Produzione - IRIS AP-RM Definizione Logiche condivise di flusso (start, action, validation)

**wfStartLogicInferLinkedProjectTypeFromSiblingCalls**
Questa logica viene eseguita in fase di creazione di un nuovo bando se attiva la funzione di creazione del progetto a partire dalla selezione del bando. La logica è abilitata solo se

- la variabile di configurazione `ap.prj.create-by-call.enabled` è valorizzata a `true`

Questa logica tenta di recuperare da un bando già presente della stessa tipologia tipo di progetto da associare al bando

**Configurazioni della StartLogic:**
- `ap.prj.create-by-call.enabled`

```javascript
var projectCreationByCallEnabled = ConfigurationUtil.getConfigValue("ap.prj.create-by-call.enabled");
if (projectCreationByCallEnabled=="true"){
  var paramMap = new Packages.java.util.HashMap();
  paramMap.put("wfItemTypeId", wfItem.getWfItemTypeId());
  paramMap.put("excludedWfItemId", wfItem.getId());
  paramMap.put("withdrawn", "0");
  var callWithSameTypeList = searchBuilderTemplate.query("getWfItemByWfItemTypeId", paramMap);
  if (CollectionUtils.isNotEmpty(callWithSameTypeList)){
    var callWithSameType = callWithSameTypeList.get(0);
    var linkedProjectType=callWithSameType.getStringMap().get("linkedProjectType");
    if (linkedProjectType!=null)
      wfItem.getStringMap().put("linkedProjectType", linkedProjectType);
  }
}
```

**wfActionLogicHandleLinkedProjectType**
Questa logica di sistema gestisce il metadato `linkedProjectType`. Per maggiori dettagli fare riferimento al modello dati dell'entità Bando

```javascript
var linkedProjectTypeInternal = wfItem.getStringMap().get("linkedProjectTypeInternal");
if (linkedProjectTypeInternal != null){
  var linkedProjectType = wfService.getWfItemType(new Integer(linkedProjectTypeInternal));
  if (linkedProjectType!=null)
    wfItem.getStringMap().put("linkedProjectType", linkedProjectType.getIdentifier());
}
```

**isCreateProjectByCallEnabled**
Questa è un'applicability rule che viene valutata prima di eseguire le validazioni cui è agganciata. Ritorna true se la variabile di conf `ap.prj.create-by-call.enabled` è valorizzata a true altrimenti no.

**Configurazioni della ValidateLogic:**
- `ap.prj.create-by-call.enabled`

```javascript
var createProjectByCallEnabled=Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.create-by-call.enabled");
if ("true"==createProjectByCallEnabled)
```
linkedProjectTypeValidator
Questa validazione verifica che il campo "Tipo progetto associato" sia valorizzato
Dal punto di vista del modello dati si tratta dell'attributo stringMap[linkedProjectType]
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

```java
var linkedProjectType=wfItem.getStringMap().get("linkedProjectType");
if (StringUtils.isEmpty(linkedProjectType)) {
    var linkedProjectTypeInternal=wfItem.getStringMap().get("linkedProjectTypeInternal");
    if (StringUtils.isEmpty(linkedProjectTypeInternal))
        errors.rejectValue("stringMap[linkedProjectTypeInternal]","error.call.
linkedProjectType.required");
}
```

checkOpenEndedAndEndDateValidator
Questa validazione controlla che se il bando di finanziamento viene marcato come "con scadenza" allora deve essere specificata la data di scadenza
Per maggiori dettagli cfr. excel modello dati dell'entità Bando di finanziamento

```java
if(object.getBooleanMap().get("openEnded")!=null){
    if((object.getBooleanMap().get("openEnded")===false)&&(object.getDateMap().get("endDate")===null))
        errors.rejectValue("booleanMap[openEnded]","error.call.
openEndedAndEndDateNotCompatible");
    errors.rejectValue("dateMap[endDate]","error.call.
openEndedAndEndDateNotCompatible");
}
```

grantorValidatorCall
Questa validazione controlla che esista almeno un ente finanziatore nel tab "Dati generali".
Dal punto di vista del modello dati si tratta degli elementi di tipo grantor

```java
var grantorSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object,
    "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "grantor", wfService);
if (grantorSet.size()<1){
    errors.rejectAndLocate("error.project.grantor.atLeast1",
        "grantor");
}
```

uniqueCodeValidator
Questa validazione controlla l'unicità del campo Codice bando.

```java
if(object.getStringMap().get("code") != null){
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("code", object.getStringMap().get("code");
    paramMap.put("wfItemIdNot", object.getId());
    var count=searchBuilderTemplate.getSingleObject("getAllCallByCode", paramMap);
    if (count>=1)
        errors.rejectValue("stringMap[code]", "error.call.code.notUnique");
```
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
In base alla visione in cui è stato creato l’oggetto, viene effettuato un setup diverso:
- Visione completa:
  - Viene estratto il dipartimento dal Direttore del Corso e viene inserito tra i Dipartimenti con ruolo di Organizzatore
  - Viene estratto l’Ateneo dalla configurazione OBBLIGATORIA rm.orgunit.external.myOrganization e viene inserito tra i Soggetti terzi, con ruolo di Coordinatore

Visione dipartimentale:
- Viene estratto il dipartimento dal Direttore del Corso e viene inserito tra i Dipartimenti, con ruolo di Coordinatore

Per maggiori dettagli cfr. modello dati dell’entità continuousTraining

```java
var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","owner", wfService);
if(!ownerWfElementSet.isEmpty()){
    var ownerWfElement = ownerWfElementSet.iterator().next();
    var owner = ownerWfElement.getPersonMap().get("ownerId");
    if(owner != null){
        var checkDate = WfUtil.getCheckDate(object, "owner", wfService);
        var positionContext = WfUtil.getPositionContext(object, positionContext, "department", gaService);
        var ownerPositionSet = WfUtil.getPositionSet(owner, checkDate, positionContext, "department", gaService);
        var department = GaUtil.getPriorityOrganizationUnit(ownerPositionSet, positionContext, "department");
        wfItem.getNumberMap().put("receivedFinancing", Packages.java.math.BigDecimal.valueOf(0));
        wfItem.getNumberMap().put("totalSubscriptionFees", Packages.java.math.BigDecimal.valueOf(0));
        wfItem.getNumberMap().put("otherFinancing", Packages.java.math.BigDecimal.valueOf(0));
        wfItem.getNumberMap().put("totalIncome", Packages.java.math.BigDecimal.valueOf(0));
        wfItem.getNumberMap().put("percentageOfEUFinacing", Packages.java.math.BigDecimal.valueOf(0));
        wfItem.getNumberMap().put("percentageOfNationalFinacing", Packages.java.math.BigDecimal.valueOf(0));
        //in teoria inutile, ma comportamento strano su cint
        wfItem.getIntegerMap().put("numberOfInternalTeachers", Packages.java.lang.Integer.valueOf(1));
        var coordinatorDictionaryId = WfUtil.getDictionaryByCode(wfService, "ouRoleContinuousTraining.coordinator");
        var coordinatorDictionary = wfService.getWfDictionary(coordinatorDictionaryId);
        //syso.println("coordinatorDictionaryId: "+coordinatorDictionaryId);
        //syso.println("coordinatorDictionary: "+coordinatorDictionary);
        if(Packages.java.lang.Boolean.FALSE.equals(GaUtil.isCurrentUserInDepartmentView())){
            //syso.println("visione completa");
            var myOrganizationIdString = ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
            if(myOrganizationIdString != null){
                //syso.println("myOrganizationIdString: "+ myOrganizationIdString);
                //syso.println("wfItem.getId(): "+ wfItem.getId());
            }
        }
    }
}
```
getId();

Integer.parseInt(myOrganizationIdString);

myOrganizationId = Packages.java.lang.

var myOrganizationId = Packages.java.lang.

myOrganizationId);

var myOrganization = gaService.

getOrganizationUnit(myOrganizationId);

var externalOrganizationUnitElement = new

Packages.it.cilea.wf.model.WfItemElement();

("externalOrganization");

(wfItem.getId());

getOrganizationUnitMap().put("ouId", myOrganization);

getWFDictionaryMap().put("roleId", coordinatorDictionary);

(externalOrganizationUnitElement);

(externalOrganizationUnitElement);

getDictionaryByCode(wfService, "ouRoleContinuousTraining.organizator");

getWFDictionary(organizatorDictionaryId);

("organizatorDictionaryId: "+ organizatorDictionaryId);

"+ organizatorDictionary);

internalOrganizationUnitWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.
cilea.wf.model.WfItemElement","internalOrganizationUnit", wfService);

var internalOrganizationUnitWfElement =

internalOrganizationUnitWfElementSet.iterator().next();

getWFDictionaryMap().put("roleId", organizatorDictionary);

{internalOrganizationUnitWfElement};

wfService.saveOrUpdate

wfService.saveOrUpdate(wfItem);

if(department != null){

var organizatorDictionaryId = WFUtil.

getDictionaryByCode(wfService, "ouRoleContinuousTraining.organizator");

getWFDictionary(organizatorDictionaryId);

("organizatorDictionaryId: "+ organizatorDictionaryId);

"+ organizatorDictionary);

var organizationalDictionaryId = WFUtil.

getDictionaryByCode(wfService, "ouRoleContinuousTraining.organizator");

getWFDictionary(organizatorDictionaryId);

("organizatorDictionaryId: "+ organizatorDictionaryId);

"+ organizatorDictionary);

internalOrganizationUnitWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.
cilea.wf.model.WfItemElement","internalOrganizationUnit", wfService);

var internalOrganizationUnitWfElement =

internalOrganizationUnitWfElementSet.iterator().next();

getWFDictionaryMap().put("roleId", organizationalDictionary);

{internalOrganizationUnitWfElement};

wfService.saveOrUpdate

wfService.saveOrUpdate(wfItem);

} else{

throw "Per creare un nuovo corso di Formazione

Continua in visione completa è necessario che sia impostata correttamente la configuration rm.orgunit.external.

myOrganization.Contattare Helpdesk";

} /*

} else{

//sys.printin("visione dipartimentale");

if(department != null){

var internalOrganizationUnitWfElementSet =

FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","

internalOrganizationUnit", wfService);

var internalOrganizationUnitWfElement =

internalOrganizationUnitWfElementSet.iterator().next();
isTypeFCO
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni. Se ritorna true allora vengono eseguite le altre validazioni altrimenti no. Questa regola considera la tipologia (WfItemType) e attiva la valutazione solo per la Formazione Continua generica, non quindi quella di Medicina. Per tutti gli oggetti che hanno questo attributo valorizzato a true vengono disattivate le validazioni e le eventuali sincronizzazioni.

configurationPresentValidatorFCO
Questa validazione controlla che sia valorizzata la configurazione rm.orgunit.external.myOrganization sia valorizzata. Nel caso in cui non lo sia, impedisce la creazione del Corso di Formazione Continua. Essa deve essere valorizzata con l'ID Iris della Organizzazione Esterna che rappresenta l’Ateneo.

continuousTrainingOrganizationUnitRoleValidator
Questa validazione controlla che ogni Dipartimento (internalOrganizationUnit) abbia un ruolo associato solo un Dipartimento o Soggetto terzo abbia il ruolo di Coordinatore. Per maggiori dettagli cfr. modello dati dell'entità in questione.
if(internalOuRole != null){
    internalOuRole = wfService.getWfDictionary(internalOuRole.getId());
    var internalOuRoleCode = internalOuRole.getStringMap().get("code");
    if(internalOuRoleCode.compareTo("ouRoleContinuousTraining.coordinator") == 0){
        coordinatorCount++;
    }else{
        isInternalOuRoleMissing = true;
    }
}
if(isInternalOuRoleMissing)
    errors.rejectAndLocate("error.continuousTraining.continuousTrainingOrganizationUnitRoleValidator.internalOrganizationUnit.role.missing", "internalOrganizationUnit");

var externalOrganizationWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","externalOrganization", wfService);
if(externalOrganizationWfElementSet.isEmpty() == false){
    var externalOuIterator = externalOrganizationWfElementSet.iterator();
    while( externalOuIterator.hasNext() ){
        var externalOuWfItemElement = externalOuIterator.next();
        var externalOuRole = externalOuWfItemElement.getWfDictionaryMap().get("roleId");
        if(externalOuRole != null){
            externalOuRole = wfService.getWfDictionary(externalOuRole.getId());
            var externalOuRoleCode = externalOuRole.getStringMap().get("code");
            if(externalOuRoleCode.compareTo("ouRoleContinuousTraining.coordinator") == 0){
                coordinatorCount++;
            }
        }
    }
}
if(coordinatorCount > 1){
    errors.rejectAndLocate("error.continuousTraining.continuousTrainingOrganizationUnitRoleValidator.role.coordinator.tooMany", "internalOrganizationUnit");
    errors.rejectAndLocate("error.continuousTraining.continuousTrainingOrganizationUnitRoleValidator.role.coordinator.tooMany", "externalOrganization");
}
**continuousTrainingFinancingValidator**
Questa validazione controlla che la somma dei vari finanziamenti inserita(calcolata) sia quella effettivamente corretta.
I dati che sono presi in considerazione sono:
- Eventuale importo percepito per la convenzione (receivedFinancing)
- Eventuali quote totali di iscrizione al corso (totalSubscriptionFees)
- Eventuali altre entrate (otherFinancing)
- Introiti complessivi (totalIncome)
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```javascript
var receivedFinancing = object.getNumberMap().get("receivedFinancing");
var totalSubscriptionFees = object.getNumberMap().get("totalSubscriptionFees");
var otherFinancing = object.getNumberMap().get("otherFinancing");
if(receivedFinancing == null && totalSubscriptionFees == null && otherFinancing == null)
{
    errors.rejectValue("numberMap[receivedFinancing]","error.continuousTraining.
    continuousTrainingFinancingValidator.atLeastOne");
    errors.rejectValue("numberMap[totalSubscriptionFees]","error.continuousTraining.
    continuousTrainingFinancingValidator.atLeastOne");
    errors.rejectValue("numberMap[otherFinancing]","error.continuousTraining.
    continuousTrainingFinancingValidator.atLeastOne");
} else
{
    receivedFinancing = receivedFinancing == null ? 0 : receivedFinancing.
    floatValue();
    totalSubscriptionFees = totalSubscriptionFees == null ? 0 :
    totalSubscriptionFees.floatValue();
    otherFinancing = otherFinancing == null ? 0 : otherFinancing.floatValue();
    //log.error("receivedFinancing: "+receivedFinancing);
    //log.error("totalSubscriptionFees: "+totalSubscriptionFees);
    //log.error("otherFinancing: "+otherFinancing);
    var totalIncome = object.getNumberMap().get("totalIncome").floatValue();
    //log.error("totalIncome: "+totalIncome);
    var calculatedTotalIncome = receivedFinancing + totalSubscriptionFees +
    otherFinancing;
    //log.error("calculatedTotalIncome: "+calculatedTotalIncome);
    if(totalIncome != calculatedTotalIncome)
    errors.rejectValue("numberMap[totalIncome]","error.continuousTraining.
    continuousTrainingFinancingValidator.totalIncome");
}
```

**continuousTrainingParticipantsValidator**
Questa validazione controlla che la somma dei vari partecipanti 'inserita'(calcolata) sia quella effettivamente corretta.
I dati che sono presi in considerazione sono:
- Numero Istituzioni Pubbliche (numberOfPublicInstitutions)
- Numero di Imprese (numberOfCompanies)
- Numero di imprese del Terzo Settore (numberOfThirdSector)
- Numero totali di partecipanti al corso (totalNumberOfParticipants)
Per maggiori dettagli cfr. modello dati dell'entità in questione.
var numberOfPublicInstitutions = object.getIntegerMap().get("numberOfPublicInstitutions");
var numberOfCompanies = object.getIntegerMap().get("numberOfCompanies");
var numberOfThirdSector = object.getIntegerMap().get("numberOfThirdSector");
if(numberOfPublicInstitutions == null && numberOfCompanies == null && numberOfThirdSector == null){
    errors.rejectValue("integerMap[numberOfPublicInstitutions]","error.continuousTraining.continuousTrainingParticipantsValidator.atLeastOne");
    errors.rejectValue("integerMap[numberOfCompanies]","error.continuousTraining.continuousTrainingParticipantsValidator.atLeastOne");
    errors.rejectValue("integerMap[numberOfThirdSector]","error.continuousTraining.continuousTrainingParticipantsValidator.atLeastOne");
} else {
    numberOfPublicInstitutions = numberOfPublicInstitutions == null ? 0 : numberOfPublicInstitutions.intValue();
    numberOfCompanies = numberOfCompanies == null ? 0 : numberOfCompanies.intValue();
    numberOfThirdSector = numberOfThirdSector == null ? 0 : numberOfThirdSector.intValue();
    //log.error("numberOfPublicInstitutions: "+numberOfPublicInstitutions);
    //log.error("numberOfCompanies: "+numberOfCompanies);
    //log.error("numberOfThirdSector: "+numberOfThirdSector);
    var totalNumberOfParticipants = object.getIntegerMap().get("totalNumberOfParticipants").intValue();
    //log.error("totalNumberOfParticipants: "+totalNumberOfParticipants);
    var calculatedTotalNumberOfParticipants = numberOfPublicInstitutions + numberOfCompanies + numberOfThirdSector;
    //log.error("calculatedTotalNumberOfParticipants: "+calculatedTotalNumberOfParticipants);
    if(totalNumberOfParticipants != calculatedTotalNumberOfParticipants)
        errors.rejectValue("integerMap[totalNumberOfParticipants]","error.continuousTraining.continuousTrainingParticipantsValidator.totalNumberOfParticipants");
}

continuousTrainingMultipleContributorValidator
Questa validazione verifica che esista almeno un Docente interno (contributor)
Per maggiori dettagli cfr. il modello modello dati dell entità in questione.

var contributorWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(object, "it.cilea.wf.model.WfItemElement","contributor", wfService);
if(contributorWfElementSet.isEmpty()){
    errors.rejectAndLocate("error.continuousTraining.contributor.required", "contributor");
}

continuousTrainingNumberOfInternalTeachersValidator
Questa validazione verifica che la somma dei docenti inserita (Numero dei docenti interni coinvolti nel corso - numberOfInternalTeachers) sia quella effettivamente corretta.
Se una persona viene inserita sia nei Responsabili scientifici e nei Componenti interni viene contata come una unica persona.
Per maggiori dettagli cfr. il modello modello dati dell entità in questione.
var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","owner", wfService);
var calculatedNumberOfInternalTeachers = ownerWfElementSet.size();
getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","contributor", wfService);

var numberOfInternalTeachers = object.getIntegerMap().get("numberOfInternalTeachers");

if (contributorWfElementSet.size() > 0) {
    var contributorWfElement = null;
    var contributorWfElementSetIterator = contributorWfElementSet.iterator();
    while (contributorWfElementSetIterator.hasNext()) {
        var found = false;
        contributorWfElement = contributorWfElementSetIterator.next();
        if(ownerWfElementSet.size() == 0){
            calculatedNumberOfInternalTeachers++;
        } else {
            var ownerWfElement = null;
            var ownerWfElementSetIterator = ownerWfElementSet.iterator();
            while (ownerWfElementSetIterator.hasNext()) {
                ownerWfElement = ownerWfElementSetIterator.next();
                if(ownerWfElement.getPersonMap().get("ownerId").getId() == contributorWfElement.getPersonMap().get("contributorId").getId()) {
                    found = true;
                    break;
                }
            }
            if(found == false){
                calculatedNumberOfInternalTeachers++;
            }
        }
    }
}

if(numberOfInternalTeachers != calculatedNumberOfInternalTeachers)
    errors.rejectValue("integerMap[numberOfInternalTeachers]","error.continuousTraining.continuousTrainingNumberOfInternalTeachersValidator");

var wfTask = object;
var wfItem = wfService.getWfItem(wfTask.getWfItemId());
var permissions = "r";
var hasPowerPermissions = Packages.it.cilea.wf.util.WfUtil.isAnyItemCreatorIdentityMatchingThisTaskIdentity(wfTask, wfService);
if (Packages.java.lang.Boolean.TRUE.equals(hasPowerPermissions))
    hasPowerPermissions=true;
else
    hasPowerPermissions=false;

var hasHelpdeskStartedFlow=Packages.it.cilea.wf.util.WfUtil.isAnyItemCreatorIdentityMatchingAuthoritiesWithThisResource(wfTask, "/HELPDESK/contract.profile", wfService, gaAuthorizationService);
if (Packages.java.lang.Boolean.FALSE.equals(hasHelpdeskStartedFlow))
    hasHelpdeskStartedFlow=false;

var hasHelpdeskStartedFlow=Packages.it.cilea.wf.util.WfUtil.isAnyItemCreatorIdentityMatchingAuthoritiesWithThisResource(wfTask, "/HELPDESK.profile", wfService, gaAuthorizationService);
if (Packages.java.lang.Boolean.TRUE.equals(hasHelpdeskStartedFlow))
    hasHelpdeskStartedFlow=true;
else
    hasHelpdeskStartedFlow=false;
}
else
    hasHelpdeskStartedFlow=true;

switch("" + String(wfItem.getWfState().getDescription())){
    case "draft":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="crwfd"
        else
            permissions="r";
        break;
    case "inquiry":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwfd"
        else
            permissions="r";
        break;
    case "fiscalCheck":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="r"
        else
            permissions="r";
        break;
    case "revisionAfterFiscalCheck":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwf"
        else
            permissions="r";
        break;
    case "approval":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwf"
        else
            permissions="r";
        break;
    case "approvedAwaitingCountersignedContract":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwf"
        else
            permissions="r";
        break;
    case "signed":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwf"
        else
            permissions="r";
        break;
    case "closed":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwf"
        else
            permissions="r";
        break;
    case "suspended":
        if(hasPowerPermissions||hasHelpdeskStartedFlow)
            permissions="rwf"
        else
            permissions="r";
        break;
}
Questa è una permissions logic che costruisce dinamicamente i permessi per i contratti per il team dipartimentali
Se l'utente che sta effettuando la creazione del contratto è in visione dipartimentale e appartiene ad un team dipartimentale, vengono assegnati i permessi di scrittura a questo team e quelli di sola lettura a livello Divisione Ricerca.

