role: name	☐ EL?	gsh_input_role * Name of the ui form element, ws param, template variable. Must start with gsh_input_
Input gsh_input_role: label	☐ EL?	Role * Label on the UI for this input
Input gsh_input_role: description	☐ EL?	VPN users can connect to the VPN. Admins can manage the VPN ι * Description on the UI for this input
Input gsh_input_role: input type	☐ EL?	⌄ Type of the data. Default value is 'string'.
Input gsh_input_role: form element type	☐ EL?	Dropdown ⌄ Form element type on UI. Default value is 'text'.
Input gsh_input_role: dropdown value format	☐ EL?	Json array of objects with "key" and "label" fields ⌄ Dropdown value format. Default value is 'csv'.
Input gsh_input_role: json values	☐ EL?	[{"key": "vpn", "label": "VPN users"},{"key": "admin", "label": "Admins"},{"key": "owner", "label": "Owners"}] * Json values e.g [{"key": "1234", "label": "Business school"},{"key": "2345", "label": "Engineering school"}]
Input gsh_input_role: form element index	☐ EL?	 Form elements will show on the form in order of "index". DefaultValue is 0. If two elements have the same index then they will show in order of configuration. Default value is '0'.
Input gsh_input_role: required	☐ EL?	○ Default value (False) ◉ True ○ False If this input is required for template to run. Default value is 'false'.
Input gsh_input_role: Jexl for showEl	☐ EL?	${gsh_input_action == 'addUser' || gsh_input_action == 'removeUse Jexl for if this field should show, note all inputs are available to use as variables

Input gsh_input_vpnStartDate

Input item configuration and validation. An input item is provided by the user using the UI or WS.

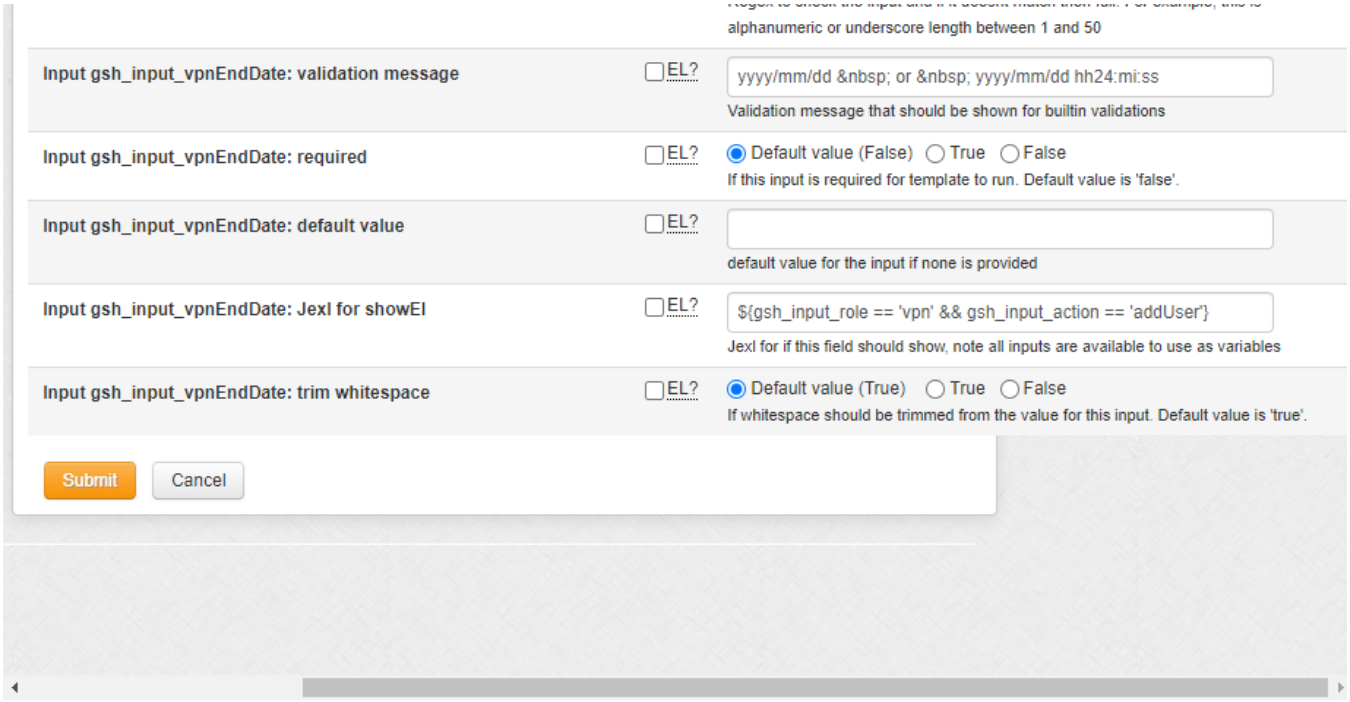
Input gsh_input_vpnStartDate: name	☐ EL?	gsh_input_vpnStartDate * Name of the ui form element, ws param, template variable. Must start with gsh_input_
Input gsh_input_vpnStartDate: label	☐ EL?	VPN role start date/time * Label on the UI for this input
Input gsh_input_vpnStartDate: description	☐ EL?	If the VPN access should be future dated, put the start date in the fu * Description on the UI for this input