```javascript
wfTask.setPermissions(permissions);

wfIdentityLogicContextVisionProjectPermissionsJs

var wfTask = object;
var wfItem = wfService.getWfItem(wfTask.getWfItemId());
var permissions = "r";
var formViews = object.getFormViews();
var isOnlyRead=true;

//recupero l'unità organizzativa associata (dipartimento) all'identity/team di questo task
var organizationUnitFromIdentity=Packages.it.cilea.ga.util.GaUtil.
getOrganizationUnitFromIdentityForDepartment(wfTask.getIdentityId(), gaService,
gaAuthorizationService);

//recupero dall'item la lista dei dipartimenti
var internalOrganizationUnitList=Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem,
    "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);

//controllo il ruolo del dipartimento
//isOnlyRead è false se il dipartimento è principale
//isOnlyRead è true se il dipartimento è aggregato
while(internalOrganizationUnitListIterator.hasNext()){
    var internalOrganizationUnit=internalOrganizationUnitListIterator.next();
    var ou = internalOrganizationUnit.getOrganizationUnitMap().get("ouId");
    if (ou.getId().equals(organizationUnitFromIdentity.getId())){
        //cerco prima roleId che è il ruolo standard.
        //se non lo trovo ricado in ouRole che è quello legacy
        var role=internalOrganizationUnit.getWfDictionaryMap().get("roleId");
        //if (role==null)
        //    role=internalOrganizationUnit.getWfDictionaryMap().get("ouRole");
        isOnlyRead=Packages.it.cilea.wf.util.WfUtil.isOnlyReadPermission(role,
        wfItem, "wfIdentityLogicContextVision");
        if (Packages.java.lang.Boolean.TRUE.equals(isOnlyRead))
            isOnlyRead=true;
        else
            isOnlyRead=false;
        break;
    }
}

//verifico se c'è match tra le identity di chi ha creato l'item e l'identity di questo task
//serve per capire se il contratto in questione è stato creato da Divisione Ricerca o da Dipartimento
//e per dare "power permissions" solo al team che ha fatto la creazione.
var hasPowerPermissions=Packages.it.cilea.wf.util.WfUtil.
isAnyItemCreatorIdentityMatchingThisTaskIdentity(wfTask, wfService);

if (Packages.java.lang.Boolean.TRUE.equals(hasPowerPermissions))
    hasPowerPermissions=true;
else {
    //se hasPowerPermissions è false allora dovrebbe essere stato creato dalla Divisione Ricerca
    //se però viene cambiato in corso d'opera il dipartimento, non ci sarà più match tra il team dipartimentale che ha creato l’item e il team dipartimentale che dovrebbe accederci.
```
// la mancanza di match ci sarebbe anche se il contratto è creato dal team dipartimentale generale e l'utente che sta accedendo fa parte di un team dipartimentale specializzato (declinato sui contratti e/o "flavoured")

// per gestire questi scenario controllo se le identity di chi ha effettuato la creazione sono quelle dipartimentali.

var isItemCreatedByDepartmentAuthority=Packages.it.cilea.wf.util.WfUtil.isAllItemCreatorIdentityOfGivenAuthority(wfTask, wfService, gaAuthorizationService, "ROLE_DEPARTMENT");

// se chi ha creato aveva identity dipartimentali...
if (Packages.java.lang.Boolean.TRUE.equals(isItemCreatedByDepartmentAuthority))
{
    if (isOnlyRead)
        hasPowerPermissions=false;
    else
        hasPowerPermissions=true;
} else {
    // se chi ha creato NON aveva identity dipartimentali (Helpdesk o Divisione Ricerca)...
    hasPowerPermissions=false;
}

switch("" + String(wfItem.getWfState().getDescription())){
    case "draft":
        if(hasPowerPermissions & & !isOnlyRead)
            permissions="crwfd"
        else if(!hasPowerPermissions & & !isOnlyRead)
        {
            permissions = "rw";
            formViews = "contract/form1.display,contract/form2.display,
            contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
        }
        else
            permissions="r";
        break;
    case "inquiry":
        if(hasPowerPermissions & & !isOnlyRead)
            permissions="rwfd"
        else if(!hasPowerPermissions & & !isOnlyRead)
        {
            permissions = "rw";
            formViews = "contract/form1.display,contract/form2.display,
            contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
        }
        else
            permissions="r";
        break;
    case "fiscalCheck":
        if(hasPowerPermissions & & !isOnlyRead)
            permissions="r"
        else if(!hasPowerPermissions & & !isOnlyRead)
        {
            permissions = "rw";
            formViews = "contract/form1.display,contract/form2.display,
            contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
        }
        else
            permissions="r";
        break;
    case "revisionAfterFiscalCheck":
        if(hasPowerPermissions & & !isOnlyRead)
            permissions="rwf"
        else if(!hasPowerPermissions & & !isOnlyRead)
        {
            permissions = "rw";
            formViews = "contract/form1.display,contract/form2.display,
            contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
        }
        else
            permissions="r";
        break;
    case "approval":
        if(hasPowerPermissions & & !isOnlyRead)
            permissions="rwf"
else if(!hasPowerPermissions && !isOnlyRead){
    permissions = "rw";
    formViews = "contract/form1.display,contract/form2.display,contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
} else
    permissions = "r";
break;

case "approvedAwaitingCountersignedContract":
    if(hasPowerPermissions && !isOnlyRead)
        permissions = "rwf";
    else if(!hasPowerPermissions && !isOnlyRead){
        permissions = "rw";
        formViews = "contract/form1.display,contract/form2.display,contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
    } else
        permissions = "r";
break;

case "signed":
    if(hasPowerPermissions && !isOnlyRead)
        permissions = "rwf";
    else if(!hasPowerPermissions && !isOnlyRead){
        permissions = "rw";
        formViews = "contract/form1.display,contract/form2.display,contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
    } else
        permissions = "r";
break;

case "closed":
    if(hasPowerPermissions && !isOnlyRead)
        permissions = "rwf";
    else if(!hasPowerPermissions && !isOnlyRead){
        permissions = "rw";
        formViews = "contract/form1.display,contract/form2.display,contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
    } else
        permissions = "r";
break;

case "suspended":
    if(hasPowerPermissions && !isOnlyRead)
        permissions = "rwf";
    else if(!hasPowerPermissions && !isOnlyRead){
        permissions = "rw";
        formViews = "contract/form1.display,contract/form2.display,contract/form3.display,contract/form4.display,contract/form5.display,contract/form6.display,contract/form7";
    } else
        permissions = "r";
break;
}

wfTask.setPermissions(permissions);
wfTask.setFormViews(formViews);

ownerValidatorContract
Questa validazione controlla che esista almeno un responsabile scientifico nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo owner
Per maggiori dettagli cfr. excel modello dati dell'entità Contratto
if (object.getId() == null)
    if (object.getPersonMap().get("owner") == null
        errors.rejectValue("personMap[owner]", "error.contract.owner.required");
    ) else {
        var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
            getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
        if (!Packages.java.lang.Boolean.TRUE.equals(wfItem.getBooleanMap().get("legacy")) && ownerSet.size() < 1 && object.getPersonMap().get("owner") == null
            errors.rejectAndLocate("error.contract.owner.atLeast1", "owner");
    }

administrativeOwnerValidatorContract
Questa validazione controlla che esista almeno un referente amministrativo nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo administrativeOwner
Per maggiori dettagli cfr. excel modello dati dell'entità Contratto

var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "administrativeOwner", wfService);
    if (ownerSet.size() < 1)
        errors.rejectAndLocate("error.contract.administrativeOwner.atLeast1", "administrativeOwner");

organizationUnitValidatorContract
Questa validazione controlla che esista almeno una unità organizzativa interna nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo internalOrganizationUnit
Per maggiori dettagli cfr. excel modello dati dell'entità Contratto

var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
    if (orgUnitSet.size() < 1)
        errors.rejectAndLocate("error.contract.multipleInternalOrganization.atLeast1", "internalOrganizationUnit");

organizationUnitRoleValidatorContract
Questa validazione controlla che esista almeno un dipartimento specificato e che uno ed uno solo sia marcato come "Principale"
Per maggiori dettagli cfr. excel modello dati dell'entità in questione
Configurazioni della ValidateLogic:
- ap.TIPOLOGIA.internalOrganizationUnit.role.main
var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
if (orgUnitSet.size()<1){
    errors.rejectAndLocate("error.contract.multipleInternalOrganization.atLeast1", "internalOrganizationUnit");
} else {
    var internalOrganizationUnitCoordinatorRole=Packages.it.cilea.core.
configuration.util.ConfigurationUtil.getConfigValue("ap.con.internalOrganizationUnit.role.main");
    if (!internalOrganizationUnitCoordinatorRole){
        throw "Configuration variable ap.con.internalOrganizationUnit.role.main
MUST BE DEFINED";
    } else {
        internalOrganizationUnitCoordinatorRole=new Packages.java.lang.Integer
(internalOrganizationUnitCoordinatorRole);
    }
    var orgUnitSetIterator=orgUnitSet.iterator();
    var count=0;
    while (orgUnitSetIterator.hasNext()){
        var element=orgUnitSetIterator.next();
        var ouRole=element.getWfDictionaryMap().get("roleId");
        if (ouRole!=null && internalOrganizationUnitCoordinatorRole.equals
(ouRole.getId())){
            count++;
        }
    }
    if (count==0){
        errors.rejectAndLocate("error.contract.multipleInternalOrganization.
missingMain", "internalOrganizationUnit");
    } else if (count>1) {
        errors.rejectAndLocate("error.contract.multipleInternalOrganization.
tooManyMain", "internalOrganizationUnit");
    }
}

ownerAndOrganizationUnitMatchValidatorContract
Questa validazione controlla che esista per ogni responsabile scientifico del contratto, una unità organizzativa che sia interna nel momento della data di
avvio del contratto (startDate) oppure se non c'è questa data, in questo momento.
dal punto di vista del modello sono dei dati di tipo: InternalOrganizationUnit, owner
Per maggiori dettagli cfr. excel modello dati dell’entità Contratto

var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
var ownerDepartmentSet=new Packages.java.util.HashSet();
var internalOrganizationUnitSet=new Packages.java.util.HashSet();
var startDateFromWfItem=wfItem.getDateMap().get("startDate");
if (startDateFromWfItem!=null){
    startDateFromWfItem=startDateFromWfItem.getTime();
    if (ownerSet.size()>0){
        var ownerSetIterator=ownerSet.iterator();
        while (ownerSetIterator.hasNext()){
            var owner=ownerSetIterator.next();
        }
    } else {
        throw "Configuration variable ap.con.internalOrganizationUnit.role.main
MUST BE DEFINED";
    }
} else {
    throw "Configuration variable ap.con.internalOrganizationUnit.role.main
MUST BE DEFINED";
}
var ownerElement = ownerSetIterator.next();

var person = ownerElement.getPersonMap().get("ownerId");

var ownerDepartmentPositionSearch = new Packages.it.cilea.ga.
command.PositionSearchCommand();

ownerDepartmentPositionSearch.setPersonId(person.getId());
ownerDepartmentPositionSearch.setDate(startDateFromWfItem);
ownerDepartmentPositionSearch.setOrganizationUnitTypeDescription("department");
ownerDepartmentPositionSearch.setDiscriminator("research");

var ownerDepartmentPositionSet = gaService.getPositionSearchList(ownerDepartmentPositionSearch, 0);
var maxPriority = Packages.it.cilea.ga.util.GaUtil.getMaxPriority(ownerDepartmentPositionSet, "research", "department");

var ownerDepartmentPositionSetIterator = ownerDepartmentPositionSet.iterator();

while (ownerDepartmentPositionSetIterator.hasNext()) {
    var position = ownerDepartmentPositionSetIterator.next();
    if (maxPriority == null || maxPriority.equals(position.getPriority())) {
        ownerDepartmentSet.add(position.getOrganizationUnit());
        break;
    }
}

if (ownerDepartmentSet.size() > 0) {
    if (orgUnitSet.size() > 0) {
        var orgUnitSetIterator = orgUnitSet.iterator();

        while (orgUnitSetIterator.hasNext()) {
            var wfItemElement = orgUnitSetIterator.next();
            var orgUnit = wfItemElement.getOrganizationUnitMap().get("ouId");
            internalOrganizationUnitSet.add(orgUnit);
        }
    }
}

if (!internalOrganizationUnitSet.containsAll(ownerDepartmentSet)) {
    var nonMatchedDepartmentCollection = Packages.org.apache.commons.collections.CollectionUtils.subtract(ownerDepartmentSet, internalOrganizationUnitSet);

    var nonMatchedDepartmentCollectionIterator = nonMatchedDepartmentCollection.iterator();
    var nonMatchedDepartmentString = ";
    var counter = 0;

    while (nonMatchedDepartmentCollectionIterator.hasNext()) {
        var nonMatchedDepartment = nonMatchedDepartmentCollectionIterator.next();
        nonMatchedDepartmentString += nonMatchedDepartment.getDescription();
    }

    var df = Packages.org.apache.commons.lang.time.FastDateFormat.getInstance("dd/MM/yyyy");
    var dateAsString = df.format(startDateFromWfItem);
    errors.rejectAndLocate("error.contract.multipleInternalOrganization.matchWithDepartmentOwnerNotFound", "internalOrganizationUnit", [dateAsString, nonMatchedDepartmentString], null);
}
customerValidatorContract
Questa validazione controlla che esista almeno un committente nel tab “Dati generali”.
Dal punto di vista del modello dati si tratta degli elementi di tipo customer
Per maggiori dettagli cfr. excel modello dati dell’entità Contratto

```javascript
var customerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", 
    "customer", wfService);
if (customerSet.size()<1)
    errors.rejectAndLocate("error.contract.customer.atLeast1", 
    "customer");
```

currencyAndTotalAmountValidatorContract
Questa validazione controlla che nel tab “Corrispettivi”, nel caso di selezione di valuta diversa da EURO, vengano specificati

- importo in valuta originaria
- tasso di cambio
- data rilevazione tasso di cambio

Dal punto di vista del modello dati si tratta degli attributi:

- totalAmountOriginalCurrency
- exchangeRate
- exchangeRateDate

Per maggiori dettagli cfr. excel modello dati dell’entità Contratto

```javascript
var currency = object.getWfDictionaryMap().get("currency");
if(currency != null){
    var currencyId = object.getWfDictionaryMap().get("currency").getId();
    var dictionary = wfService.getWfDictionary(currencyId);
    var currencyCode = dictionary.getStringMap().get("code");

    if (currencyCode != "currency.eur"){ 
        if (!object.getNumberMap().get("totalAmountOriginalCurrency")
            errors.rejectValue("numberMap[totalAmountOriginalCurrency]", 
            "error.contract.funding.totalAmountOriginalCurrency.
            required");
        
        if (!object.getNumberMap().get("exchangeRate"){ 
            errors.rejectValue("numberMap[exchangeRate]","error.contract.
            funding.exchangeRate.required");
        
        if (!object.getDateMap().get("exchangeRateDate")
            errors.rejectValue("dateMap[exchangeRateDate]","error.contract.
            funding.exchangeRateDate.required");
    
    }

}
```

wfUgovPjSenderValidatorContract
Questa logica, effettua check di consistenza sulle mappature degli oggetti IRIS/UGOV e, se superati, invia il contratto verso UGOV PJ rispettando le regole di NON SOVRASCRITTURA lato UGOV PJ.
Per maggiori dettagli su queste regole consultare il team di UGOV PJ.
Le controparti dei contratti IRIS vengono create su UGOV con le regole di mappature delle tassonomie che vengono condivise in fase di attivazione.
Se in corso d’opera dovesse nascere l’esigenza di censire delle nuove tipologie è possibile contattare l’HD.
Di seguito vengono riportate le informazioni inviate ad UGOV PJ, con la specifica dell’attributo del modello dell’entità Contratto IRIS
• Codice contratto: **identifier**
• Nome contratto: **description** troncata a 255 caratteri per limitazioni lato UGOV PJ
• Importo contratto **totalAmount**
  L'invio viene effettuato se l'importo è non nullo e maggiore di zero.
• Data inizio validità: **startDate**
• Data fine validità: **endDate**
• Riferimento Data Management System (Sistema Documentale): **externalDmsIdentifier**
  Inviato l'identifier del contratto se valorizzata a true la variabile di configurazione `ap.con.externalDmsIdentifier.autogenerated`, di default questa NON è definita.
  In tutti gli altri casi viene inviato **externalDmsIdentifier**.
• Responsabili scientifici: **owner**
  Inviate anche date di inizio e fine se valorizzata a true la variabile di configurazione `ap.contract.owner.startDate.required`, di default questa NON è definita.
  La data di fine viene inviata SOLO se minore della data recuperata dalla configurazione `ap.<wfItemType>.contributorAndOwner.validator.date.end.discriminator`.
• Partecipanti: **contributor**
  Inviate anche date di inizio e fine se valorizzata a true la variabile di configurazione `ap.contract.contributor.startDate.required`, di default questa NON è definita.
  La data di fine viene inviata SOLO se minore della data recuperata dalla configurazione `ap.<wfItemType>.contributorAndOwner.validator.date.end.discriminator`.
• Referenti interni: **administrativeOwner**
  Inviati se valorizzata a true la variabile di configurazione `ap.con.administrativeOwner.send2ugov.enable`. In caso di invio viene valutata anche la configurazione `ap.con.administrativeOwner.role.ugov.default` in cui bisogna specificare l'identificativo del ruolo PJ.
  Inviate anche date di inizio e fine se valorizzata a true la variabile di configurazione `ap.contract.administrativeOwner.startDate.required`.
  La data di fine viene inviata SOLO se minore della data recuperata dalla configurazione `ap.<wfItemType>.contributorAndOwner.validator.date.end.discriminator`.
• Unità organizzative interne: **internalOrganizationUnit**
• Enti Committenti: **customer**
  Inviato se valorizzata a true (che è anche il valore di default) la variabile di configurazione `ap.contract.customer.send2ugov.enable`.
• Tipo contratto: come da mappatura tassonomia ap-item-type.xls
• Schema di finanziamento: come da mappatura tassonomia ap-item-type.xls
• Stato contratto: come da mappatura tassonomia ap-item-type.xls

Per maggiori dettagli cfr. excel modello dati dell'entità Contratto

**Configurazioni della ValidateLogic:**

- `ap.contract.customer.send2ugov.enable`
- `ap.con.cup.send2ugov.enable`
- `ap.con.externalDmsIdentifier.autogenerated`
- `ap.contract.administrativeOwner.startDate.required`
- `ap.contract.contributor.startDate.required`
- `ap.TIPOLOGIA.contributorAndOwner.validator.date.end.discriminator`
- `ap.con.administrativeOwner.role.ugov.default`
- `ap.con.administrativeOwner.send2ugov.enable`

```javascript
var Logic = function(wfItem, dtoPj, superLogic, errors, messageUtil, request, gaService, wfService, wfTaskService, searchBuilderTemplate, log) {
    this.wfItem=wfItem;
    this.dtoPj=dtoPj;
    this.superLogic=superLogic;
    this.errors=errors;
    this.messageUtil=messageUtil;
    this.request=request;
    this.gaService=gaService;
    this.wfService=wfService;
    this.wfTaskService=wfTaskService;
    this.searchBuilderTemplate=searchBuilderTemplate;
    this.log=log;
    this.setInternalOrgUnitList = function() {
        var internalOrganizationUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","internalOrganizationUnit", wfService);
        var internalOrganizationUnitSetIterator = internalOrganizationUnitSet.iterator();
        while (internalOrganizationUnitSetIterator.hasNext()){  
            var element = internalOrganizationUnitSetIterator.next();
            var orgUnit = element.getOrganizationUnitMap().get("ouId");
            var orgUnitRole = element.getWfDictionaryMap().get("roleId");
            dtoPj.getDTOUoPjs().add(superLogic.getOrganizationUnit(orgUnit, orgUnitRole, errors));
        }
    }
}
```
this.setFinanciers = function() {
    var sendCustomer2Ugov = Packages.it.cilea.core.configuration.util.
        ConfigurationUtil.getConfigValue("ap.contract.customer.send2ugov.enable");
    if (sendCustomer2Ugov && sendCustomer2Ugov.trim().equalsIgnoreCase("true")){
        var contractorSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
            getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "customer", wfService);
        var contractorSetIterator = contractorSet.iterator();
        while (contractorSetIterator.hasNext()){{
            var element = contractorSetIterator.next();
            var enteFinanziatore = superLogic.getEnteFinanziatore(element, this.getFinancing(), contractorSet.size(), errors);
            dtoPj.getDTOEfPjs().add(enteFinanziatore);
        }
    }
    this.setPartners = function() {};
    this.getResearchNature = function() {
        return null;
    };
    this.getContribution = function() {
        return null;
    };
    this.getCofinancing = function() {
        return null;
    };
    this.getFinancing = function() {
        return wfItem.getNumberMap().get("totalAmount");
    };
    this.getStartDate = function() {
        return wfItem.getDateMap().get("startDate");
    };
    this.getEndDate = function() {
        return wfItem.getDateMap().get("endDate");
    };
    this.getCurrentEndDate = function() {
        return null;
    };
    this.getIdentifier = function() {
        return wfItem.getIdentifier();
    };
    this.getIdPj = function() {
        if (wfItem.getIntegerMap().get("ugovId") != null){
            return wfItem.getIntegerMap().get("ugovId");
        } else {
            return null;
        }
    };
    this.getDescription = function() {
        if (wfItem.getDescription() != null) {
            var length = wfItem.getDescription().length();
            return wfItem.getDescription().substring(0, (length > 255) ? 255 : length);
        } else {
            return null;
        }
    };
    this.getCup = function() {
        var sendCup2Ugov = Packages.it.cilea.core.configuration.util.ConfigurationUtil.
            getConfigValue("ap.con.cup.send2ugov.enable");
        if (sendCup2Ugov & sendCup2Ugov.trim().equalsIgnoreCase("true")){
            return wfItem.getStringMap().get("cup");
        } else {
            return null;
        }
    }
}
this.getAcronym = function() {
    if(wfItem.getStringMap().get("acronym") != null){
        var length = wfItem.getStringMap().get("acronym").length();
        return wfItem.getStringMap().get("acronym").substring(0,(length>50)?50:
        length);
    } else {
        return null;
    }
}

this.getExternalDmsIdentifier = function() {
    var externalDmsIdentifierAutogenerated = Packages.it.cilea.core.configuration.
util.ConfigurationUtil.getConfigValue("ap.con.externalDmsIdentifier.autogenerated");
    if("true".equals(externalDmsIdentifierAutogenerated)){
        return wfItem.getIdentifier();
    }else {
        return wfItem.getStringMap().get("externalDmsIdentifier");
    }
}

this.isSynchronizable = function() {
    var financing=this.getFinancing();
    var legacy=wfItem.getBooleanMap().get("legacy");
    if (Packages.java.lang.Boolean.TRUE.equals(legacy) || financing==null ||
    Packages.java.math.BigDecimal.ZERO.equals(financing))
        return Packages.java.lang.Boolean.FALSE;
    else
    return Packages.java.lang.Boolean.TRUE;
}

this.getProjectType = function() {
    return superLogic.getProjectType(wfItem);
}

this.getFinanceSchema = function() {
    return this.getProjectType();
}

this.getPjInternalCode = function() {
    return null;
}

this.getGrantNumber = function() {
    return null;
}

this.getManHoursCofinancing = function() {
    return null;
}

this.getFlPnrr = function() {
    return new Packages.java.lang.Integer("0");
}

this.isAdministrativeOwnerStartDateRequired = function() {
    var administrativeOwnerStartDateRequired=Packages.it.cilea.core.configuration.
    util.ConfigurationUtil.getConfigValue("ap.contract.administrativeOwner.starDate.required");
    if (administrativeOwnerStartDateRequired){
        if (administrativeOwnerStartDateRequired.trim().equalsIgnoreCase
        ("true"){
            return Packages.java.lang.Boolean.TRUE;
        }else if (administrativeOwnerStartDateRequired.trim().equalsIgnoreCase
        ("false"){
            return Packages.java.lang.Boolean.FALSE;
        }else {
            throw "ap.contract.administrativeOwner.starDate.required MUST
        BE true or false or null";
        }
    }else {
        return Packages.java.lang.Boolean.TRUE;
    }
}

this.isContributorStartDateRequired = function() {
    var contributorStartDateRequired=Packages.it.cilea.core.configuration.util.
    ConfigurationUtil.getConfigValue("ap.contract.contributor.starDate.required");
    if (contributorStartDateRequired){
        if (contributorStartDateRequired.trim().equalsIgnoreCase
        ("true"){
            return Packages.java.lang.Boolean.TRUE;
        }else if (contributorStartDateRequired.trim().equalsIgnoreCase
        ("false"){
            return Packages.java.lang.Boolean.FALSE;
        }else {
            throw "ap.contract.contributor.starDate.required MUST
        BE true or false or null";
        }
    }else {
        return Packages.java.lang.Boolean.TRUE;
    }
}
if (contributorStartDateRequired.trim().equalsIgnoreCase("true")){
    return Packages.java.lang.Boolean.TRUE;
} else if (contributorStartDateRequired.trim().equalsIgnoreCase("false")){
    return Packages.java.lang.Boolean.FALSE;
} else {
    throw "ap.contract.contributor.startDate.required MUST BE true or false";
}
} else {
    return Packages.java.lang.Boolean.TRUE;
};
this.isOwnerStartDateRequired = function() {
    var ownerStartDateRequired=Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.contract.owner.startDate.required");
    if (ownerStartDateRequired){
        if (ownerStartDateRequired.trim().equalsIgnoreCase("true")){
            return Packages.java.lang.Boolean.TRUE;
        } else if (ownerStartDateRequired.trim().equalsIgnoreCase("false")){
            return Packages.java.lang.Boolean.FALSE;
        } else {
            throw "ap.contract.owner.startDate.required MUST BE true or false";
        }
    } else {
        return Packages.java.lang.Boolean.TRUE;
    }
};
new Logic(wfItem, dtoPj, superLogic, errors, messageUtil, request, gaService, wfService, wfTaskService, searchBuilderTemplate, log);

**uniqueIdentifierContract**
Questa validazione controlla l'unicità dell'identificativo dell'oggetto.
Questa verifica viene fatta per i casi di generazione di codice "parlante" con possibile collisione con identificativi già presenti.

```javascript
var paramMap = new Packages.java.util.HashMap();
paramMap.put("identifier", object.getIdentifier());
var count=searchBuilderTemplate.getSingleObject("getWfItemCount", paramMap);
if (count>1)
    errors.rejectValue("identifier", "error.contract.identifier.notUnique");
```

**fiscalCheckFieldsetValidatorContract**
Questa validazione impedisce l'edit degli attributi

- fiscalCheckPerson
- fiscalCheckTariff
- fiscalCheckVatCode
- fiscalCheckNotes

relativi alla sezione "Controllo fiscale" del tab "Corrispettivi"
var reject = 0;
var discriminator = ["fiscalCheckPerson", "fiscalCheckTariff", "fiscalCheckVatCode", "fiscalCheckNotes"];
var column = ["person", "string", "string", "string"];
columnLen = column.length;
var value = [];
for (i = 0; i < columnLen; i++) {
  if (Packages.java.lang.Boolean.TRUE.equals(Packages.it.cilea.wf.util.WfUtil.isFileAttributeChanged(wfService, wfItem, column[i] + "." + discriminator[i]))) {
    reject = 1;
    break;
  }
}
if (reject > 0) {
  errors.rejectValue("personMap[fiscalCheckPerson]", "error.contract.fiscalCheck.edit.notAllowed");
  errors.rejectValue("stringMap[fiscalCheckTariff]", "error.contract.fiscalCheck.edit.notAllowed");
  errors.rejectValue("stringMap[fiscalCheckVatCode]", "error.contract.fiscalCheck.edit.notAllowed");
  errors.rejectValue("stringMap[fiscalCheckNotes]", "error.contract.fiscalCheck.edit.notAllowed");
}

isAccountancyInfoEnabledTrue
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Verifica che la configurazione ap.contract.accountancyInfo.enabled sia settata a TRUE, e abilita quindi il campo su cui si va a fare la validazione dopo Configurazioni della ValidateLogic:

* ap.contract.accountancyInfo.enabled

var accountancyInfoConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.contract.accountancyInfo.enabled");
var accountancyInfo = null;
if (accountancyInfoConf != null) {
  try {
    accountancyInfo = new Packages.java.lang.Boolean(Packages.org.apache.commons.lang.StringUtils.trim(accountancyInfoConf));
  } catch (e) {} 
}
if (accountancyInfo != null && accountancyInfo == true)

isFiscalCheckEnabledTrue
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Verifica che la configurazione ap.contract.fiscalCheck.enabled sia settata a TRUE, e abilita quindi il campo su cui si va a fare la validazione dopo Configurazioni della ValidateLogic:

* ap.contract.fiscalCheck.enabled
var fiscalCheckConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.contract.fiscalCheck.enabled");
var fiscalCheck = null;
if (fiscalCheckConf!=null){
    try{
        fiscalCheck = new Packages.java.lang.Boolean(Packages.org.apache.commons.lang.StringUtils.trim(fiscalCheckConf));
    }catch(e){}
}

if (fiscalCheck != null && fiscalCheck == true)
    isYearGreaterOrEqualThanConfigurationContract

Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Verifica anche che il contratto in questione sia relativo ad un anno maggiore o uguale a quello definito nella configurazione ap.con.year.validation.threashold.
Se la configurazione NON è definita o non può essere convertita in un numero intero valido viene settato uguale a 0.
Come default, NON viene definita nel sistema.
Questo approccio serve per evitare lo scatenarsi delle validazioni su vecchi contratti che non rispettano nuove e più stringenti validazioni, ed evitare quindi di doverre fare una bonifica a tappeto.

Configurazioni della ValidateLogic:

- ap.con.year.validation.threashold

var yearTreasholdConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.con.year.validation.threashold");
var year = new Packages.java.lang.Integer(0);
if (yearTreasholdConf!=null){
    try{
        year = new Packages.java.lang.Integer(Packages.org.apache.commons.lang.StringUtils.trim(yearTreasholdConf));
    }catch(e){}
}

if (object.getYear() != null && object.getYear().compareTo(year) >= 0)
    areWarningEnabledContract

Questa è un'applicability rule che pilota i messaggi di warning per i contratti.
Se la configurazione ap.con.warningAcknowlegement.validation.enabled è settata a true o non è valorizzata i messaggi di warning sono abilitati.
Se la configurazione ap.con.warningAcknowlegement.validation.enabled è settata a false i messaggi di warning NON sono abilitati.
Come default, NON viene valorizzata nel sistema.

Configurazioni della ValidateLogic:

- ap.con.warningAcknowlegement.validation.enabled
var warningAcknowlegementFlagConf=Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.con.warningAcknowlegement.validation.enabled");
warningAcknowlegementFlagConf=Packages.org.apache.commons.lang.StringUtils.trim(warningAcknowlegementFlagConf);
(!Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase("false", warningAcknowlegementFlagConf))

**isExternalDmsEnabledContract**
Questa è un'applicability rule che pilota le validazioni relative fascicolo del DMS di Ateneo (Document Management System).
Se la variabile **ap.con.externalDmsIdentifier.enabled** è abilitata
Se la variabile **ap.con.externalDmsIdentifier.enabled** è settata a false la validazione NON è abilitata

**Configurazioni della ValidateLogic:**
- **ap.con.externalDmsIdentifier.enabled**

var flagConf=Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.con.externalDmsIdentifier.enabled");
flagConf=Packages.org.apache.commons.lang.StringUtils.trim(flagConf);
(!Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase("false",flagConf))

**wfActionLogicCreateRmItem**
Questa logica viene chiamata in ingresso nello stato approved e crea un nuovo rmItem a partire da quello di AP.
Se esiste già, allora lo aggiorna.
Questo è collegato all'item di AP essendo il suo identificativo presente fra i suoi metadati (e viceversa).
Vengono aggiunti/aggiornati anche i dipartimenti associati all'item.

function createNewOrgUnitElementEquipment(department, rmItem){
    var orgUnitElement = new Packages.it.cilea.ga.model.OrganizationUnitElement();
    orgUnitElement.setDiscriminator("equipment");
    orgUnitElement.setOrganizationUnit(department);
    orgUnitElement.setOrganizationUnitId(department.getOrganizationUnitId());
    orgUnitElement.getRmItemMap().put("equipment", rmItem);
    return orgUnitElement;
}

var wfItem= wfService.getWfItem(object.getId());
var rmItemMap=wfItem.getRmItemMap();
var rmItem=rmItemMap.get("equipment");
if(rmItem==null){
    rmItem = new Packages.it.cilea.ga.model.RmItem();
    var today = new Packages.java.util.Date();
    rmItem.setCreateDate(today);
    rmItem.setRmItemTypeId(10);
    rmItem.getIntegerMap().put("apItemSourceId", object.getId());
}
rmItem.setDescription(wfItem.getDescription());
rmItem.getStringMap().put("apItemSourceIdentifier", object.getIdentifier());
//aggiungo o sovrascrivo l'rmItem
rmItemMap.put("equipment", rmItem);
//sovrascrivo la mappa, ci metto quella aggiornata
wfItem.setRmItemMap(rmItemMap);
//mi tengo da parte l'id prima del salvataggio
//lo userò dopo per vedere se devo cercare orgUnitElement
var rmItemId=rmItem.getId();

//salvo
gaService.saveOrUpdate(rmItem);
wfService.saveOrUpdate(wfItem);

//PARTE PER AGGIUNTA AL DIPARTIMENTO

//prima tolgo tutti gli orgUnitElement, se ce ne sono attaccati all'rmItem
//poi li riaggiungo tutti
if(rmItemId != null){
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("rmItemValueId", rmItem.getId());
    var orgUnitElementList = searchBuilderTemplate.query("getOrganizationUnitElementByRmItemId", paramMap);
    var orgUnitElementListIter = orgUnitElementList.iterator();
    while(orgUnitElementListIter.hasNext()){
        var orgUnitElement = orgUnitElementListIter.next();
        gaService.deleteOrganizationUnitElement(orgUnitElement.getId());
    }
}

var departmentSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
currentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","internalOrganizationUnit", wfService);
var departmentSetIterator = departmentSet.iterator();
while(departmentSetIterator.hasNext()){
    var departmentElement = departmentSetIterator.next();
    var departmentEmpty = departmentElement.getOrganizationUnitMap().get("ouId");
    var department = gaService.getOrganizationUnit(departmentEmpty.getId());
    //a differenza della versione di prima, so già che non ci sono orgUnitElement
    //attaccati al mio rmItem
    //quindi ne aggiungo direttamente uno
    department.getOrganizationUnitElementSet().add(  
        createNewOrgUnitElementEquipment(department, rmItem)  
    );
    gaService.saveOrUpdate(department);
}

wfActionLogicDeleteRmItem

Questa logica viene chiamata in ingresso nello stato submitted ed elimina il rmItem agganciato precedentemente (se presente) a quello di AP. Questo è collegato all'item di AP essendo il suo identificativo presente fra i suoi metadati (e viceversa). Vengono eliminati anche i dati dei dipartimenti associati all'item.

//Logica che viene chiamata in enter allo stato submitted (In Valutazione)
//Se proviene da approved(Approvato) avrà sicuramente un rmItem collegato che
//dovo eliminare
//In ordine quindi devo scollegare-eliminare dati/link)  
//-dai dipartimenti agganciati  
//-dalla versione che è stata creata poco prima  
//-eliminare l'rmItem poiché inutile

 *}
//log.error("Sono in wfActionLogicDeleteRmItem");

var wfItem = wfService.getWfItem(object.getId());
var rmItemMap = wfItem.getRmItemMap();
var rmItem = rmItemMap.get("equipment");
//log.error("rmItem ricavato" + rmItem);

// eseguo tutta la procedura solo se effettivamente ho anche un rmItem collegato (in precedenza)
if(rmItem != null){
    //log.error("rmItem non nullo");

    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("rmItemValueId", rmItem.getId());
    //ricerco le organizzazioni a cui sganciare l'equipment
    var orgUnitElementList = searchBuilderTemplate.query("getOrganizationUnitElementByRmItemId", paramMap);
    //log.error("orgUnitElementList size: " + orgUnitElementList.size());

    var orgUnitElementListIter = orgUnitElementList.iterator();
    while(orgUnitElementListIter.hasNext()){
        var orgUnitElement = orgUnitElementListIter.next();
        //log.error("orgUnitElement id: " + orgUnitElement.getId());
        gaService.deleteOrganizationUnitElement(orgUnitElement.getId());
        //var orgUnitId = gaService.getOrganizationUnit orgUnitElement.
        //dovrei anche salvare la orgUnit?
        //log.error("orgUnitElement deleted ");
    }

    //ricerco le altre possibili versioni che hanno questo dato equipment
    var wfItemDataList = searchBuilderTemplate.query("getWfItemDataByRmItemId", paramMap);
    //log.error("wfItemDataList size: " + wfItemDataList.size());

    var wfItemDataListIter = wfItemDataList.iterator();
    while(wfItemDataListIter.hasNext()){
        var wfItemData = wfItemDataListIter.next();
        var wfItem = wfService.getWfItem(wfItemData.getWfItem().getId());
        //log.error("wfItem id: " + wfItemData.getWfItem().getId());
        //log.error("wfItem: " + wfItem);
        //rmItemMap.put("equipment", null);
        //wfItem.setRmItemMap(rmItemMap);
        wfItem.getRmItemMap().put("equipment", null);
        wfService.saveOrUpdate(wfItem);
        //log.error("wfItem saved and data deleted ");
    }

    //una volta che ho cancellato gli elementi a cui poteva essere agganciato
    //posso andare a cancellare l'rmItem
    gaService.deleteRmItem(rmItem.getId());
    //log.error("RM_ITEM DELETED");
    //facendo così dovrei eliminare anche tutti dati agganciati all'rmItem}
**wfActionLogicManufacturerCsv**
Questa logica viene invocata in ingresso nello stato approved se la variabile di configurazione `ap.eqp.ugovpj.inventory.enabled` è settata a true.
La configurazione menzionata è quella che abilita il recupero dei beni di inventario da UGOV.
La logica crea o aggiorna il metadato `manufacturerCsv` con l’elenco delle denominazioni dei produttori dei singoli beni di inventario che fanno parte dell'attrezzatura.
Nel caso in cui una grande attrezzatura sia composta da beni di cui una parte prodotti da una stessa azienda, questa verrà presentata una sola volta.
Stante la natura aggregata del metadato, NON sarà possibile risalire a quale azienda abbia prodotto quale bene.
A questo scopo è sempre possibile utilizzare, a partire dell'identificativo del singolo bene di inventario, il servizio di UGOV Inventario per il recupero dell’informazione.
**Configurazioni della ActionLogic:**
- `ap.eqp.ugovpj.inventory.enabled`

```javascript
var inventoryEnabled = ConfigurationUtil.getLenientBoolean("ap.eqp.ugovpj.inventory.enabled", false);
if (inventoryEnabled){
    var manufacturerArray=[];
    var manufacturerCsv=null;
    var inventoryGoodSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "inventoryGood", wfService);
    var inventoryGoodSetIterator=inventoryGoodSet.iterator();
    while (inventoryGoodSetIterator.hasNext()){
        var inventoryGoodElement=inventoryGoodSetIterator.next();
        var inventoryGoodRmItem=inventoryGoodElement.getRmItemMap().get("inventoryGoodId");
        var manufacturer=inventoryGoodRmItem.getStringMap().get("manufacturer");
        if (manufacturer!=null && manufacturerArray.indexOf(manufacturer)==-1)
            manufacturerArray.push(manufacturer);
    }
    for (var i=0;i<manufacturerArray.length;i++){
        if (manufacturerCsv==null)
            manufacturerCsv=new StringBuilder();
        if (i>0)
            manufacturerCsv.append(", ");
        manufacturerCsv.append(manufacturerArray[i]);
    }
    if (manufacturerCsv !=null)
        wfItem.getStringMap().put("manufacturerCsv", manufacturerCsv.toString());
    else
        wfItem.getStringMap().put("manufacturerCsv", null);
}
```

**deleteRmItemEquipmentValidator**
Questa validazione va a cancellare i dati relativi all’rmItem equipment quando viene cancellato l’oggetto padre `wfItem`
Per maggiori dettagli cfr. il modello dati dell’entità in questione.
//log.error("Sono in deleteRmItemEquipmentValidator");
var wfItem = wfService.getWfItem(object.getId());
var rmItemMap =wfItem.getRmItemMap();
var rmItem = rmItemMap.get("equipment");

if (rmItem != null){

    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("rmItemValueId", rmItem.getId());

    var orgUnitElementList = searchBuilderTemplate.query("getOrganizationUnitElementByRmItemId", paramMap);
    //log.error("orgUnitElementList size: " + orgUnitElementList.size());

    var orgUnitElementListIter = orgUnitElementList.iterator();
    while(orgUnitElementListIter.hasNext()){ 
        var orgUnitElement = orgUnitElementListIter.next();
        gaService.deleteOrganizationUnitElement(orgUnitElement.getId());
    }

    //ricerco le altre possibili versioni che hanno questo dato equipment
    var wfItemDataList = searchBuilderTemplate.query("getWfItemDataByRmItemId", paramMap);
    //log.error("wfItemDataList size: " + wfItemDataList.size());

    var wfItemDataListIter = wfItemDataList.iterator();
    while(wfItemDataListIter.hasNext()){ 
        var wfItemData = wfItemDataListIter.next();
        var wfItem = wfService.getWfItem(wfItemData.getWfItem().getId());
        wfItem.getRmItemMap().put("equipment", null);
        wfService.saveOrUpdate(wfItem);
    }

    //una volta che ho cancellato gli elementi a cui poteva essere agganciato
    //posso andare a cancellare l'rmItem
    gaService.deleteRmItem(rmItem.getId());
    //facendo così dovrei eliminare anche tutti dati agganciati all'rmItem
}
//se non fosse popolato sarebbe strano

isEquipmentPlaceRequired
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le validazioni a cui è agganciata altrimenti no.
Questa regola ritorna true se la configurazione ap.eqp.place.required è settata a true.

Configurazioni della ValidateLogic:

* ap.eqp.place.required

```java
    {
      "true"==StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.eqp.place.
required"))
    }
```
isEquipmentPurchaseValueRequired
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le validazioni a cui è agganciata altrimenti no.
Questa regola ritorna true se la configurazione ap.eqp.purchaseValue.required è settata a true.
Configurazioni della ValidateLogic:
  * ap.eqp.purchaseValue.required

```
  {
    "true"==StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.eqp.
purchaseValue.required"))
  }
```

isEquipmentInUseRequired
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le validazioni a cui è agganciata altrimenti no.
Questa regola ritorna true se la configurazione ap.eqp.inUse.required è settata a true.
Configurazioni della ValidateLogic:
  * ap.eqp.inUse.required

```
  {
    "true"==StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.eqp.
required"))
  }
```

isEquipmentEsfriClassificationRequired
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le validazioni a cui è agganciata altrimenti no.
Questa regola ritorna true se la configurazione ap.eqp.esfriClassification.required è settata a true.
Configurazioni della ValidateLogic:
  * ap.eqp.esfriClassification.required

```
  {
    "true"==StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.eqp.
esfriClassification.required"))
  }
```

isEquipmentInventoryGoodEnabled
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le validazioni a cui è agganciata altrimenti no.
Questa regola ritorna true se la configurazione ap.eqp.ugovpj.inventory.enabled è settata a true.
Questa configurazione abilita l’import dei Beni di Inventario da UGOV.
Configurazioni della ValidateLogic:
  * ap.eqp.ugovpj.inventory.enabled