Input gsh_input_vpnStartDate: input type	☐ EL?	Type of the data. Default value is 'string'.
Input gsh_input_vpnStartDate: form element type	☐ EL?	Form element type on UI. Default value is 'text'.
Input gsh_input_vpnStartDate: max length	☐ EL?	Max length. Default value is '500'.
Input gsh_input_vpnStartDate: form element index	☐ EL?	Form elements will show on the form in order of "index". DefaultValue is 0. If two elements have the same index then they will show in order of configuration. Default value is '0'.
Input gsh_input_vpnStartDate: validation type	☐ EL?	Regex * Type of validation on the input
Input gsh_input_vpnStartDate: validation regex	☐ EL?	^[0-9V :]{10,20}$ * Regex to check the input and if it doesnt match then fail. For example, this is alphanumeric or underscore length between 1 and 50
Input gsh_input_vpnStartDate: validation message	☐ EL?	yyyy/mm/dd or yyyy/mm/dd hh24:mi Validation message that should be shown for builtin validations
Input gsh_input_vpnStartDate: required	☐ EL?	◉ Default value (False) ○ True ○ False If this input is required for template to run. Default value is 'false'.
Input gsh_input_vpnStartDate: default value	☐ EL?	default value for the input if none is provided
Input gsh_input_vpnStartDate: Jexl for showEl	☐ EL?	${gsh_input_role == 'vpn' && gsh_input_action == 'addUser'} Jexl for if this field should show, note all inputs are available to use as variables
Input gsh_input_vpnStartDate: trim whitespace	☐ EL?	◉ Default value (True) ○ True ○ False If whitespace should be trimmed from the value for this input. Default value is 'true'.

Input gsh_input_vpnEndDate

Input item configuration and validation. An input item is provided by the user using the UI or WS.

Input gsh_input_vpnEndDate: name	☐ EL?	gsh_input_vpnEndDate * Name of the ui form element, ws param, template variable. Must start with gsh_input_
Input gsh_input_vpnEndDate: label	☐ EL?	VPN role end date/time * Label on the UI for this input
Input gsh_input_vpnEndDate: description	☐ EL?	If the VPN access should be expire on a certain date, put the end da * Description on the UI for this input
Input gsh_input_vpnEndDate: input type	☐ EL?	Type of the data. Default value is 'string'.
Input gsh_input_vpnEndDate: form element type	☐ EL?	Form element type on UI. Default value is 'text'.
Input gsh_input_vpnEndDate: max length	☐ EL?	Max length. Default value is '500'.
Input gsh_input_vpnEndDate: form element index	☐ EL?	Form elements will show on the form in order of "index". DefaultValue is 0. If two elements have the same index then they will show in order of configuration. Default value is '0'.
Input gsh_input_vpnEndDate: validation type	☐ EL?	Regex * Type of validation on the input
Input gsh_input_vpnEndDate: validation regex	☐ EL?	^[0-9V :]{10,20}$ * Regex to check the input and if it doesnt match then fail. For example, this is


alphanumeric or underscore length between 1 and 50
Input gsh_input_vpnEndDate: validation message
EL?
yyyy/mm/dd   or   yyyy/mm/dd hh24:mi:ss
Validation message that should be shown for builtin validations
Input gsh_input_vpnEndDate: required
EL?
Default value (False)
True
False
If this input is required for template to run. Default value is 'false'.
Input gsh_input_vpnEndDate: default value
EL?
default value for the input if none is provided
Input gsh_input_vpnEndDate: Jexl for showEl
EL?
${gsh_input_role == 'vpn' && gsh_input_action == 'addUser'}
Jexl for if this field should show, note all inputs are available to use as variables
Input gsh_input_vpnEndDate: trim whitespace
EL?
Default value (True)
True
False
If whitespace should be trimmed from the value for this input. Default value is 'true'.
Submit
Cancel