```
  {
    "true"==StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.eqp.ugovpj.
inventory.enabled"))
  }
```

urlPublicCalendarEquipmentValidator
Questa validazione verifica che sia presente il metadato urlPublicCalendar se il metadato publicCalendar è valorizzato a true.
Il messaggio di errore è contenuto nell’etichetta error.equipment.urlPublicCalendar.required
Per maggiori dettagli cfr. il modello modello dati dell’entità in questione.
if (Boolean.TRUE.equals(object.getBooleanMap().get("publicCalendar"))) {
    if (StringUtils.isBlank(object.getStringMap().get("urlPublicCalendar"))) {
        errors.rejectValue("stringMap[urlPublicCalendar]","error.equipment.urlPublicCalendar_required");
    }
}

repairabilityEquipmentValidator
Questa validazione verifica che sia presente il metadato repairability se il metadato inUse è valorizzato a false.
Il messaggio di errore è contenuto nell'etichetta error.equipment.repairability_required
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

if (Boolean.FALSE.equals(object.getBooleanMap().get("inUse"))) {
    if (StringUtils.isBlank(object.getWfDictionaryMap().get("repairability"))) {
        errors.rejectValue("wfDictionaryMap[repairability]","error.equipment.repairability_required");
    }
}

tariffEquipmentValidator
Questa validazione verifica che sia presente il metadato tariff se è presente il dizionario di valore applications.tariff (RM - Attrezzature - Applicazioni) nella sezione Applicazioni.
Il messaggio di errore è contenuto nell'etichetta error.equipment.tariff_required
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

var applicationsSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","applications", wfService);
var applicationsSetIterator=applicationsSet.iterator();
while(applicationsSetIterator.hasNext()){
    var applicationsElement=applicationsSetIterator.next();
    var dictionary=applicationsElement.getWfDictionaryMap().get("application");
    if (dictionary!=null && "applications.tariff"==dictionary.getStringMap().get("code")) {
        if (StringUtils.isBlank(object.getStringMap().get("tariff"))) {
            errors.rejectValue("stringMap[tariff]","error.equipment.tariff_required");
        }
    }
}

purchaseValueInventoryGoodValidator
Questa validazione verifica che, se il calcolo automatico è attivato, il valore di acquisto totale sia la somma dei singoli valori di acquisto dei beni di inventario che sono stati importati da UGOV.
Il messaggio di errore è contenuto nell'etichetta error.equipment.autocalculatedInventoryGoodsTotalValue.inconsistent
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.
// controllo solo nel caso in cui il calcolo automatico è attivato
if (object.getBooleanMap().get("autocalculatedInventoryGoodsTotalValue") == true) {
    var inventoryGoodSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
    (object, "it.cilea.wf.model.WfItemElement", "inventoryGood", wfService);
    var inventoryGoodSetIterator = inventoryGoodSet.iterator();
    var calculatedPurchaseValue = BigDecimal.ZERO;
    while (inventoryGoodSetIterator.hasNext()) {
        var inventoryGoodElement = inventoryGoodSetIterator.next();
        var inventoryGoodRmItem = inventoryGoodElement.getRmItemMap().get
        ("inventoryGoodId");
        var conventionalValue = inventoryGoodRmItem.getNumberMap().get
        ("conventionalValue");
        // se conventionalValue non è presente, cioè null, aggiungo zero
        calculatedPurchaseValue = calculatedPurchaseValue.add(conventionalValue != null
            ? conventionalValue : BigDecimal.ZERO);
    }
    if (!calculatedPurchaseValue.stripTrailingZeros().equals(object.getNumberMap().get
        ("purchaseValue").stripTrailingZeros())) {
        errors.rejectAndLocate("error.equipment.autocalculatedInventoryGoodsTotalValue.
            inconsistent", "purchaseValue");
    }
}

externalOrganizationOwnerEquipmentValidator
Questa validazione verifica che sia presente il metadato externalOrganizationOwner se il metadato internalOwnership è valorizzato a false.
Il messaggio di errore è contenuto nell'etichetta error.equipment.externalOrganizationOwner.required
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

if (Boolean.FALSE.equals(object.getBooleanMap().get("internalOwnership")))
    if (object.getOrganizationUnitMap().get("externalOrganizationOwner") == null) {
        errors.rejectValue("organizationUnitMap[externalOrganizationOwner]","error.equipment.
            externalOrganizationOwner.required");
    }

wfStartLogicEquipment
Questa logica viene eseguita in fase di creazione di una nuova attrezzatura.
Effettua il setup dei seguenti attributi con i valori tra parentesi
- publicCalendar (false)
- internalOwnership (true)
- autocalculatedInventoryGoodsTotalValue (true)
Il setting viene effettuato se la configurazione ap.eqp.ugovpj.inventory.enabled è valorizzata a true

Per maggiori dettagli cfr. modello dati dell'entità equipment.
Configurazioni della StartLogic:
- ap.eqp.ugovpj.inventory.enabled
if ("true"==StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.eqp.ugovpj.inventory.enabled"))){
    if(wfItem.getBooleanMap().get("autocalculatedInventoryGoodsTotalValue")==null) {
        wfItem.getBooleanMap().put("autocalculatedInventoryGoodsTotalValue", true);
    }
}
if(wfItem.getBooleanMap().get("publicCalendar")==null) {
    wfItem.getBooleanMap().put("publicCalendar",false);
}
if(wfItem.getBooleanMap().get("internalOwnership")==null) {
    wfItem.getBooleanMap().put("internalOwnership",true);
}
true;

equipmentValidatorLaboratory
Questa validazione controlla che, se esiste una grande attrezzatura collegata al laboratorio, questa attrezzatura sia in stato "approved", altrimenti lancia un errore.
Dal punto di vista del modello dati si tratta degli elementi di tipo laboratoryEquipmentLink
Per maggiori dettagli cfr. modello dati delle entità laboratorio e grande attrezzatura

//log.error("entro nel validator");
var equipmentSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getChildWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "laboratoryEquipmentLink", wfService);

//log.error("equipmentSet size= "+equipmentSet.size());
if(equipmentSet.size()>0){
    var equipmentSetIterator = equipmentSet.iterator();
    while (equipmentSetIterator.hasNext()) {
        var equipmentElement = equipmentSetIterator.next();
        var equipment = equipmentElement.getChild();
        var equipmentState = equipment.getWfState().getDescription();
        //log.error("stato dell'equipment"+equipmentState);
        if(!Packages.org.apache.commons.lang.StringUtils.equals("approved", equipmentState))
        {
            //log.error("trovato un laboratory non approvato");
            errors.rejectAndLocate("error.laboratory.equipment.mustBeApproved", "laboratoryEquipmentLink");
        }
    }
}

wfActionLogicCreatePersonElementIncoming
Questa logica viene triggerata in ingresso nello stato validated.
Crea un nuovo elemento nella sezione della mobilità in ingresso all'interno del profilo personale (CV Scientifico)
Se esiste già allora lo aggiorna.

function obtainIncomingTypeDictionary(incomingTypeDescription){
    //prima di tutto ne cerco
    var incomingTypeDictionaryList = gaService.getGaDictionarySearchList(incomingTypeDescription, "incomingType", null, null, null);
    //se non ne trovo allora creo da zero il dizionario, lo salvo e lo restituisco
    if(incomingTypeDictionaryList.size() == 0){
        var newIncomingTypeDictionary = new Packages.it.cilea.ga.model.GaDictionary();
        newIncomingTypeDictionary.setDescription(incomingTypeDescription);
        newIncomingTypeDictionary.setDiscriminator("incomingType");
        newIncomingTypeDictionary.getStringMap().put("MANAGED_BY", "system");
        //in modo tale che non possa essere modificato da interfaccia
    }
else{
    // altrimenti lo restituisco direttamente
    // non vado a ciclare tutta la lista, ne ho solo uno di dictionary fatto così
    return incomingTypeDictionaryList.get(0);
}
var newPersonElementIncoming = new Packages.it.cilea.ga.model.PersonElement();

// ed ora passo ad agganciarci tutte le informazioni che gli servono
newPersonElementIncoming.setPersonId(tutorPerson.getPersonId());
newPersonElementIncoming.setDiscriminator("incoming");
newPersonElementIncoming.getPersonMap().put("tutoree", wfItem.getPersonMap().get("owner");

// visto che il miurExternalOrganization è un elemento ma unico, ricavo il primo elemento e da lì il tutto
var miurExternalOrganizationUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "miurExternalOrganization", wfService);

if(!miurExternalOrganizationUnitSet.isEmpty()){
    var miurExternalOrganizationUnit = miurExternalOrganizationUnitSet.iterator().next().getOrganizationUnitMap().get("externalOrganization");
    newPersonElementIncoming.getOrganizationUnitMap().put("miurExternalOrganization", miurExternalOrganizationUnit);
}

newPersonElementIncoming.getStringMap().put("apItemSourceIdentifier", wfItemIdentifier);
newPersonElementIncoming.getIntegerMap().put("apItemSourceId", wfItemId);

if(tutorWfItemElement.getDateMap().get("startDate")!=null)
    newPersonElementIncoming.getDateMap().put("startDate", tutorWfItemElement.getDateMap().get("startDate");
if(tutorWfItemElement.getDateMap().get("endDate")!=null)
    newPersonElementIncoming.getDateMap().put("endDate", tutorWfItemElement.getDateMap().get("endDate");

// gestione dictionary
var incomingTypeDescription = wfItem.getWfItemType().getDescription();
var incomingTypeDictionary = obtainIncomingTypeDictionary(incomingTypeDescription);
newPersonElementIncoming.getDictionaryMap().put("incomingType", incomingTypeDictionary);

// aggiungo metadato per marcare come è stato gestito
newPersonElementIncoming.getStringMap().put("managedBy", "system");

// ora che è completo, vado a salvarlo
gaService.saveOrUpdate(newPersonElementIncoming);

// lo aggiungo alla person e salvo la person
tutorPerson.getPersonElementSet().add(newPersonElementIncoming);
gaService.saveOrUpdate(tutorPerson);

}
var wfItem = wfService.getWfItem(object.getId());
var wfItemId = wfItem.getId();
var wfItemIdentifier = wfItem.getIdentifier();

var tutorWfItemElementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "tutor", wfService);

if(!tutorWfItemElementSet.isEmpty()){
    var tutorWfItemElementSetIterator = tutorWfItemElementSet.iterator();
    while(tutorWfItemElementSetIterator.hasNext()){
        var tutorWfItemElement = tutorWfItemElementSetIterator.next();
        var tutorPerson = tutorWfItemElement.getPersonMap().get("contributorId"); //restituisce la persona completa?
        tutorPerson = gaService.getPerson(tutorPerson.getId()); //nel dubbio la ricavo dal service

        var incomingPersonElementSet = tutorPerson.getPersonElementSet("incoming");
        var incomingPersonElementSetIterator = incomingPersonElementSet.iterator();
        //vado a ciclare il set per vedere se quell'element esiste già
        //se esiste già, lo rimuovo
        while(incomingPersonElementSetIterator.hasNext()){
            var incomingPersonElement = incomingPersonElementSetIterator.next();
            var incomingPersonElementSourceIdentifier = incomingPersonElement.getStringMap().get("apItemSourceIdentifier");
            var incomingPersonElementSourceId = incomingPersonElement.getIntegerMap().get("apItemSourceId");

            //doppio controllo, così gestisco il caso in cui l'item non abbia un identifier come può succedere se proviene da tabelle di frontiera
            if(Packages.org.apache.commons.lang.StringUtils.equals(wfItemIdentifier, incomingPersonElementSourceIdentifier) || (wfItemId==incomingPersonElementSourceId) ){
                tutorPerson.getPersonElementSet().remove(incomingPersonElement);
                gaService.deletePersonElement(incomingPersonElement.getId());
                gaService.saveOrUpdate(tutorPerson);
            }
        }
    }
}

externalOrganizationValidatorMobilityIncoming
Questa validazione controlla che sia presente una istituzione esterna
Per maggiori dettagli cfr. modello dati della Mobilità in ingresso
var miurExternalOrganizationSet=FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","miurExternalOrganization", wfService);
if(miurExternalOrganizationSet.size()==0)
errors.rejectAndLocate("error.person.internal.mobilityIncoming.miurExternalOrganization");

externalOrganizationAndRequestPendingValidatorMobilityIncoming
Questa validazione controlla che non sia presente sia una istituzione esterna (da anagrafica) che la richiesta di inserimento
Per maggiori dettagli cfr. modello dati della Mobilità in ingresso

var miurExternalOrganizationSet=FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","miurExternalOrganization", wfService);
if(miurExternalOrganizationSet.size()>0 && object.getStringMap().get("miurExternalOrganizationsRequested")!=null)
errors.rejectAndLocate("error.person.internal.mobilityIncoming.miurExternalOrganizationsRequested", "miurExternalOrganization");
errors.rejectValue("stringMap[miurExternalOrganizationsRequested]","error.person.internal.mobilityIncoming.miurExternalOrganizationsRequested");

dercValidatorMobilityIncoming
Questa validazione verifica che non siano presenti più di 3 parole chiave ERC.
Per maggiori dettagli cfr. modello dati della Mobilità in ingresso

var ercSet=Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","ercKeyword", wfService);
if(!ercSet.isEmpty())
if(ercSet.size()>3)
errors.rejectAndLocate("error.mobilityIncoming.tooMany.ERC","ercKeyword");

deletePersonElementFromOldTutor
Questa logica si occupa di eliminare dal profilo personale del tutor/persona in ingresso (CV Scientifico) il collegamento tra le due persone
Questa logica viene triggerata in fase di salvataggio.

//log.error("Inside deletePersonElementFromOldTutor");
var wfItem = wfService.getWfItem(object.getId());
var wfItemId = wfItem.getId();
var wfItemIdentifier = wfItem.getIdentifier();

//ricavo i vecchi tutor memorizzati su db
var tutorApItemElementSetOLD = wfItem.getWfItemElementSet("tutor");
//ricavo i tutor aggiornati dal fragment
var tutorApItemElementSetNEW = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "it.cilea.wf.model.WfItemElement","tutor", wfService);

//ciclo quelli ricavati dal DB per andare a ricavare quelli che sono stati eliminati
var tutorApItemElementSetOLDIterator = tutorApItemElementSetOLD.iterator();
while(tutorApItemElementSetOLDIterator.hasNext()){
    //log.error("inside old tutor set");
    //if(tutorApItemElementSetOLDIterator != null){
        var tutorApItemElement = tutorApItemElementSetOLDIterator.next();
        if(!tutorApItemElementSetNEW.contains(tutorApItemElement)){
            //log.error("Found old tutor");
            var tutorPerson = tutorApItemElement.getPersonMap().get("contributorId");
            tutorPerson = gaService.getPerson(tutorPerson.getId());
            //log.error("is: " + tutorPerson.getDisplayValue());
            var incomingPersonElementSet = tutorPerson.getPersonElementSet("incoming");
            var incomingPersonElementSetIterator = incomingPersonElementSet.iterator();
            while(incomingPersonElementSetIterator.hasNext()){ //log.error("inside incoming set of tutor");
                var incomingPersonElement = incomingPersonElementSetIterator.next();
                var incomingPersonElementSourceIdentifier = incomingPersonElement.getStringMap().get("apItemSourceIdentifier");
                var incomingPersonElementSourceId = incomingPersonElement.getIntegerMap().get("apItemSourceId");
                if( Packages.org.apache.commons.lang.StringUtils.equals(wfItemIdentifier, incomingPersonElementSourceIdentifier) ||
                    (wfItemId == incomingPersonElementSourceId) ){
                    tutorPerson.getPersonElementSet().remove(incomingPersonElement);
                    gaService.deletePersonElement(incomingPersonElement.getId());
                    gaService.saveOrUpdate(tutorPerson);
                    //log.error("deleted old personelement and person saved");
                }
            }
        }else{
            //log.error("tutorApItemElementSetOLDIterator IS NULL");
        }
    }

isTypePlantVariety
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera la tipologia (WfItemType) e attiva la validazione solo per le proprietà intellettuali di tipo Varietà vegetale.

wfService.refresh(object);
object.getWfItemType().getIdentifier() == "PAT.PLANT-VARIETY"
**isTypeIndustrial**

Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera la tipologia (WfItemType) e attiva la validazione solo per le proprietà intellettuali di tipo **Brevetto industriale**.

```java
wsService.refresh(object);
object.getWfItemType().getIdentifier() == "PAT.INDUSTRIAL"
```

**intellectualPropertyInventionLinkValidator**

```java
var linkSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "inventionIntellectualPropertyLink", wsService);
if (linkSet.isEmpty()){
    errors.rejectAndLocate("error.intellectualProperty.invention.required", "inventionIntellectualPropertyLink");
} else{
    var withdrawnInventions = 0;
    var linkSetIterator = linkSet.iterator();
    while(linkSetIterator.hasNext()){
        var link = linkSetIterator.next();
        var inventionParent = wsService.getWfItem(link.getParentId());
        if(!inventionParent.getWithdrawn())
            withdrawnInventions++;
    }
    if(linkSet.size() == withdrawnInventions)
        errors.rejectAndLocate("error.intellectualProperty.invention.required", "inventionIntellectualPropertyLink");
}
```

**wfStartLogicPatent**

Questa logica viene eseguita in fase di creazione di una nuova proprietà intellettuale.

```java
var applicationNumber = wfItem.getStringMap().get(“applicationNumber”);
wfItem.setIdentifier(applicationNumber);
wfItem.getStringMap().put(“applicationNumberEpoDocCompliant”, applicationNumber);

var applicationDate = wfItem.getDateMap().get(“applicationDate”);
if (applicationDate != null){
    var year = applicationDate.get(Packages.java.util.Calendar.YEAR);
wfItem.setYear(year);
}

wfItem.getBooleanMap().put(“visibleOnPortal”, true);
if(wfItem.getBooleanMap().get(“recoveryMetadataFromEPO”) == null || Packages.java.lang.Boolean.FALSE.equals(wfItem.getBooleanMap().get("recoveryMetadataFromEPO"))){
    var linkedInventionId = wfItem.getIntegerMap().get("linkedInventionId");
    var linkedInvention = wsService.getWfItem(linkedInventionId);
    var inventionOwnerSet = FragmentUtil.
```
getCurrentFragmentSetByParentAndDiscriminator(linkedInvention, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
    var inventionOwnerSetIterator = inventionOwnerSet.iterator();
    while (inventionOwnerSetIterator.hasNext()) {
        var inventionOwner = inventionOwnerSetIterator.next();

        var patentOwnerSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
        var patentOwnerSetIterator = patentOwnerSet.iterator();
        while (patentOwnerSetIterator.hasNext()) {
            var patentOwner = patentOwnerSetIterator.next();

            if (inventionOwner.getPersonMap().get("ownerId").getId() != patentOwner.getPersonMap().get("ownerId").getId()) {
                var ownerElement = new WfItemElement();
                ownerElement.setDiscriminator("owner");
                ownerElement.setWfItemId(wfItem.getId());
                ownerElement.getPersonMap().put("ownerId", inventionOwner.getPersonMap().get("ownerId").getId());
                wfService.saveOrUpdate(ownerElement);
            }
        }
    }

    if (wfItem.getBooleanMap().get("recoveryMetadataFromEPO") == null || Packages.java.lang.Boolean.FALSE.equals(wfItem.getBooleanMap().get("recoveryMetadataFromEPO"))) {
        var inventionIntellectualPropertyLink = new Packages.it.cilea.wf.model.WfItemLink();
        inventionIntellectualPropertyLink.setParentId(linkedInventionId);
        inventionIntellectualPropertyLink.setChildId(wfItem.getId());
        inventionIntellectualPropertyLink.setDiscriminator("inventionIntellectualPropertyLink");
        var parentWfItemLinkSet = wfItem.getParentWfItemLinkSet();
        parentWfItemLinkSet.add(inventionIntellectualPropertyLink);
        wfItem.setParentWfItemLinkSet(parentWfItemLinkSet);
        wfItem.getIntegerMap().put("linkedInventionId", null);
        wfItem.getBooleanMap().put("isPriority", false);
        wfService.saveOrUpdate(inventionIntellectualPropertyLink);
    }

    wfService.saveOrUpdate(wfItem);

intellectualPropertyOwnerValidator
if (object.getId()==null){
  if (object.getPersonMap().get("owner") ==null)
    errors.rejectValue("personMap[owner]", "error.intellectualProperty.owner.
required");
} else {
  if (object.getPersonMap().get("owner") ==null)
  {
    var ownerSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","owner",
wService);
    if(ownerSet.isEmpty()){
      errors.rejectAndLocate("error.intellectualProperty.owner.
required", "owner");
    }
  }
}

intellectualPropertyApplicantEnabled
Questa è un’applicability rule che pilota le percentuali di titolarità.
Se la variabile ap.pat.ownership.percentage.enabled è settata a true o non è valorizzata vengono eseguite le logiche intellectualPropertyApplicantValidator e intellectualPropertyOwnershipValidator.
Se la variabile ap.pat.ownership.percentage.enabled è settata a false i messaggi di warning NON vengono eseguite le logiche intellectualPropertyApplicantValidator e intellectualPropertyOwnershipValidator.

var percentageEnableConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getConfigValue("ap.pat.ownership.percentage.enabled");
(!Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase("false",
percentageEnableConf))

intellectualPropertyApplicantValidator
Questa validazione controlla le percentuali di titolarità.
Verifica che sia presente almeno un titolare.
Verifica che se almeno un titolare ha la percentuale di titolarità, allora anche gli altri devono avere una percentuale.
Questa validazione scatta sempre, per disattivarla bisogna settare a false la configurazione ap.pat.ownership.percentage.enabled.
Se la configurazione ap.pat.ownership.percentage.enabled è settata a false nel fragment della titolarità non verrà visualizzato il campo relativo alla percentuale.
Per maggiori dettagli cfr. excel modello dati dell’entità Proprietà Intellettuali.
var applicantSet=FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","applicant", wfService);
  var applicantSetIterator=applicantSet.iterator();

  var applicantWithPercentageCount=0;
  var applicantTotalPercentageOwnership=BigDecimal.ZERO;
  while (applicantSetIterator.hasNext())
  {
    var currentApplicant=applicantSetIterator.next();
    var percentage=currentApplicant.getNumberMap().get("ownershipPercentage");
    if (percentage!=null){
      applicantWithPercentageCount++;
      applicantTotalPercentageOwnership=applicantTotalPercentageOwnership.add(percentage);
    }
  }

  if (applicantSet.size()==0){
    errors.rejectAndLocate("error.patent.applicant.required", "applicant");
  }
  if (applicantWithPercentageCount>0 && applicantWithPercentageCount!=applicantSet.size()){
    errors.rejectAndLocate("error.patent.applicant.ownershipPercentage", "applicant");
  }

intellectualPropertyOwnershipValidator
Questa validazione controlla che la somma delle percentuali di titolarità sia 100.
Questa validazione scatta sempre, per disattivarla bisogna settare a false la configurazione `ap.pat.ownership.percentage.enabled`.
Se la configurazione `ap.pat.ownership.percentage.enabled` è settata a false nel fragment della titolarità non verrà visualizzato il campo relativo alla percentuale.
Per maggiori dettagli cfr. excel modello dati dell'entità Proprietà Intellettuali.
var applicantSet=FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","applicant", wfService);
var applicantSetIterator=applicantSet.iterator();

var applicantWithPercentageCount=0;
var applicantTotalPercentageOwnership=BigDecimal.ZERO;

while (applicantSetIterator.hasNext())
{
var currentApplicant=applicantSetIterator.next();
var percentage=currentApplicant.getNumberMap().get("ownershipPercentage");

if (percentage!=null){
applicantWithPercentageCount++;
applicantTotalPercentageOwnership=applicantTotalPercentageOwnership.add(percentage);
}
}

if (!applicantTotalPercentageOwnership.equals(new BigDecimal(100))){
errors.rejectAndLocate("error.patent.applicant.totalOwnershipPercentage", "applicant");
}

intellectualPropertyLegalStatusValidator
Validator che controlla che sia inserito lo stato legale della proprietà intellettuale collegata

var legalStatusSet=FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","legalStatus", wfService);

if (legalStatusSet.isEmpty()){
errors.rejectAndLocate("error.intellectualProperty.legalStatus.required", "legalStatus");
}

uniquePatentValidator
Validator che controlla che non esista un'altra Proprietà Intellettuale con lo stesso Numero di Deposito.
Dal punto di vista del modello dati si tratta dell'attributo stringMamp[applicationNumber].
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto
var applicationNumber = object.getStringMap().get("applicationNumber");

if (applicationNumber != null) {
    var wfItemDataSearchCommand = new Packages.it.cilea.wf.command.WfItemDataSearchCommand();
    wfItemDataSearchCommand.setStringValue(applicationNumber);
    wfItemDataSearchCommand.setDiscriminator("applicationNumber");
    var wfItemDataSet = wfService.getWfItemDataSearchList(wfItemDataSearchCommand, 0);
    var wfItemDataSetIterator = wfItemDataSet.iterator();
    while (wfItemDataSetIterator.hasNext()) {
        var wfItemData = wfItemDataSetIterator.next();
        var wfItemObtained = wfService.getWfItem(wfItemData.getWfItem().getId());
        if (!wfItemObtained.getId().equals(object.getId()) && Packages.java.lang.Boolean.FALSE.equals(wfItemObtained.getWithdrawn())) {
            errors.rejectValue("stringMap[applicationNumber]", "error.patent.nonUniqueApplicationNumber");
            break;
        }
    }
}

**wfActionLogicSaveRPDFV3Evaluation**

*it.cilea.wf.surplus.logic.action.save.rpdf.WfActionLogicSaveRPDFV3Evaluation*

**wfStartLogicProjectDepartmentFundYear**

Questa logica viene eseguita in fase di creazione di un progetto per il fondo di Ateneo (ATE).
Viene impostato il valore dell'anno della campagna in corso.

var year = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("DEPARTMENT_FUND_YEAR");
wfItem.setYear(new Packages.java.lang.Integer(year));
true;

**wfStartLogicProjectDepartmentFundPositionLastDate**

Questa logica viene eseguita in fase di creazione di un progetto per il fondo di Ateneo (ATE).
Viene impostato il valore del dipartimento di riferimento per ogni singola scheda avviata.
if(wfItem.getOrganizationUnitMap().get("department") == null){
    var positionlastDate = Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue("DEPARTMENT_FUND_POSITION_LAST_DATE");
    //25/02/2020
    var intLastYear = parseInt(lastYear);
    var positionSearch = new Packages.it.cilea.ga.command.
PositionSearchCommand();
    positionSearch.setPersonId(wfItem.getPersonMap().get("owner").getId());
    positionSearch.setYear(intLastYear);
    positionSearch.setOrganizationUnitTypeDescription("department");
    var positionList = gaService.getPositionSearchList(positionSearch, 0);
    var maxPriority = Packages.it.cilea.ga.util.GaUtil.getMaxPriority
(positionList, null, positionSearch.getOrganizationUnitTypeDescription());
    var positionListIterator=positionList.iterator();
    while (positionListIterator.hasNext()){
        var position = positionListIterator.next();
        if (maxPriority==null || maxPriority.equals(position.
getPriority())){
            wfItem.getOrganizationUnitMap().put("department",
            position.getOrganizationUnit());
        }
    }
}
true;

projectDepartmentFundCreateValidator
Questa validazione viene eseguita in ingresso allo stato draft per i progetti per il fondo di Ateneo (ATE).
Viene consentita la creazione di una nuova scheda solo nell'intervallo previsto per la campagna in corso.
Questo intervallo viene definito con le configurazioni:

- DEPARTMENT_FUND_START_DATE
- DEPARTMENT_FUND_END_DATE

Le date devono avere il seguente formato yyyy-MM-dd HH:mm:ss

var configurationStart = "DEPARTMENT_FUND_START_DATE";
var configurationEnd = "DEPARTMENT_FUND_END_DATE";
var startDate = Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getConfigValue(configurationStart);
var endDate = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue
(configurationEnd);
if (startDate == null || endDate == null)
    throw "Configuration variables DEPARTMENT_FUND_START_DATE and
DEPARTMENT_FUND_END_DATE must be defined with format yyyy-MM-dd HH:mm:ss";
var formatter = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
var now = new java.util.Date();
var ateStart = formatter.parse(startDate);
var ateEnd = formatter.parse(endDate);
if (!(now.after(ateStart) & now.before(ateEnd))) {
    errors.reject("error.operation.notallowed");
}

projectDepartmentFundTeamValidator
Questa validazione viene eseguita in ingresso allo stato draft per i progetti per il fondo di Ateneo (ATE).
Viene avviata una nuova scheda solo se il responsabile è contenuto nell'apposito team denominato:
Team docenti e ricercatori per i progetti di Ateneo.
La validazione consente la creazione della scheda solo se la persona selezionata ha un utente agganciato censito nel team con id presente nella conf ap.

project.departmentFund.team.start
La validazione usa anche le seguenti conf:

- DEPARTMENT_FUND_POSITION_SERVICE_DATE
- DEPARTMENT_FUND_POSITION_LAST_DATE
Il formato di queste date è dd/MM/yyyy

```java
if(object.getPersonMap().get("owner") != null){
    var findUser = 0;
    var ownerId = object.getPersonMap().get("owner").getId();
    var userSet = gaService.getPerson(ownerId).getUserSet();
    userSetIterator = userSet.iterator();
    while (userSetIterator.hasNext()){
        var user = userSetIterator.next();
        var userId = user.getId();
        var searchCommand = new Packages.it.cilea.ga.command.TeaUserLinkSearchCommand();
        searchCommand.setUserId(userId);
        var linkList = gaService.getTeamUserLinkSearchList(searchCommand, 0);
        var listIterator = linkList.iterator();
        while(listIterator.hasNext()){
            var teamUserLink = listIterator.next();
            var teamId = teamUserLink.getTeamId();
            if(teamId == Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.project.departmentFund.team.start")){
                findUser = 1;
                break;
            }
        }
        if(findUser == 1){
            break;
        }
    }
    if(findUser == 0){
        errors.reject("error.start.position.team.departmentFund", [ConfigurationUtil.getConfigValue("DEPARTMENT_FUND_POSITION_SERVICE_DATE"),ConfigurationUtil.getConfigValue("DEPARTMENT_FUND_POSITION_LAST_DATE")], null);
    }
}
```

**projectDepartmentFundCheckDateEnterValidator**

Questa validazione viene eseguita in ingresso ad ogni stato per i progetti per il fondo di Ateneo (ATE).

Vengono controllate le date delle fasi successive prima di procedere con lo spostamento di stato per evitare di portare troppo in avanti nel flusso il progetto rispetto alle date previste dalla campagna.

```java
var gaUserDetail = Packages.it.cilea.ga.authorization.context.GaAuthorizationUserHolder.getUser();
if (!("ROLE_ADMIN"==gaUserDetail.getCurrentAuthorityIdentifier())){
    var state = wFItem.getWFState().getDescription();
    var configurationKeyStart = "DEPARTMENT_FUND_STATE_START_DATE_" + state.toUpperCase();
    var stateStartDate = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue(configurationKeyStart);
    var configurationKeyEnd = "DEPARTMENT_FUND_STATE_END_DATE_" + state.toUpperCase();
    var stateEndDate = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue(configurationKeyEnd);
    var formatter = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    var now = new java.util.Date();
    var start = formatter.parse(stateStartDate);
    var end = formatter.parse(stateEndDate);
    if(!(now.after(start) && now.before(end))){
        errors.reject("error.operation.notallowed");
    }
}```
Questa è una permissions logic che costruisce dinamicamente i permessi per i progetti per i team dipartimentali. Se la variabile di configurazione `ap.project-centralized-full-validation-flow.department.enable` è settata a `true` o non è valorizzata allora vengono assegnati al team dipartimentale i seguenti permessi:

- draft: `rwfd`
- added: `rwfd`
- submitted: `rf`
- approvedUnderNegotiation: `rf`

Se la variabile di configurazione `ap.project-centralized-full-validation-flow.department.enable` è settata a `false` allora vengono assegnati al team dipartimentale i seguenti permessi:

- draft: `none`
- added: `none`
- submitted: `r`
- approvedUnderNegotiation: `r`

Se la variabile di configurazione `ap.project-centralized-full-validation-flow.department.create` è settata a `true` o non è valorizzata allora viene consentita la creazione di nuovi progetti ai team dipartimentali. Se la variabile di configurazione `ap.project-centralized-full-validation-flow.department.create` è settata a `false` allora NON viene consentita la creazione di nuovi progetti ai team dipartimentali. In caso di modifica di queste variabili di configurazione procedere con un reload delle Configurazioni ed effettuare un nuovo upload della tassonomia per forzare anche il reload delle risorse che pilotano la visibilità dei bottoni di creazione dell’oggetto.

```javascript
var wfTask=object;
var wfItem=wfService.getWfItem(wfTask.getWfItemId());
var permissions=wfTask.getPermissions();
var departmentPreFlowEnableParam = Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue("ap.project-centralized-full-validation-flow.department.
enable");

var departmentPreFlowEnable="true"==departmentPreFlowEnableParam||!
departmentPreFlowEnableParam)?true:false;

switch("" + String(wfItem.getWfState().getDescription())){
    case "draft":
        if(departmentPreFlowEnable)
            permissions="crwfd"
        else
            permissions="";
        break;
    case "added":
        if(departmentPreFlowEnable)
            permissions="rwfd"
        else
            permissions="";
        break;
    case "submitted":
        if(departmentPreFlowEnable)
            permissions="rf";
        else
            permissions="r";
        break;
    case "approvedUnderNegotiation":
        if(departmentPreFlowEnable)
            permissions="rf";
        else
            permissions="r";
        break;
}

wfTask.setPermissions(permissions);
```
isYearGreaterOrEqualThanConfigurationProject
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Verifica anche che il progetto in questione sia relativo ad un anno maggiore o uguale a quello definito nella configurazione ap.prj.year.validation.
threshold.
Se la configurazione NON è definita o non può essere convertita in un numero intero valido viene settato uguale a 0.
Questo approccio serve per evitare lo scatenarsi delle validazioni su vecchi progetti che non rispettano nuove e più stringenti validazioni, ed evitare quindi di dovere fare una bonifica a tappeto.
Configurazioni della ValidateLogic:
- ap.prj.year.validation.threashold

areWarningEnabledProject
Questa è un'applicability rule che pilota i messaggi di warning per i progetti.
Se la variabile ap.prj.warningAcknowlegement.validation.enabled è settata a true o non è valorizzata i messaggi di warning sono abilitati
Se la variabile ap.prj.warningAcknowlegement.validation.enabled è settata a false i messaggi di warning NON sono abilitati
Configurazioni della ValidateLogic:
- ap.prj.warningAcknowlegement.validation.enabled

isExternalDmsEnabledProject
Questa è un'applicability rule che pilota le validazioni relative fascicolo del DMS di Ateneo (Document Management System).
Se la variabile ap.prj.externalDmsIdentifier.enabled è abilitata
Se la variabile ap.prj.externalDmsIdentifier.enabled è settata a false la validazione NON è abilitata
Configurazioni della ValidateLogic:
- ap.prj.externalDmsIdentifier.enabled

isCupRequiredProject
Questa è un'applicability rule che pilota le validazioni di obbligatorietà relative al CUP (cup).
Se la variabile ap.prj.cup.required è settata a true o non è valorizzata la validazione è abilitata
Se la variabile ap.prj.cup.required è settata a false la validazione NON è abilitata
Configurazioni della ValidateLogic:
Di seguito vengono riportate le informazioni inviate ad UGOV PJ, con la specifica dell'attributo del modello dell'entità Progetto IRIS

Se in corso d'opera dovesse nascere l'esigenza di censire delle nuove tipologie è possibile contattare l'HD.

Le controparti dei progetti IRIS vengono create su UGOV con le regole di mappature delle tassonomie che vengono condivise in fase di attivazione.

Per maggiori dettagli su queste regole consultare il team di UGOV PJ.

Questa logica, effettua check di consistenza sulle mappature degli oggetti IRIS/UGOV e, se superati, invia il progetto verso UGOV PJ rispettando le regole di NON SOVRASCRITTURA lato UGOV PJ.

Di seguito vengono riportate le informazioni inviate ad UGOV PJ, con la specifica dell'attributo del modello dell'entità Progetto IRIS

- Codice progetto: identifier
- Nome progetto
  Viene valutata la variabile di configurazione `ap.prj.descriptionAndGrantNumberConcatenationEnabled`.
  Se valorizzata a true allora vengono usati gli attributi concatenati `stringMap[grantNumber]-description`.

- Data inizio validità:
  Viene usato l'attributo `startDate` se non sono presenti delle proroghe (elementi di tipo extension). Se valorizzata a true allora viene usato l'attributo `originalEndDate`.

- Data fine validità del progetto dal punto di vista contabile:
  Viene usato l'attributo `endDate`

- Data fine validità del progetto dal punto di vista contabile:
  Viene usato l'attributo `endDate` se sono presenti delle proroghe (elementi di tipo extension). Se valorizzata a true allora viene usato l'attributo `originalEndDate`.

- CUP: `cup`
  Invio viene effettuato se l'importo è non nullo e maggiore di zero.

- Responsabili scientifici: `owner`
  Invio viene effettuato se l'importo è non nullo e maggiore di zero.

- Partecipanti: `contributor`
  Invio viene effettuato se l'importo è non nullo e maggiore di zero.

- Referenti interni: `administrativeOwner`
  Invio viene effettuato se l'importo è non nullo e maggiore di zero.

La data di fine viene inviata SOLO se minore della data recuperata dalla configurazione `ap.<wfItemType>.contributorAndOwner.validator.date.end.discriminator`.

Per i Responsabili scientifici, nella scheda, può essere visibile il Ruolo (visibilità pilotata da configurazione `ap.prj.owner.role.enabled` con default false).

E' necessario quindi inserire quel CUP in UGOV CO prima di potere effettuare invio, oppure è possibile disattivare l'invio settando `ap.project.cup.send2ugov.enable` a false.

Riferimento Data Management System (Sistema Documentale): `externalDmsIdentifier`

Invio viene effettuato se l'importo è non nullo e maggiore di zero.
• Unità organizzative interne: `internalOrganizationUnit`
• Finanziatori: `grantor`
  Invio se valorizzata a true la variabile di configurazione `ap.project.grantor.send2ugov.enable`.
  Se valorizzata a true la configurazione `ap.prj.grantor.financingQuota.enabled` nel form è possibile inserire anche le quote di finanziamento associate, che vengono inviate a PJ
• Partner: `partner`
  Invio se valorizzata a true la variabile di configurazione `ap.project.partner.send2ugov.enable`.
  Schema di finanziamento: come da mappatura tassonomia `ap-item-type.xls`
• Stato progetto: come da mappatura tassonomia `ap-item-type.xls`

Per maggiori dettagli cfr. excel modello dati dell’entità Progetto

Configurazioni della ValidateLogic:
• `ap.TIPOLOGIA.partner.main.required`
• `ap.prj.administrativeOwner.role.ugov.default`
• `ap.project.grantor.send2ugov.enable`
• `ap.project.partner.send2ugov.enable`
• `rm.orgunit.external.myOrganization`
• `ap.project.funding.internalCost.send2ugov.forced`
• `ap.prj.descriptionAndGrantNumberConcatenationEnabled`
• `ap.project.cup.send2ugov.enable`
• `ap.prj.externalDmsIdentifier.autogenerated`
• `ap.project.contributor.startDate.required`
• `ap.project.owner.startDate.required`
• `ap.pr.project.grantor.financingQuota.enabled`

```javascript
var Logic = function(wfItem, dtoPj, superLogic, errors, messageUtil, request, gaService, wfService, wfTaskService, searchBuilderTemplate, log) {
this.wfItem=wfItem;
this.dtoPj=dtoPj;
this.superLogic=superLogic;
this.errors=errors;
this.messageUtil=messageUtil;
this.request=request;
this.gaService=gaService;
this.wfService=wfService;
this.wfTaskService=wfTaskService;
this.searchBuilderTemplate=searchBuilderTemplate;
this.log=log;

this.setInternalOrgUnitList = function() {
  var internalOrganizationUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","internalOrganizationUnit", wfService);
  var internalOrganizationUnitSetIterator=internalOrganizationUnitSet.iterator();
  while (internalOrganizationUnitSetIterator.hasNext()){
    var element = internalOrganizationUnitSetIterator.next();
    var orgUnit = element.getOrganizationUnitMap().get("ouId");
    var orgUnitRole = element.getWfDictionaryMap().get("roleId");
    dtoPj.getDTOUoPjs().add(superLogic.getOrganizationUnit(orgUnit, orgUnitRole, errors));
  }
}

this.setFinanciers = function() {
  var sendGrantor2Ugov = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.project.grantor.send2ugov.enable");
  if (sendGrantor2Ugov && sendGrantor2Ugov.trim().equalsIgnoreCase("true")){
    var contractorSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","grantor",wfService);
    var contractorSetIterator = contractorSet.iterator();
    while (contractorSetIterator.hasNext()){
      var element = contractorSetIterator.next();
      var orgUnit = element.getOrganizationUnitMap().get("ouId");
      var orgUnitRole = element.getWfDictionaryMap().get("roleId");
      dtoPj.getDTOEFpjs().add(superLogic.getOrganizationUnit(orgUnit, orgUnitRole, errors));
    }
  }
}
```

this.setPartners = function() {
    var sendPartner2Ugov = Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue("ap.project.partner.send2ugov.enable");
    if (sendPartner2Ugov & & sendPartner2Ugov.trim().equalsIgnoreCase("true")) {
        var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "partner", wfService);
        var partnerSetIterator=partnerSet.iterator();
        while (partnerSetIterator.hasNext()) {
            var element=partnerSetIterator.next();
            var organizationUnit = element.getOrganizationUnitMap().get("partnerId");
            var partnerRole = element.getWfDictionaryMap().get("partnerRole");
            var partecipante = superLogic.getPartecipante(organizationUnit, partnerRole, errors);
            dtoPj.getDTOPartPjs().add(partecipante);
        }
    }
}

this.getContribution = function() {
    return wfItem.getNumberMap().get("internalContribution");
};

this.getCofinancing = function() {
    return wfItem.getNumberMap().get("internalCofinancing");
};

this.getFinancing = function() {
    var coordinatorRoleId = Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getConfigValue("ap.project.partner.role.main");
    var myOrganization = Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getConfigValue("rm.orgunit.external.myOrganization");
    var forcedPjInternalCost = Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getConfigValue("ap.project.financing.internalCost.send2ugov.forced");
    if (forcedPjInternalCost == null)
        forcedPjInternalCost = false;
    else if (forcedPjInternalCost.equalsIgnoreCase("true")) {
        forcedPjInternalCost = true;
    } else {
        forcedPjInternalCost = false;
    }

    if (!forcedPjInternalCost) {
        if (coordinatorRoleId == null || myOrganization == null) {
            throw "Configuration variables ap.project.partner.role.main and rm.orgunit.
external.myOrganization MUST BE DEFINED";
        } else {
            coordinatorRoleId = new Packages.java.lang.Integer(coordinatorRoleId);
            myOrganization = new Packages.java.lang.Integer(myOrganization);
        }
    }
    var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "partner", wfService);
    var partnerSetIterator = partnerSet.iterator();
    while (partnerSetIterator.hasNext()) {
        var element = partnerSetIterator.next();
        var orgUnitRole = element.getWfDictionaryMap().get("partnerRole");
        if (orgUnitRole == null) {
            errors.rejectAndLocate("error.project.partner.missingRole", "partner");
            throw "Detected partner without role";
        } else if (!Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase("false",
confVal)) {
            var orgUnit = element.getOrganizationUnitMap().get("partnerId");
            if (myOrganization.equals(orgUnit.getId()) && coordinatorRoleId.equals
(orgUnitRole.getId())) {  // New line
                var globalCost = wfItem.getNumberMap().get("globalCost");
                if (globalCost == null)
                    globalCost = Packages.java.math.BigDecimal.ZERO;
                var globalContribution = wfItem.getNumberMap().get("globalContribution");
            }
        }
    }
}
if (globalContribution == null)
    globalContribution = Packages.java.math.BigDecimal.ZERO;

var internalCofinancing = wfItem.getNumberMap().get("internalCofinancing");
if (internalCofinancing == null)
    internalCofinancing = Packages.java.math.BigDecimal.ZERO;

var manHoursCofinancing = wfItem.getNumberMap().get("manHoursCofinancing");
if (manHoursCofinancing == null)
    manHoursCofinancing = Packages.java.math.BigDecimal.ZERO;

return globalCost.subtract(globalCost.subtract(globalContribution).subtract(internalCofinancing).subtract(manHoursCofinancing));
StringBuilder.append('-');
StringBuilder.append(wfItem.getDescription());
}
}
descriptionAndGrantNumberConcatenation = stringBuilder.toString();
if(descriptionAndGrantNumberConcatenation == ""){
    return null;
} else {
    var length=descriptionAndGrantNumberConcatenation.length();
    return descriptionAndGrantNumberConcatenation.substring(0, (length>255)?255:length);
}

} else {
    if(wfItem.getDescription() != null){
        var length=wfItem.getDescription().length();
        return wfItem.getDescription().substring(0,(length>255)?255:length);
    } else {
        return null;
    }
}
});
this.getCup = function() {
    var sendCup2Ugov = Packages.it.cilea.core.configuration.util.ConfigurationUtilgetConfigValue("ap.project.cup.send2ugov.enable");
    if (sendCup2Ugov && sendCup2Ugov.trim().equalsIgnoreCase("true")){
        return wfItem.getStringMap().get("cup");
    } else {
        return null;
    }
};
this.getAcronym = function() {
    if(wfItem.getStringMap().get("acronym") != null){
        var length = wfItem.getStringMap().get("acronym").length();
        return wfItem.getStringMap().get("acronym").substring(0,(length>50)?50:length);
    } else {
        return null;
    }
};
this.getExternalDmsIdentifier = function() {
    var externalDmsIdentifierAutogenerated = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.externalDmsIdentifier.autogenerated");
    if("true".equals(externalDmsIdentifierAutogenerated)){
        return wfItem.getIdentifier();
    }else {
        return wfItem.getStringMap().get("externalDmsIdentifier");
    }
};
this.isSynchronizable = function() {
    var legacy = wfItem.getBooleanMap().get("legacy");
    var financing = this.getFinancing();
    if (Packages.java.lang.Boolean.TRUE.equals(legacy) || financing==null || Packages.java.math.BigDecimal.ZERO.equals(financing))
        return Packages.java.lang.Boolean.FALSE;
    else
        return Packages.java.lang.Boolean.TRUE;
};
this.getResearchNature = function() {
    if(wfItem.getWfDictionaryMap().get("researchNature") == null){
        return null;
    } else {
        var researchNatureId = wfItem.getWfDictionaryMap().get("researchNature").getId();
        return (wfService.getWfDictionary(researchNatureId)).getStringMap().get("sourceId");
    }
};
this.getProjectType = function() {
    return superLogic.getProjectType(wfItem);
};

this.getFinanceSchema = function() {
    return this.getProjectType();
};

this.getPjInternalCode = function() {
    return null;
};

this.getGrantNumber = function() {
    if (wfItem.getStringMap().get("grantNumber") == null) {
        return null;
    } else {
        return wfItem.getStringMap().get("grantNumber");
    }
};

this.getManHoursCofinancing = function() {
    return wfItem.getNumberMap().get("manHoursCofinancing");
};

this.getFlPnrr = function() {
    if (Packages.java.lang.Boolean.TRUE.equals(wfItem.getBooleanMap().get("pnrr"))){
        return new Packages.java.lang.Integer("1");
    } else if (Packages.java.lang.Boolean.FALSE.equals(wfItem.getBooleanMap().get("pnrr"))){
        return new Packages.java.lang.Integer("0");
    } else {
        throw "attribute booleanMap[pnrr] MUST BE valorized";
    }
};

this.isAdministrativeOwnerStartDateRequired = function() {
    var administrativeOwnerStartDateRequired=Packages.it.cilea.core.configuration.
util.ConfigurationUtil.getConfigValue("ap.project.administrativeOwner.starDate.required");
    if (administrativeOwnerStartDateRequired){
        if (administrativeOwnerStartDateRequired.trim().equalsIgnoreCase("true")){
            return Packages.java.lang.Boolean.TRUE;
        } else if (administrativeOwnerStartDateRequired.trim().equalsIgnoreCase("false")){
            return Packages.java.lang.Boolean.FALSE;
        } else {
            throw "ap.project.administrativeOwner.starDate.required MUST BE true or false or null";
        }
    } else {
        return Packages.java.lang.Boolean.TRUE;
    }
};

this.isContributorStartDateRequired = function() {
    var contributorStartDateRequired=Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue("ap.project.contributor.starDate.required");
    if (contributorStartDateRequired){
        if (contributorStartDateRequired.trim().equalsIgnoreCase("true")){
            return Packages.java.lang.Boolean.TRUE;
        } else if (contributorStartDateRequired.trim().equalsIgnoreCase("false")){
            return Packages.java.lang.Boolean.FALSE;
        } else {
            throw "ap.project.contributor.starDate.required MUST BE true or false";
        }
    } else {
        return Packages.java.lang.Boolean.TRUE;
    }
};

this.isOwnerStartDateRequired = function() {
    var ownerStartDateRequired = Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue("ap.project.owner.starDate.required");
    if (ownerStartDateRequired){
        if (ownerStartDateRequired.trim().equalsIgnoreCase("true")){
            return Packages.java.lang.Boolean.TRUE;
        } else if (ownerStartDateRequired.trim().equalsIgnoreCase("false")){
            return Packages.java.lang.Boolean.FALSE;
        } else {
            throw "ap.project.owner.starDate.required MUST BE true or false";
        }
    } else {
        return Packages.java.lang.Boolean.TRUE;
    }
}
ethicCommiteeDeleteInconsistentDataProject
Questa validazione controlla che se non sono previste attività di ricerca che prevedono sperimentazioni passibili di giudizio da parte del Comitato Etico e controlla che se non è stata presentata la richiesta di parere all’OPBA o al Comitato Etico di Ateneo di riferimento o al Comitato Etico di un Ente Terzo che ha facoltà di concedere questo tipo di autorizzazioni.
I dati particolari di queste sottosezioni vengono svuotati se viene selezionato NO.
Dal punto di vista del modello dati si tratta degli elementi di tipo ethicCommiteeTopic
Per maggiori dettagli cfr. excel modello dati dell’entità Progetto

```java
//questa funzione elimina tutti i dati del tab, tranne ovviamente ethicCommiteeApplicability
//viene chiamata quando non è applicabile
function deleteAll(){
    //dati riguardanti il topic
    var wfItemElementSet = object.getWfItemElementSet();
    var ethicCommiteeTopicSet= Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "ethicCommiteeTopic", wfService);
    var ethicCommiteeTopicSetIterator=ethicCommiteeTopicSet.iterator();
    while(ethicCommiteeTopicSetIterator.hasNext()){;
        var wfItemElement=ethicCommiteeTopicSetIterator.next();
        var ethicCommiteeTopic= wfItemElement.getWfDictionaryMap().get("ethicCommiteeTopicId");
        var description= ethicCommiteeTopic.getDescription();
        //log.error("Topic description (ethicCommiteeTopicSet): "+description);
        wfService.deleteWfItemElement(wfItemElement.getId());
        //log.error("Eliminato con wfService");
        wfItemElementSet.remove(wfItemElement);
        //log.error("Eliminato dal set");
    }
    object.setWfItemElementSet(wfItemElementSet);
    //log.error("Sostituito set con nuovo");
    //dati riguardanti l’autorizzazione
    //log.error("ethicCommiteeSubmitted: "+object.getBooleanMap().get("ethicCommiteeSubmitted");
    object.getBooleanMap().put("ethicCommiteeSubmitted", null);
    //log.error("ethicCommiteeSubmitDate: "+object.getDateMap().get("ethicCommiteeSubmitDate");
    object.getDateMap().put("ethicCommiteeSubmitDate", null);
    object.getWfDictionaryMap().put("ethicCommiteeAuthorization", null);
    deleteAuthorizationData();
}
```
// questa funzione elimina i dati riguardanti l'autorizzazione
// viene chiamata quando non viene concessa
function deleteAuthorizationData()
{
    //log.error("ethicCommiteeAuthorizationNumber: "+object.getStringMap().get("ethicCommiteeAuthorizationNumber");
    object.getStringMap().put("ethicCommiteeAuthorizationNumber", null);

    //log.error("ethicCommiteeAuthorizationDate: "+object.getDateMap().get("ethicCommiteeAuthorizationDate");
    object.getDateMap().put("ethicCommiteeAuthorizationDate", null);

    var orgUnitMap=object.getOrganizationUnitMap().put("ethicCommiteeAuthorizationOrganization", null);

    wfService.saveOrUpdate(object);
    //log.error("SALVATO!");
}

// quando elimino da una mappa, non mi importa gestire se fosse popolata o meno
// al massimo il remove mi restituisce null
//log.error("Sono nel ethicCommiteeDeleteInconsistentData");

// prima di tutto vado a vedere ethicCommiteeApplicability
// SE NON E' SI (true), allora cancello tutti i dati sottostanti

var ethicCommiteeApplicability = object.getBooleanMap().get("ethicCommiteeApplicability");
if (ethicCommiteeApplicability!=true){
    // chiamo la funzione per eliminare tutto
    deleteAll();
} else{
    // non mi interessa ethicCommiteeSubmitted, deve essere per forza true se sono qui
    // SE E' SI, allora vado a vedere ethicCommiteeAuthorization
    var ethicCommiteeAuthorization= object.getWfDictionaryMap().get("ethicCommiteeAuthorization");

    // SE NON E' SPECIFICATO (null) O 'NON CONCESSO', allora cancello i dati relativi a
    // TRANNE la data di presentazione
    // ci può essere il caso che sia stata fatta la richiesta e si stia aspettando il
    // responso

    // non specificato, caso per evitare eccezioni
    if (ethicCommiteeAuthorization==null){
        deleteAuthorizationData();
    } else{
        var ethicCommiteeAuthorizationDescription= wfService.getWfDictionary (ethicCommiteeAuthorization.getId()).getDescription();
        if (!Packages.org.apache.commons.lang.StringUtils.equals("Concessa", ethicCommiteeAuthorizationDescription)){
            deleteAuthorizationData();
        }
    }
}

ownerValidatorProject
Questa validazione controlla che esista almeno un responsabile scientifico nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo owner
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto
if (object.getId()==null){
    if (object.getPersonMap().get("owner") == null)
        errors.rejectValue("personMap[owner]", "error.project.owner.required");
} else {
    var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
        getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
        "owner", wfService);
    if (!Packages.java.lang.Boolean.TRUE.equals(wfItem.getBooleanMap().get("legacy")) && ownerSet.size() < 1 && object.getPersonMap().get("owner") == null)
        errors.rejectAndLocate("error.project.owner.atLeast1", "owner");
}

**onlyOneOwnerValidatorProject**
Questa validazione controlla che esista un solo responsabile scientifico nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo **owner**
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
"owner", wfService);
if (ownerSet.size() > 1){
    errors.rejectAndLocate("error.project.owner.max1", "owner");
}

**ownerValidatorWithStartEndDateProject**
Questa validazione opera sugli elementi **owner** (responsabile scientifico) figli dell'oggetto radice.
Viene verificato che esista almeno un responsabile scientifico e, se specificato un intervallo di validità per il singolo owner, viene verificato che sia compreso all'interno della validità dell'oggetto padre.
Gli attributi per le date di inizio e fine sono **startDate** e **endDate** sia per l'elemento **owner** che per l'oggetto padre.
if (object.getId()==null){
    if (object.getPersonMap().get("owner")==null)
        errors.rejectValue("personMap[owner]","error.project.owner.required");
} else {
    var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
    "owner", wfService);
    if (!Packages.java.lang.Boolean.TRUE.equals(wfItem.getBooleanMap().get("legacy"))){
        if (ownerSet.size()<1 && object.getPersonMap().get("owner")==null)
            errors.rejectAndLocate("error.project.owner.atLeast1", "owner");
        var ownerSetIterator=ownerSet.iterator();
        var wfItemStartDate=object.getDateMap().get("startDate");
        var wfItemEndDate=object.getDateMap().get("endDate");
        var formatter =  Packages.org.apache.commons.lang.time.FastDateFormat.
        getInstance("dd/MM/yyyy");
        while (ownerSetIterator.hasNext())
        {
            var owner=ownerSetIterator.next();
            var wfItemElementStartDate=owner.getDateMap().get("startDate");
            if( wfItemElementStartDate!= null && wfItemStartDate &&
            (wfItemElementStartDate.before(wfItemStartDate) || (wfItemEndDate!=null && wfItemElementStartDate.after
            (wfItemEndDate))))
            {
                ownerWithWrongInterval.push(owner.getPersonMap().get
                ("ownerId").getDisplayValue());
            }
        }
        if (ownerWithWrongInterval.length>0){
            errors.rejectAndLocate("error.project.owner.wrongInterval",
            "owner", [ownerWithWrongInterval.toString()], "error.project.owner.wrongInterval");
        }
    }
}

dateIntervalOwnerValidatorProject
Questa validazione opera sugli elementi owner (responsabile scientifico) figli dell'oggetto radice.
Gli attributi per le date di inizio e fine sono startDate e endDate sia per l'elemento owner che per l'oggetto padre.
Per i Responsabili scientifici, nella scheda, può essere visibile il Ruolo (visibilità pilotata da configuration ap.prj.owner.role.enabled con default false).
La logica prende in considerazione una configuration (ap.prj.owner.date.interval.enabled) che abilita il controllo per la non sovrapposizione degli intervalli.
La validazione può combinare la verifica degli intervalli con l'eventuale definizione di quali ruoli coinvolgere nel controllo (vedi configuration ap.prj.owner.date.interval.roles).
Configurazioni della ValidateLogic:

- ap.prj.owner.role.enabled
- ap.prj.owner.date.interval.enabled
- ap.prj.owner.date.interval.roles
- ap.project.owner.startDate.required

var roleEnabled = ConfigurationUtil.getConfigValue("ap.prj.owner.role.enabled");
var checkInterval = ConfigurationUtil.getConfigValue("ap.prj.owner.date.interval.
roles");
var checkIntervalRoles = ConfigurationUtil.getConfigValue("ap.prj.owner.date.interval.
roles");
var isOwnerStartDateRequired = ConfigurationUtil.getConfigValue("ap.project.owner.
startDate.required");

var checkIntervalRolesArray = null;
if (checkIntervalRoles != null) {
    checkIntervalRolesArray = checkIntervalRoles.split(",");
ownerRoleSend2UgovValidatorProject

Questa validazione opera sugli elementi owner (responsabile scientifico) figli dell’oggetto radice. Viene verificato che esista almeno un responsabile scientifico.

Gli attributi per le date di inizio e fine sono startDate e endDate sia per l’elemento owner che per l’oggetto padre.

Per i Responsabili scientifici, nella scheda, può essere visibile il Ruolo (visibilità pilotata da configuration ap.prj.owner.role.enabled con default false) (A)

In tal caso la logica prende in considerazione due altre configurazione:
- quali ruoli devono essere sincronizzati con PJ (ap.prj.owner.send2ugov.roles con formato comma separated dei codici di dizionario) (B);
- il numero massimo di Responsabili da sincronizzare (ap.prj.owner.send2ugov.number) (C).

Le possibili combinazioni della configurazione sono (AB) oppure (AC) oppure (ABC).

Configurazioni della ValidateLogic:
var roleEnabled = ConfigurationUtil.getConfigValue("ap.prj.owner.role.enabled");
var rolesSend2UgovArrayConfiguration = ConfigurationUtil.getConfigValue("ap.prj.owner.send2ugov.roles");
var maxToSendConfiguration = ConfigurationUtil.getConfigValue("ap.prj.owner.send2ugov.number");

var rolesArray = null;
var rolesSend2UgovArray = null;
if (rolesSend2UgovArrayConfiguration != null) {
    rolesSend2UgovArray = rolesSend2UgovArrayConfiguration.split(",");
}

var ownerToSend = [];
if (roleEnabled && roleEnabled.equalsIgnoreCase("true") && ((rolesSend2UgovArray != null && rolesSend2UgovArray.length < 1) || (maxToSendConfiguration != null && maxToSendConfiguration == 0))) {
    errors.reject("error.project.ugov.pj.invalidOwnerRoleConfiguration");
    throw "The configurations ap.prj.owner.send2ugov.roles and ap.prj.owner.send2ugov.number MUST BE defined";
}

var ownerSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
if (ownerSet.size() > 0) {
    var ownerElement = null;
    var ownerSetIterator = ownerSet.iterator();
    while (ownerSetIterator.hasNext()) {
        ownerElement = ownerSetIterator.next();
        if (roleEnabled && roleEnabled.equalsIgnoreCase("true")) {
            if (rolesSend2UgovArray != null && rolesSend2UgovArray.length > 0) {
                if (! Packages.java.util.Arrays.asList(rolesSend2UgovArray).contains(ownerElement.getWfDictionaryMap().get("roleId").getInternalSourceIdentifier())) {
                    continue;
                }
            }
        }
        ownerToSend.push(ownerElement);
    }
}
if (maxToSendConfiguration != null && maxToSendConfiguration < ownerToSend.length) {
    errors.rejectAndLocate("error.project.ugov.pj.invalidOwnersNumber", "owner", maxToSendConfiguration.toString(), null);
} else if (ownerToSend.length == 0) {
    errors.rejectAndLocate("error.project.ugov.pj.noOwner", "owner");
}

organizationUnitValidatorProject
Questa validazione controlla che esista almeno una unità organizzativa interna nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo internalOrganizationUnit
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
if (orgUnitSet.size()<1)
    errors.rejectAndLocate("error.project.multipleInternalOrganization.atLeast1", "internalOrganizationUnit");
internalOrganizationUnitCostRequiredValidatorProject
Questa validazione controlla la sezione "Strutture Interne" presente nel tab "Finanziamento" e valuta le configurazioni ap.prj.
internalOrganizationUnitCost.requireOneForEveryInternalOrganizationUnit.enabled e ap.prj.internalOrganizationUnitCost.
notMandatoryIfOnlyOneInternalOrganizationUnit.enabled.
Le strutture selezionabili sono solo quelle presenti nella sezione "Strutture Interne" nel tab "Soggetti interni".

<table>
<thead>
<tr>
<th>requireOneForEveryInternalOrganizationUnit</th>
<th>notMandatoryIfOnlyOneInternalOrganizationUnit</th>
<th>Controllo</th>
</tr>
</thead>
</table>
| true                                      | true                                          | • Se viene inserito solo un elemento internalOrganizationUnit allora non è necessario inserire l'elemento internalOrganizationUnitCost associato perché si presume che tutto il finanziamento vada all'unica struttura  
• Se vengono inseriti più di un elemento internalOrganizationUnit allora è necessario inserire l'elemento internalOrganizationUnitCost associato per ogni internalOrganizationUnit |
| false/non presente                       | false/non presente                            | • È necessario che sia presente almeno un elemento internalOrganizationUnitCost |
| true                                      | false/non presente                            | • È necessario inserire l'elemento internalOrganizationUnitCost associato per ogni internalOrganizationUnit |
| false/non presente                       | true                                          | • Se viene inserito solo un elemento internalOrganizationUnit allora non è necessario inserire l'elemento internalOrganizationUnitCost associato perché si presume che tutto il finanziamento vada all'unica struttura  
• Se vengono inseriti più di un elemento internalOrganizationUnit allora è necessario inserire almeno un elemento internalOrganizationUnitCost |

Dal punto di vista del modello dati si tratta degli elementi di tipo internalOrganizationUnitCost (elemento della sezione "Strutture Interne" presente nel tab "Finanziamento") e internalOrganizationUnit (elemento della sezione "Strutture Interne" presente nel tab "Soggetti interni")
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto
Configurazioni della ValidateLogic:
- ap.prj.internalOrganizationUnitCost.requireOneForEveryInternalOrganizationUnit.enabled
- ap.prj.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit.enabled
var requireOneForEveryInternalOrganizationUnitConf = ConfigurationUtil.getConfigValue("ap.prj.internalOrganizationUnitCost.requireOneForEveryInternalOrganizationUnit.enabled");
var notMandatoryIfOnlyOneInternalOrganizationUnitConf = ConfigurationUtil.getConfigValue("ap.prj.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit.enabled");

var internalOrganizationUnitCostSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnitCost", wfService);
var internalOrganizationUnitSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);

if ((requireOneForEveryInternalOrganizationUnitConf && "true" == requireOneForEveryInternalOrganizationUnitConf) && (notMandatoryIfOnlyOneInternalOrganizationUnitConf && "true" == notMandatoryIfOnlyOneInternalOrganizationUnitConf)) {
    if (internalOrganizationUnitSet.size() > 1) {
        if (internalOrganizationUnitSet.size() != internalOrganizationUnitCostSet.size()) {
            errors.rejectAndLocate("error.project.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit", "internalOrganizationUnitCost");
        }
    } else {
        if (!requireOneForEveryInternalOrganizationUnitConf || "false" == requireOneForEveryInternalOrganizationUnitConf) {
            if (internalOrganizationUnitCostSet.size() == 0) {
                errors.rejectAndLocate("error.project.internalOrganizationUnitCost.atLeast1", "internalOrganizationUnitCost");
            }
        } else {
            if (!requireOneForEveryInternalOrganizationUnitConf || "false" == requireOneForEveryInternalOrganizationUnitConf) {
                if (internalOrganizationUnitSet.size() != internalOrganizationUnitCostSet.size()) {
                    errors.rejectAndLocate("error.project.internalOrganizationUnitCost.requireOneForEveryInternalOrganizationUnit", "internalOrganizationUnitCost");
                }
            }
        }
    }
}

if (internalOrganizationUnitCostSet.size() == 0) {
    errors.rejectAndLocate("error.project.internalOrganizationUnitCost.requireOneForEveryInternalOrganizationUnit", "internalOrganizationUnitCost");
}

ethicCommiteeValidatorProject

Questa validazione controlla la coerenza delle informazioni relative al Comitato etico.

In particolare controlla:

- Che esista almeno un Comitato Etico nel tab “Comitato Etico”
- Che deve essere stata presentata la richiesta di parere all’OPBA o al Comitato Etico di Ateneo di riferimento o al Comitato Etico di un Ente Terzo che ha facoltà di concedere questo tipo di autorizzazioni nel tab “Comitato Etico”
- Che solo SE il campo “Autorizzazione” ha valore “Concessa” allora:
  - Deve essere anche presente il relativo numero identificativo nel tab “Comitato Etico” (altrimenti non verrà richiesto)
  - Deve essere anche presente la relativa data di concessione nel tab “Comitato Etico” (altrimenti non verrà richiesto)
  - Deve essere anche presente l’ente che l’ha concessa nel tab “Comitato Etico” (altrimenti non verrà richiesto)

E’ possibile disattivare i singoli controlli settando a false la relativa variabile di configurazione:
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeTopic.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeSubmitted.enabled

Ricordando che le ultime tre variabili dipendono comunque dal campo "Autorizzazione" che deve avere valore "Concessa":

Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationNumber.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationDate.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationOrganization.enabled

Di default queste variabili non vengono previste, e se non sono presenti o settate con un valore diverso da false, i singoli controlli vengono effettuati. Dal punto di vista del modello dati si tratta degli elementi di tipo ethicCommitteeTopic.
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto
Configurazioni della ValidateLogic:

Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeTopic.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeSubmitted.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationNumber.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationDate.enabled
Ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationOrganization.enabled

```javascript
var ethicCommitteeApplicability = object.getBooleanMap().get("ethicCommitteeApplicability");
if (ethicCommitteeApplicability==true){
    var ethicCommitteeTopicConfigMap = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getLenientBoolean("ap.p.rj.validation.ethicCommitteeValidatorProject.ethicCommitteeTopic.enabled", true);
    if(ethicCommitteeTopicConfigMap){
        var ethicCommitteeTopicSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "ethicCommitteeTopic", wfService);
        if(ethicCommitteeTopicSet.isEmpty()){
            errors.rejectAndLocate("error.project.ethicCommitteeTopic.atLeast1", "ethicCommitteeTopic");
        }
    }
    var ethicCommitteeSubmittedConfigMap = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getLenientBoolean("ap.prj.validation.ethicCommitteeValidatorProject.ethicCommitteeSubmitted.enabled", true);
    if(ethicCommitteeSubmittedConfigMap){
        var ethicCommitteeSubmitted = object.getBooleanMap().get("ethicCommitteeSubmitted");
        if(ethicCommitteeSubmitted!=true){
            errors.rejectValue("booleanMap[ethicCommitteeSubmitted]","error.project.ethicCommitteeSubmitted.required");
        }
    }
    var ethicCommitteeSubmitted = object.getBooleanMap().get("ethicCommitteeSubmitted");
    if(ethicCommitteeSubmitted==true){
        var ethicCommitteeAuthorizationDictionary= object.getWfDictionaryMap().get("ethicCommitteeAuthorization");
        if(ethicCommitteeAuthorizationDictionary!=null){
            var ethicCommitteeAuthorizationDictionaryDescription= wfService.getWfDictionary(ethicCommitteeAuthorizationDictionary.getId()).getDescription();
            if (ethicCommitteeAuthorizationDictionaryDescription.equals("Concessa")){
                var ethicCommitteeAuthorizationNumberConfigMap = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getLenientBoolean("ap.prj.validation.ethicCommitteeValidatorProject.ethicCommitteeAuthorizationNumber.enabled", true);
                if(ethicCommitteeAuthorizationNumberConfigMap){
                    var ethicCommitteeAuthorizationNumber = object.getStringMap().get("ethicCommitteeAuthorizationNumber");
                    if(ethicCommitteeAuthorizationNumber==null){
                        errors.rejectValue("stringMap[ethicCommitteeAuthorizationNumber]","error.project.ethicCommitteeAuthorizationNumber.requiredIfGranted");
                    }
                }
            }
        }
```
**expenditureStartDateAndExpenditureEndDateValidator**  
Questa validazione controlla che la data di ammissibilità delle spese preceda la data di fine validità del progetto.  
Dal punto di vista del modello dati si tratta degli attributi **expenditureStartDate** e **expenditureEndDate**  
Per maggiori dettagli cfr. il modello dati dell'entità in questione.

```java
if(object.getDateMap().get("expenditureEndDate") != null && object.getDateMap().get("expenditureStartDate") != null){
    if(object.getDateMap().get("expenditureStartDate").getTime().after(object.getDateMap().get("expenditureEndDate").getTime())){
        errors.rejectValue("dateMap[expenditureStartDate]","error.date.expenditureStartDateEndDateNotCompatible");
        errors.rejectValue("dateMap[expenditureEndDate]","error.date.expenditureStartDateEndDateNotCompatible");
    }
}
```

**projectEndDateAndExpenditureEndDateValidator**  
Questa validazione controlla che la data di fine ammissibilità delle spese sia successiva alla data di fine del progetto.  
Dal punto di vista del modello dati si tratta degli attributi **endDate** e **expenditureEndDate**.  
Il controllo viene fatto solo per la data di fine ammissibilità delle spese, poiché i pagamenti (la data di inizio ammissibilità delle spese) potrebbero iniziare prima dell'inizio del progetto (la data di inizio del progetto).  
Per maggiori dettagli cfr. il modello dati dell'entità in questione.

```java
if(object.getDateMap().get("expenditureEndDate") != null && object.getDateMap().get("endDate") != null){
    if(object.getDateMap().get("endDate").getTime().after(object.getDateMap().get("expenditureEndDate").getTime())){
        errors.rejectValue("dateMap[endDate]","error.date.projectEndDateAndExpenditureEndDate");
        errors.rejectValue("dateMap[expenditureEndDate]","error.date.projectEndDateAndExpenditureEndDate");
    }
}
```
researchNatureValidatorProject
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera la configurazione ap.prj.researchNature.required
Per tutti gli oggetti che hanno questo attributo valorizzato a true vengono disattivate le validazioni e le eventuali sincronizzazioni.
Configurazioni della ValidateLogic:

- ap.prj.researchNature.required

```java
{
    equalsIgnoreCase("true")
}
```

isKeywordSdgRequired
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera il campo booleanMap[developmentCooperation].
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesto l’inserimento di almeno un SDG - Sustainable Development Goals.

```java
{
    Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get('developmentCooperation'))
}
```

partnerAndGlobalCostEnabled
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le validazioni a cui è agganciata altrimenti no.
Questa regola ritorna true se la configurazione NON è presente o NON è settata a true.
Questa configurazione forza l’invio a Ugov-PJ dell’attributo internalCost.
Configurazioni della ValidateLogic:

- ap.project.financing.internalCost.send2ugov.forced

```java
{
    "true" != StringUtils.trimToEmpty(ConfigurationUtil.getConfigValue("ap.project.financing.internalCost.send2ugov.forced"))
}
```

organizationUnitRoleValidatorProject
Questa validazione controlla che esista almeno una unità organizzativa interna e che una ed una sola struttura sia marcata come principale.
Viene controllato inoltre che la stessa struttura non sia presente più volte (indipendentemente dal ruolo scelto).
Questa validazione si aspetta che sia definita la configurazione ap.prj.internalOrganizationUnit.role.main con l’id numerico interno di IRIS associato al dizionario del ruolo principale.
Dal punto di vista del modello dati si tratta degli elementi di tipo internalOrganizationUnit
Per maggiori dettagli cfr. excel modello dati dell’entità Progetto
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.internalOrganizationUnit.role.main
var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
if (orgUnitSet.size()<1)
    errors.rejectAndLocate("error.project.multipleInternalOrganization.atLeast1", "internalOrganizationUnit");
else {
    internalOrganizationUnitCoordinatorRole=Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.internalOrganizationUnit.role.main");
    if (!internalOrganizationUnitCoordinatorRole)
        throw "Configuration variable ap.prj.internalOrganizationUnit.role.main MUST BE DEFINED";
    else {
        internalOrganizationUnitCoordinatorRole=new Packages.java.lang.Integer(internalOrganizationUnitCoordinatorRole);
    }
}

var orgUnitSetIterator=orgUnitSet.iterator();
var count=0;
var ouCounter=0;
while (orgUnitSetIterator.hasNext()){
    var element=orgUnitSetIterator.next();
    var ouId=element.getOrganizationUnitMap().get("ouId");
    if (ouCounter[""+ouId"])
        errors.rejectAndLocate("error.project.multipleInternalOrganization.duplicatedOrganization", "internalOrganizationUnit");
    ouCounter[""+ouId"]=1;
    var ouRole=element.getWfDictionaryMap().get("roleId");
    if (ouRole!=null && internalOrganizationUnitCoordinatorRole.equals(ouRole.getId()))
        count++;
}
if (count==0)
    errors.rejectAndLocate("error.project.multipleInternalOrganization.missingMain", "internalOrganizationUnit");
else if (count>1)
    errors.rejectAndLocate("error.project.multipleInternalOrganization.tooManyMain", "internalOrganizationUnit");
}

ownerAndOrganizationUnitMatchValidatorProject
Questa validazione controlla che esista per ogni responsabile scientifico del progetto, una unità organizzativa che sia interna nel momento della data di avvio del progetto (startDate) oppure se non c'è questa data in questo momento.
dal punto di vista del modello sono dei dati di tipo: internalOrganizationUnit, owner
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
var ownerDepartmentSet = new Packages.java.util.HashSet();
var internalOrganizationUnitSet = new Packages.java.util.HashSet();
var startDateFromWfItem = wfItem.getDateMap().get("startDate");
if (startDateFromWfItem != null){
    startDateFromWfItem = startDateFromWfItem.getTime();
if (ownerSet.size() > 0) {
    var ownerSetIterator = ownerSet.iterator();
    while (ownerSetIterator.hasNext()) {
        var ownerElement = ownerSetIterator.next();

        var person = ownerElement.getPersonMap().get("ownerId");
        var ownerDepartmentPositionSearch = new Packages.it.cilea.ga.
            command.PositionSearchCommand();
        ownerDepartmentPositionSearch.setPersonId(person.getId());
        ownerDepartmentPositionSearch.setDate(startDateFromWfItem);
        ownerDepartmentPositionSearch.setOrganizationUnitTypeDescription
            ("department");
        ownerDepartmentPositionSearch.setDiscriminator ("research");
        var ownerDepartmentPositionSet = gaService.getPositionSearchList
            (ownerDepartmentPositionSearch, 0);
        var maxPriority = Packages.it.cilea.ga.util.GaUtil.
            getMaxPriority(ownerDepartmentPositionSet, "research", "department");
        var ownerDepartmentSet = ownerDepartmentPositionSet.iterator();
        while (ownerDepartmentSet.hasNext()) {
            var position = ownerDepartmentSet.next();
            if (maxPriority == null || maxPriority.equals(position.
                getPriority())) {
                var ownerDepartmentSet = add(position.
                    getOrganizationUnit());
                break;
            }
        }
    }

    if (ownerDepartmentSet.size() > 0) {
        if (orgUnitSet.size() > 0) {
            var orgUnitSetIterator = orgUnitSet.iterator();
            while (orgUnitSetIterator.hasNext()) {
                var wfItemElement = orgUnitSetIterator.next();
                var orgUnit = wfItemElement.getOrganizationUnitMap().get
                    ("ouId");
                internalOrganizationUnitSet.add(orgUnit);
            }
        }
    }

    if (!internalOrganizationUnitSet.containsAll(ownerDepartmentSet)) {
        var nonMatchedDepartmentCollection = Packages.org.apache.commons.
            collections.CollectionUtils.subtract(ownerDepartmentSet,
                internalOrganizationUnitSet);
        var nonMatchedDepartmentCollectionIterator =
            nonMatchedDepartmentCollection.iterator();
        var nonMatchedDepartmentString = ";
        var counter = 0;
        while (nonMatchedDepartmentCollectionIterator.hasNext()) {
            var nonMatchedDepartment =
                nonMatchedDepartmentCollectionIterator.next();
            if (counter++ > 0)
                nonMatchedDepartmentString += ", ";
            nonMatchedDepartmentString += nonMatchedDepartment.
                getDescription();
        }
        var df = Packages.org.apache.commons.lang.time.FastDateFormat.
            getInstance("dd/MM/yyyy");
        var dateAsString = df.format(startDateFromWfItem);
        errors.rejectAndLocate("error.project.multipleInternalOrganization.
            matchWithDepartmentOwnerNotFound", "internalOrganizationUnit", 
            [dateAsString, nonMatchedDepartmentString], null);
**contributorRoleValidatorProject**
Questa validazione controlla che il ruolo del partecipante venga inserito (se è visibile nel form)
Dal punto di vista del modello dati si tratta dell'attributo `roleId`
Per maggiori dettagli cfr. excel modello dati dell'entità in questione
Configurazioni della ValidateLogic:
- `ap.prj.contributor.role.enabled`

```java
if (Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.contributor.role.enabled").equalsIgnoreCase("true")){
    if(object.getWfDictionaryMap().get("roleId") == null ){
        errors.rejectAndLocate("error.project.contributor.role.required", "contributor");
    }
}
```

**ownerRoleAndDepartmentValidatorProject**
Questa validazione controlla che il ruolo del responsabile (`roleid`) venga sia inserito se la variabile `ap.prj.owner.role.enabled` è settata a true
Questa validazione controlla che la struttura del responsabile (`ouId`) venga sia inserita se la variabile `ap.prj.owner.department.enabled` è settata a true
Per maggiori dettagli cfr. excel modello dati dell'entità in questione
Configurazioni della ValidateLogic:
- `ap.prj.owner.department.enabled`
- `ap.prj.owner.role.enabled`

```java
var wfItemElement=object;
var ownerDepartmentEnabled= Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.owner.department.enabled");
if ("true"==ownerDepartmentEnabled){
    var ouId = wfItemElement.getOrganizationUnitMap().get("ouId");
    if (ouId==null)
        errors.rejectValue("organizationUnitMap[ouId]","error.notNull");
}
var ownerRoleEnabled= Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.owner.role.enabled");
if ("true"==ownerRoleEnabled){
    var roleId = wfItemElement.getWfDictionaryMap().get("roleId");
    if (roleId==null)
        errors.rejectValue("wfDictionaryMap[roleId]","error.notNull");
}
```

**pnrrCallAndProjectInconsistencyValidator**
Questa validazione controlla la scelta effettuata relativa al flag PNRR.
Viene abilitata SOLO se la configurazione `ap.prj.create-by-call.enabled` è valorizzata con true.
Verifica se tale scelta è concorde alla scelta effettuata al Bando agganciato, se dovesse essere discorde verrà sollevata un'eccezione, in tal caso l'utente potrà proseguire SOLAMENTE se sceglie di voler ignorare tale ambiguità.
Dal punto di vista del modello dati si tratta degli attributi `booleanMap[pnrr]` e `booleanMap[pnrrInconsistencyConfirmation]`
Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configurazioni della ValidateLogic:
- `ap.prj.create-by-call.enabled`

```java
```
//In caso di errori non faccio reject, perché vengono gestiti nel validator

grantorValidatorProject

Questa validazione controlla che esista almeno un ente finanziatore nel tab "Dati generali".

Dal punto di vista del modello dati si tratta degli elementi di tipo grantor

Se la variabile di configurazione ap.prj.grantor.withoutGrantorConfirmation.enabled è valorizzata con true viene consentito di procedere senza ente finanziatore dopo aver confermato di volere procedere con dati NON completi.

Questo scenario viene consentito per evitare di bloccare la sottomissione della scheda qualora non venga trovato in anagrafica centralizzata l'ente finanziatore e non si voglia attendere l'inserimento in anagrafica centralizzata.

A posteriori è possibile, per operatori interni di Ateneo, filtrare i progetti marcati come ente finanziatore mancante e procedere alla bonifica dei dati.

Configurazioni della ValidateLogic:

- ap.prj.grantor.withoutGrantorConfirmation.enabled
grantorFinancingQuotaRequiredValidatorProject

Questa validazione controlla viene attivata dalla configurazione `ap.prj.grantor.financingQuota.enabled`.
Verifica che vengano inserite le quote di finanziamento associate a ogni ente e che la loro somma sia corretta.
Dal punto di vista del modello dati si tratta degli elementi di tipo `grantor`.
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

Configurazioni della ValidateLogic:

- `ap.prj.grantor.financingQuota.enabled`
topicValidatorProject
Questa validazione controlla che esista almeno una area tematica
Dal punto di vista del modello dati si tratta degli elementi di tipo topic
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

```javascript
var topicSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "topic", wfService);
if (topicSet.size()<1)
    errors.rejectAndLocate("error.project.topic.atLeast1", "topic");
```

evaluationValidatorProject
Questa validazione controlla che esista almeno una valutazione nel tab "Valutazione".
Dal punto di vista del modello dati si tratta degli elementi di tipo evaluation
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

```javascript
var evaluationSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "evaluation", wfService);
if (evaluationSet.size()<1)
    errors.rejectAndLocate("error.project.evaluation.atLeast1", "evaluation");
```

administrativeOwnerValidatorProject
Questa validazione controlla che esista almeno un referente amministrativo nel tab "Soggetti interni".
Dal punto di vista del modello dati si tratta degli elementi di tipo administrativeOwner
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

```javascript
var ownerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "administrativeOwner", wfService);
if (ownerSet.size()<1)
    errors.rejectAndLocate("error.project.administrativeOwner.atLeast1", "administrativeOwner");
```

currencyAndInternalContributionValidatorProject
Questa validazione controlla che nel tab "Finanziamento", nel caso di selezione di valuta diversa da EURO, vengano specificati

- importo in valuta originaria
- tasso di cambio
- data rilevazione tasso di cambio

Dal punto di vista del modello dati si tratta degli attributi:

- internalContributionOriginalCurrency
- exchangeRate
- exchangeRateDate

Per maggiori dettagli cfr. excel modello dati dell'entità Progetto
```javascript
var currency = object.getWfDictionaryMap().get("currency");
if(currency != null){
    var currencyId = object.getWfDictionaryMap().get("currency").getId();
    var dictionary = wfService.getWfDictionary(currencyId);
    var currencyCode = dictionary.getStringMap().get("code");
    if (currencyCode != "currency.eur"){
        if (!object.getNumberMap().get("internalContributionOriginalCurrency"))
            errors.rejectValue("numberMap[internalContributionOriginalCurrency]","error.project.funding.internalContributionOriginalCurrency.required");
        if (!object.getNumberMap().get("exchangeRate"))
            errors.rejectValue("numberMap[exchangeRate]","error.project.funding.exchangeRate.required");
        if (!object.getDateMap().get("exchangeRateDate"))
            errors.rejectValue("dateMap[exchangeRateDate]","error.project.funding.exchangeRateDate.required");
    }
}
```

checkPartnerValidatorProject
Questa validazione controlla la coerenza delle informazioni inserite nel tab "Partner".
Precisamente viene verificato che:

- sia presente un partner
- ci sia uno ed un solo partner marcato come Coordinatore (ad eccezione del caso in cui l’Ateneo sia Partner Unico)
- nel caso in cui l’Ateneo sia Partner Unico non devono essere presenti altri partner
- l’Ateneo sia presente una ed una sola volta tra i partner
- per ogni partner sia selezionato il ruolo

Non è sempre presente in tutti gli Atenei la voce di dizionario "Partner Unico";
Il Partner Unico (abilitato di default) è gestito dalla variabile di configurazione `ap.prj.partner.role.uniquePartner.enabled`
e il suo valore di dizionario dalla variabile `ap.prj.partner.role.uniquePartner.code`. Di default sono tutti attivi i check elencati.
E’ possibile rilassare alcuni settando a false le variabili indicate di seguito:

- **Deve esistere un Partner marcato come Coordinatore** => `ap.<itemType>.partner.main.required`
- **Devono essere inseriti altri partner se esiste un "Coordinatore"** => `ap.<itemType>.partner.coordinatorAndPartnerRoleCheck.enabled`
- **Non devono esistere altri partner oltre a quello marcato come "Partner Unico"** => `ap.<itemType>.partner.coordinatorAndPartnerRoleCheck.enabled`

Esempio di variabile: `ap.prj.research.int.partner.main.required`.
Nei casi in cui le variabili non ci siano o non siano valorizzate a false il relativo check verrà effettuato.
La variabile `ap.<itemType>.partner.main.required` pilota anche il finanziamento che viene inviato a UGOV. Per maggiori informazione controllare `wfUgovPjSenderValidatorProject`

Questa validazione si aspetta obbligatoriamente che siano definite le seguenti variabili di configurazione:

- `rm.orgunit.external.myOrganization`: identificativo dell’unità organizzativa relativa all’Ateneo con mappatura verso UGOV AC

Questa validazione si aspetta obbligatoriamente che sia definito il seguente dizionario:

- `partnerRole.main`: dizionario relativo al ruolo partner "Coordinatore"

Configurazioni della ValidateLogic:

- `rm.orgunit.external.myOrganization`
- `ap.TIPOLOGIA.partner.main.required`
- `ap.TIPOLOGIA.partner.coordinatorAndPartnerRoleCheck.enabled`
- `ap.prj.partner.role.uniquePartnerAndPartnerRoleCheck.enabled`
- `ap.prj.partner.role.uniquePartner.code`
var coordinatorRoleId = WfUtil.getDictionaryByCode(wfService, "partnerRole.main");
var confUniquePartnerEnabled = ConfigurationUtil.getConfigValue("ap.prj.partner.role.uniquePartner.enabled");
var confUniquePartner = ConfigurationUtil.getConfigValue("ap.prj.partner.role.uniquePartner.code");
var myOrganization = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
if (coordinatorRoleId == null || myOrganization == null)
  throw "Configuration variable rm.orgunit.external.myOrganization and dictionary partnerRole.main MUST BE DEFINED";
else {
  coordinatorRoleId = new Packages.java.lang.Integer(coordinatorRoleId);
  myOrganization = new Packages.java.lang.Integer(myOrganization);
}
if (!confUniquePartnerEnabled || "true" == confUniquePartnerEnabled)
  if (confUniquePartner != null)
    uniquePartnerRoleId = WfUtil.getDictionaryByCode(wfService, confUniquePartner);
    if (uniquePartnerRoleId == null)
      throw "Dictionary " + confUniquePartner + " MUST BE DEFINED";
  else {
    uniquePartnerRoleId = new Packages.java.lang.Integer(uniquePartnerRoleId);
  }
else {
  throw "Configuration variable ap.prj.partner.role.uniquePartner.code MUST BE DEFINED";
}
}

var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "partner", wfService);
var partnerSetIterator = partnerSet.iterator();
var counterOtherPartner = 0;
var counterCoordinator = 0;
var counterUniquePartner = 0;
var counterMyOrganization = 0;
while (partnerSetIterator.hasNext()){
  var find = 0;
  var element = partnerSetIterator.next();
  var orgUnitRole = element.getWfDictionaryMap().get("partnerRole");
  if (orgUnitRole == null)
    errors.rejectAndLocate("error.project.partner.missingRole", "partner");
    break;
  var orgUnit = element.getOrganizationUnitMap().get("partnerId");
  if (orgUnit == null)
    errors.rejectAndLocate("error.project.partner.missingOrgUnit", "partner");
    break;
  if (coordinatorRoleId.equals(orgUnitRole.getId()))
    counterCoordinator++;
    find++;
  } else if (!confUniquePartnerEnabled || "true" == confUniquePartnerEnabled) && confUniquePartner != null{
    if (uniquePartnerRoleId.equals(orgUnitRole.getId()))
      counterUniquePartner++;
      find++;
  }
  if (find == 0){
    counterOtherPartner++;
  }
if(myOrganization.equals(orgUnit.getId())){
    counterMyOrganization++;
}
}
}
if (!errors.hasErrors()){
    if (counterCoordinator > 1){
        errors.rejectAndLocate("error.project.partner.tooManyMain", "partner");
    } else if (counterUniquePartner > 1){
        errors.rejectAndLocate("error.project.partner.tooManyUniquePartner", "partner");
    } else if (counterMyOrganization > 1){
        errors.rejectAndLocate("error.project.partner.tooManyMyOrganization", "partner");
    } else if (partnerSet.size() > 0){
        if (counterMyOrganization == 0){
            errors.rejectAndLocate("error.project.partner.myOrganizationRequired", "partner");
        } else if (counterCoordinator > 0 && counterUniquePartner > 0){
            errors.rejectAndLocate("error.project.partner.coordinatorAndUniquePartner", "partner");
        } else {
            var mainRoleCheck = Packages.it.cilea.wf.util.WfUtil.getParametricConfiguration("ap.<itemType>.partner.main.required", object.getWfItemType(), null);
            if (!StringUtils.equalsIgnoreCase("false", mainRoleCheck)){
                if (counterOtherPartner > 0 && counterCoordinator == 0)
                {
                    errors.rejectAndLocate("error.project.partner.missingMain", "partner");
                }
            }
        }
    var coordinatorAndPartnerRoleCheck = WfUtil.getParametricConfiguration("ap.<itemType>.partner.coordinatorAndPartnerRoleCheck.enabled", object.getWfItemType(), null);
    if (!StringUtils.equalsIgnoreCase("false", coordinatorAndPartnerRoleCheck)){
        if (counterOtherPartner == 0 && counterCoordinator>0) {
            errors.rejectAndLocate("error.project.partner.coordinatorAndNoOtherPartner", "partner");
        }
    }
    var uniquePartnerAndPartnerRoleCheck = WfUtil.getParametricConfiguration("ap.<itemType>.partner.uniquePartnerAndPartnerRoleCheck.enabled", object.getWfItemType(), null);
    if (!StringUtils.equalsIgnoreCase("false", uniquePartnerAndPartnerRoleCheck)){
        if (counterOtherPartner > 0 && counterUniquePartner > 0){
            errors.rejectAndLocate("error.project.partner.otherPartnerAndUniquePartner", "partner");
        }
    }
    } else if(partnerSet.size() == 0){
        errors.rejectAndLocate("error.project.partner.required", "partner");
    }
}

partnerAndRequestedGlobalCostValidatorProject

Nel caso in cui l'Ateneo risulti Coordinatore viene imposto che siano compilati i seguenti attributi nel fieldset "Dati finanziamento richiesto" del tab "Finanziamento" se la configurazione ap.project.financing.requested.enabled è true (default true):

- Contributo globale del Progetto per tutto il partenariato (EURO) - requestedGlobalContribution
- Costo globale del Progetto per tutto il partenariato (EURO) - requestedGlobalCost

Per maggiori dettagli cfr. excel modello dati dell'entità Progetto.
Questa validazione si aspetta che siano definite le seguenti variabili di configurazione:

- rm.orgunit.external.myOrganization: identificativo dell'unità organizzativa relativa all'Ateneo con mappatura verso UGOV AC
Questa validazione si aspetta che sia definito il seguente dizionario:

- partnerRole.main: dizionario relativo al ruolo partner "Coordinatore"

**Configurazioni della ValidateLogic:**

- ap.project.financing.requested.enabled
- rm.orgunit.external.myOrganization
- ap.project.financing.internalCost.send2ugov.forced

```javascript
var financingRequestedEnabled = ConfigurationUtil.getLenientBoolean("ap.project.financing.requested.enabled");
if (financingRequestedEnabled){
    var coordinatorRoleId = WfUtil.getDictionaryByCode(wfService, "partnerRole.main");
    var myOrganization = ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
    if (coordinatorRoleId == null || myOrganization == null){
        throw "Dictionary partnerRole.main and Configuration rm.orgunit.external.myOrganization MUST BE DEFINED";
    } else {
        coordinatorRoleId = new Packages.java.lang.Integer(coordinatorRoleId);
        myOrganization = new Packages.java.lang.Integer(myOrganization);
    }
    var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "partner", wfService);
    var partnerSetIterator=partnerSet.iterator();
    while (partnerSetIterator.hasNext()){var element=partnerSetIterator.next();
        var orgUnitRole = element.getWfDictionaryMap().get("partnerRole");
        var orgUnit = element.getOrganizationUnitMap().get("partnerId");
        if(myOrganization.equals(orgUnit.getId()) && orgUnitRole!=null &&
            coordinatorRoleId.equals(orgUnitRole.getId())){
            if (!object.getNumberMap().get("requestedGlobalContribution"){
                errors.rejectValue("numberMap[requestedGlobalContribution]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.requestedGlobalContribution")], null);
            }
            if (!object.getNumberMap().get("requestedGlobalCost"){
                errors.rejectValue("numberMap[requestedGlobalCost]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.requestedGlobalCost")],
            null);
            }
            if (!object.getWfDictionaryMap().get("requestedCurrency"){
                errors.rejectValue("wfDictionaryMap[requestedCurrency]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.requestedCurrency")],
            null);
            }
        }
    }
}
```
**projectInternalOrganizationUnitRoleValidator**

Validator per controllare che tutti i Dipartimenti abbiano il ruolo valorizzato.
Dal punto di vista del modello dati si tratta degli attributi `internalOrganizationUnit`.
Per maggiori dettagli cfr. modello dati dell'entità in questione

```javascript
var internalOrganizationUnitWfElementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
if(!internalOrganizationUnitWfElementSet.isEmpty()){
  var internalOrganizationUnitWfElementSetIterator =
  internalOrganizationUnitWfElementSet.iterator();
  while(internalOrganizationUnitWfElementSetIterator.hasNext()){
    var internalOrganizationUnitWfElement =
    internalOrganizationUnitWfElementSetIterator.next();
    if(internalOrganizationUnitWfElement.getWfDictionaryMap().get("roleId")
    == null){
      errors.rejectAndLocate("error.project.internalOrganizationUnit.
    roleRequired", "internalOrganizationUnit");
    }
  }
}
```

**partnerAndGlobalCostValidatorProject**

Nel caso in cui l'Ateneo risulti Coordinatore viene imposto che siano compilati i seguenti attributi nel fieldset "Dati finanziamento assegnato" del tab "Finanziamento":

- Contributo globale del Progetto per tutto il partenariato (EURO) - `globalContribution`
- Costo globale del Progetto per tutto il partenariato (EURO) - `globalCost`
- Costo globale del Progetto per tutto il partenariato (NON EURO) - `globalCostOriginalCurrency`
- Contributo globale del Progetto per tutto il partenariato (NON EURO) - `globalContributionOriginalCurrency`

Per maggiori dettagli cfr. excel modello dati dell'entità Progetto.
Questa validazione si aspetta che siano definite le seguenti variabili di configurazione:

- `rm.orgunit.external.myOrganization`: identificativo dell'unità organizzativa relativa all'Ateneo con mappatura verso UGOV AC

Questa validazione si aspetta che sia definito il seguente dizionario:

- `partnerRole.main`: dizionario relativo al ruolo partner "Coordinatore"

**Configurazioni della ValidateLogic:**

- `rm.orgunit.external.myOrganization`
- `ap.project.financing.internalCost.send2ugov.forced`
var coordinatorRoleId = WfUtil.getDictionaryByCode(wfService, "partnerRole.main");
var myOrganization = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
if (coordinatorRoleId == null || myOrganization == null)
    throw "Dictionary partnerRole.main and Configuration rm.orgunit.external.myOrganization MUST BE DEFINED";
else {
    coordinatorRoleId = new Packages.java.lang.Integer(coordinatorRoleId);
    myOrganization = new Packages.java.lang.Integer(myOrganization);
}
var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "partner", wfService);
var partnerSetIterator = partnerSet.iterator();
while (partnerSetIterator.hasNext()){
    var element=partnerSetIterator.next();
    var orgUnitRole = element.getWfDictionaryMap().get("partnerRole");
    var orgUnit = element.getOrganizationUnitMap().get("partnerId");
    if(myOrganization.equals(orgUnit.getId()) && orgUnitRole!=null && coordinatorRoleId.equals(orgUnitRole.getId())){
        if (!object.getNumberMap().get("globalContribution")){ errors.rejectValue("numberMap[globalContribution]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.globalContribution")], null); }
        if (!object.getNumberMap().get("globalCost")){ errors.rejectValue("numberMap[globalCost]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.globalCost")], null); }
        var currency = object.getWfDictionaryMap().get("currency");
        if(currency != null){
            var currencyId = object.getWfDictionaryMap().get("currency").getId();
            var dictionary = wfService.getWfDictionary(currencyId);
            var currencyCode = dictionary.getStringMap().get("code");
            if (currencyCode != "currency.eur"){
                if (!object.getNumberMap().get("globalCostOriginalCurrency")){ errors.rejectValue("numberMap[globalCostOriginalCurrency]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.globalCostOriginalCurrency")], null); }
                if (!object.getNumberMap().get("globalContributionOriginalCurrency")){ errors.rejectValue("numberMap[globalContributionOriginalCurrency]", "error.project.funding.partnerAndGlobalCost.required", [messageUtil.findMessage("label.project.globalContributionOriginalCurrency")], null); }
            }
        }
    }
}
checkInconsistentFundingValidatorProject
Questa validazione controlla la coerenza delle informazioni di finanziamento assegnato sia per le internalOrganizationUnit (unità organizzative interne) che per i partner.
Viene controllato che le somme dei costi, contributi e cofinanziamenti per internalOrganizationUnit e partner coincidano con i totali specificati a livello di progetto.

Vengono valutate le seguenti configurazioni (che hanno tutte default a true)

- checkInconsistentFundingValidatorProject
  
- ap.project.financing.internalOrganizationUnitCost.consistence.assigned.validation.enabled
  Se true viene verificato che la somma degli importi assegnati per singola struttura di Ateneo coincidano con il totale a livello di progetto
  
- ap.project.financing.partnerCost.consistence.assigned.validation.enabled
  Se true viene verificato che la somma degli importi assegnati per singolo partner coincidano con il totale a livello di progetto
  
- ap.project.financing.requested.enabled
  Se true significa che è stato abilitato anche la gestione del finanziamento richiesto globale
  
- ap.project.financing.internalOrganizationUnitCost.requested.enabled
  Se true significa che è stato abilitato anche la gestione del finanziamento richiesto per singola struttura di Ateneo
  
- ap.project.financing.internalOrganizationUnitCost.consistence.requested.validation.enabled
  Se true viene verificato che la somma degli importi richiesti per singola struttura di Ateneo coincidano con il totale a livello di progetto

La validazione viene effettuata solo se anche ap.project.financing.internalOrganizationUnitCost.requested.enabled e ap.project.financing.requested.enabled sono a true

- ap.project.financing.partnerCost.requested.enabled
  Se true significa che è stato abilitato anche la gestione del finanziamento richiesto per singolo partner
  
- ap.project.financing.partnerCost.consistence.requested.validation.enabled
  Se true viene verificato che la somma degli importi richiesti per singolo partner di Ateneo coincidano con il totale a livello di progetto

La validazione viene effettuata solo se anche ap.project.financing.partnerCost.requested.enabled e ap.project.financing.requested.enabled sono a true

Vengono valutate anche le seguenti configurazioni (che hanno tutte default a false)

- ap.prj.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit.enabled
  Se la configurazione è valorizzata a true esiste una sola struttura censita (internalOrganizationUnit) allora la validazione dei totali viene disattivata.

Questo perché si suppone che, nel caso di una singola struttura, tutto il finanziamento venga attribuito a quella struttura.

- ap.prj.partnerCost.notMandatoryIfOnlyOnePartner.enabled
  Se la configurazione è valorizzata a true esiste un solo partner censita (partner) allora la validazione dei totali viene disattivata.

Questo perché si suppone che, nel caso di un singolo partner, tutto il finanziamento venga attribuito a quel partner.

Per maggiori dettagli cfr. excel modello dati dell'entità Progetto.
Configurazioni della ValidateLogic:

- ap.project.financing.requested.enabled
- ap.project.financing.internalOrganizationUnitCost.requested.enabled
- ap.project.financing.internalOrganizationUnitCost.consistence.requested.validation.enabled
- ap.project.financing.internalOrganizationUnitCost.consistence.assigned.validation.enabled
- ap.project.financing.partnerCost.requested.enabled
- ap.project.financing.partnerCost.consistence.requested.validation.enabled
- ap.project.financing.partnerCost.consistence.assigned.validation.enabled
- ap.prj.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit.enabled
- ap.prj.partnerCost.notMandatoryIfOnlyOnePartner.enabled

```javascript
var requestedIouCostValidationList=[
  {
    "totalValueDiscriminator":"requestedInternalCost", "elementDiscriminator":"
    internalOrganizationUnitCost", "elementValueDiscriminator":"ouRequestedCost", "totalCurrencyDiscriminator":"
    requestedCurrency", "totalCurrencyElementValueDiscriminator":"requestedCurrency"},
  {
    "totalValueDiscriminator":"requestedInternalContribution", "elementDiscriminator":"
    internalOrganizationUnitCost", "elementValueDiscriminator":"ouRequestedContribution", "totalCurrencyDiscriminator":"
    requestedCurrency", "totalCurrencyElementValueDiscriminator":"requestedCurrency"},
  {
    "totalValueDiscriminator":"requestedInternalCofinancing", "elementDiscriminator":"
    internalOrganizationUnitCost", "elementValueDiscriminator":"ouRequestedCofinancing", "totalCurrencyDiscriminator":"
    requestedCurrency", "totalCurrencyElementValueDiscriminator":"requestedCurrency"},
  {
    "totalValueDiscriminator":"requestedManHoursCofinancing", "elementDiscriminator":"
    internalOrganizationUnitCost", "elementValueDiscriminator":"ouRequestedManHoursCofinancing", "totalCurrencyDiscriminator":"
    requestedCurrency", "totalCurrencyElementValueDiscriminator":"requestedCurrency"}
];
```
var assignedIouCostValidationList=[
{
"totalValueDiscriminator":"internalCost", "elementDiscriminator":"internalOrganizationUnitCost", "elementValueDiscriminator":"ouCost", "totalCurrencyDiscriminator":"currency", "totalCurrencyElementValueDiscriminator":"currency"},
{
"totalValueDiscriminator":"internalContribution", "elementDiscriminator":"internalOrganizationUnitCost", "elementValueDiscriminator":"ouContribution", "totalCurrencyDiscriminator":"currency", "totalCurrencyElementValueDiscriminator":"currency"},
{
"totalValueDiscriminator":"internalCofinancing", "elementDiscriminator":"internalOrganizationUnitCost", "elementValueDiscriminator":"ouCofinancing", "totalCurrencyDiscriminator":"currency", "totalCurrencyElementValueDiscriminator":"currency"},
{
"totalValueDiscriminator":"manHoursCofinancing", "elementDiscriminator":"internalOrganizationUnitCost", "elementValueDiscriminator":"ouManHoursCofinancing", "totalCurrencyDiscriminator":"currency", "totalCurrencyElementValueDiscriminator":"currency"}
];
var assignedPartnerCostValidationList=[
{
"totalValueDiscriminator":"globalCost", "elementDiscriminator":"partnerCost", "elementValueDiscriminator":"partnerCost", "totalCurrencyDiscriminator":"currency", "totalCurrencyElementValueDiscriminator":"currency"},
{
"totalValueDiscriminator":"globalContribution", "elementDiscriminator":"partnerCost", "elementValueDiscriminator":"partnerContribution", "totalCurrencyDiscriminator":"currency", "totalCurrencyElementValueDiscriminator":"currency"}
];
var allValidations=[];
var globalRequestedEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.requested.enabled");
var iouCostRequestedEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.internalOrganizationUnitCost.requested.enabled");
var iouCostRequestedValidationEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.internalOrganizationUnitCost.consistence.requested.validation.enabled");
var iouCostAssignedValidationEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.internalOrganizationUnitCost.consistence.assigned.validation.enabled");
var partnerCostRequestedEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.partnerCost.requested.enabled");
var partnerCostRequestedValidationEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.partnerCost.consistence.requested.validation.enabled");
var partnerCostAssignedValidationEnabled=ConfigurationUtil.getLenientBoolean("ap.project.financing.partnerCost.consistence.assigned.validation.enabled");
var notMandatoryIfOnlyOneInternalOrganizationUnit = ConfigurationUtil.getLenientBoolean("ap.prj.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit.enabled", false);
var internalOrganizationUnitSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
var isIouValidationNeeded=true;
var isPartnerValidationNeeded=true;
isIouValidationNeeded &= !(notMandatoryIfOnlyOneInternalOrganizationUnit && internalOrganizationUnitSet.size()==1 && internalOrganizationUnitCostSet.size()==0);

isPartnerValidationNeeded &= !(notMandatoryIfOnlyOnePartner && partnerSet.size()==1 && partnerCostSet.size()==0);

if (iouCostAssignedValidationEnabled && isIouValidationNeeded)
    for(var i=0; i<assignedIouCostValidationList.length; i++)
        allValidations.push(assignedIouCostValidationList[i]);

if (partnerCostAssignedValidationEnabled && isPartnerValidationNeeded)
    for(var i=0; i<assignedPartnerCostValidationList.length; i++)
        allValidations.push(assignedPartnerCostValidationList[i]);

if (globalRequestedEnabled && iouCostRequestedEnabled && iouCostRequestedValidationEnabled && isIouValidationNeeded)
    for(var i=0; i<requestedIouCostValidationList.length; i++)
        allValidations.push(requestedIouCostValidationList[i]);

if (globalRequestedEnabled && partnerCostRequestedEnabled && partnerCostAssignedValidationEnabled && isPartnerValidationNeeded)
    for(var i=0; i<requestedPartnerCostValidationList.length; i++)
        allValidations.push(requestedPartnerCostValidationList[i]);

for(var i=0; i<allValidations.length; i++) {
    var validation=allValidations[i];
    WfUtil.totalCheckerRejecter(wfItem, validation.totalValueDiscriminator, validation.elementDiscriminator, validation.elementValueDiscriminator, validation.totalCurrencyDiscriminator, validation.totalCurrencyElementValueDiscriminator, wfService, errors);
}

uniqueIdentifierProject
Questa validazione controlla l'unicità dell'identificativo dell'oggetto.
Questa verifica viene fatta per i casi di generazione di codice "parlante" con possibile collisione con identificativi già presenti.

financingSumProjectValidator
Questa validazione controlla che la somma dei campi Contributo totale Ateneo, Cofinanziamento Ateneo in risorse e Cofinanziamento Ateneo in personale coincida con il valore del Costo totale del progetto per l'Ateneo.
Questa validazione valuta anche le configurazioni ap.prj.grantor.financingQuota.enabled e ap.prj.grantor.financingQuota.validation.enabled.
SOLO nel caso in cui la prima configurazione viene settata a true e la seconda non è presente o è settata a true, viene controllato che la somma delle quote associate agli Enti Finanziatori deve essere uguale al Contributo Totale Ateneo.
IN TUTTI GLI ALTRI CASI questo controllo NON viene effettuato.
Configurazioni della ValidateLogic:
- ap.prj.grantor.financingQuota.enabled
- ap.prj.grantor.financingQuota.validation.enabled
var cost = wfItem.getNumberMap().get("internalCost") != null ? wfItem.getNumberMap().get("internalCost") : BigDecimal.ZERO;
var contribution = wfItem.getNumberMap().get("internalContribution") != null ? wfItem.getNumberMap().get("internalContribution") : BigDecimal.ZERO;
var internalCoFinancing = wfItem.getNumberMap().get("internalCofinancing") != null ? wfItem.getNumberMap().get("internalCofinancing") : BigDecimal.ZERO;
var manHoursCofinancing = wfItem.getNumberMap().get("manHoursCofinancing") != null ? wfItem.getNumberMap().get("manHoursCofinancing") : BigDecimal.ZERO;

if (!cost.stripTrailingZeros().equals(contribution.add(internalCoFinancing).add(manHoursCofinancing).stripTrailingZeros())) {
    errors.rejectValue("numberMap[internalCost]", "error.project.financing.sum");
}

var financingQuotaConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.grantor.financingQuota.enabled");
var financingQuotaValidationConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.grantor.financingQuota.validation.enabled");
if ((financingQuotaConf && financingQuotaConf.trim().equalsIgnoreCase("true")) && (!financingQuotaValidationConf || financingQuotaValidationConf.trim().equalsIgnoreCase("true"))) {
    var grantorSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "grantor", wfService);
    var grantorSetIterator = grantorSet.iterator();
    var sum = BigDecimal.ZERO;
    while (grantorSetIterator.hasNext()) {
        var element = grantorSetIterator.next();
        sum = sum.add(element.getNumberMap().get("financingQuota") != null ? element.getNumberMap().get("financingQuota") : BigDecimal.ZERO);
    }
    if (object.getNumberMap().get("internalContribution") != null) {
        if (!sum.stripTrailingZeros().equals(object.getNumberMap().get("internalContribution").stripTrailingZeros())) {
            errors.rejectAndLocate("error.project.grantor.quota.internalContribution.inconsistent", "grantor");
        }
    }
}

internalOrganizationUnitEditLinkedElementValidator
Questa validazione impedisce la modifica della Struttura in internalOrganizationUnit se tale struttura è presente in internalOrganizationUnitCost (finanziamenti associati a unità organizzative interne) e contiene dei valori compilati
Agisce in caso di edit dei valori del fragment internalOrganizationUnit

WfUtil.deleteElementAssociatedToPreviousOuOfThisOrgUnitElement(object, wfService, gaService, errors);

internalOrganizationUnitDeleteLinkedElementValidator
Questa validazione impedisce l’eliminazione di un elemento in internalOrganizationUnit se la struttura associata è presente in InternalOrganizationUnitCost (finanziamenti associati a unità organizzative interne) e se questo elemento contiene dei valori compilati
Agisce in caso di eliminazione dei valori del fragment internalOrganizationUnit

WfUtil.deleteElementAssociatedToThisOrgUnitElement(object, wfService, gaService, errors);

internalOrganizationUnitCostEditValidator
Questa validazione consente l’inserimento delle sole strutture censite nella sezione "Soggetti interni"
if (!Packages.java.lang.Boolean.TRUE.equals(object.getWfItem().getBooleanMap().get("legacy"))){
    var iouSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object.getWfItem(), "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);
    var iouSetIterator = iouSet.iterator();
    var ouSet = new Packages.java.util.HashSet();
    while (iouSetIterator.hasNext()){
        var element = iouSetIterator.next();
        ouSet.add(element.getOrganizationUnitMap().get("ouId"));
    }
    if (!ouSet.contains(object.getOrganizationUnitMap().get("ouId"))){
        errors.reject("error.project.internalOrganizationUnitCost.editOrAdd");
    }
}

internalOrganizationUnitCostRequestedEditValidator
Questa validazione controlla la presenza degli importi richiesti se le conf ap.project.financing.requested.enabled e ap.project.financing.internalOrganizationUnitCost.requested.enabled sono true
Configurazioni della ValidateLogic:
• ap.project.financing.requested.enabled
• ap.project.financing.internalOrganizationUnitCost.requested.enabled

if (!Packages.java.lang.Boolean.TRUE.equals(object.getWfItem().getBooleanMap().get("legacy"))){
    var globalRequestedEnabled = ConfigurationUtil.getLenientBoolean("ap.project.financing.requested.enabled");
    var iouCostRequestedEnabled = ConfigurationUtil.getLenientBoolean("ap.project.financing.internalOrganizationUnitCost.requested.enabled");
    if (globalRequestedEnabled && iouCostRequestedEnabled) {
        if (object.getWfDictionaryMap().get("requestedCurrency") == null) 
            errors.rejectValue("wfDictionaryMap[requestedCurrency]", "error.notNull");
        if (object.getNumberMap().get("ouRequestedCost") == null) 
            errors.rejectValue("numberMap[ouRequestedCost]", "error.notNull");
        if (object.getNumberMap().get("ouRequestedContribution") == null) 
            errors.rejectValue("numberMap[ouRequestedContribution]", "error.notNull");
        if (object.getNumberMap().get("ouRequestedCofinancing") == null) 
            errors.rejectValue("numberMap[ouRequestedCofinancing]", "error.notNull");
        if (object.getNumberMap().get("ouRequestedManHoursCofinancing") == null) 
            errors.rejectValue("numberMap[ouRequestedManHoursCofinancing]", "error.notNull");
    }
}

partnerEditValidator
Questa validazione impedisce la modifica del Partner in partner se tale Partner è presente in partnerCost (finanziamenti associati a partner).
Agisce in caso di modifica dei valori del fragment.
if (!Packages.java.lang.Boolean.TRUE.equals(object.getWfItem().getBooleanMap().get("legacy"))){
    var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object.getWfItem(), "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "partner", wfService);
    var partnerCostSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object.getWfItem(), "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "partnerCost", wfService);
    var partnerSetIterator = partnerSet.iterator();
    var partnerCostSetIterator = partnerCostSet.iterator();
    var partnerSet = new Packages.java.util.HashSet();
    while (partnerSetIterator.hasNext()){
        var element = partnerSetIterator.next();
        partnerSet.add(element.getOrganizationUnitMap().get("partnerId"));
    }
    var partnerCostSet = new Packages.java.util.HashSet();
    while (partnerCostSetIterator.hasNext()){
        var element = partnerCostSetIterator.next();
        partnerCostSet.add(element.getOrganizationUnitMap().get("partnerId"));
    }
    if(!partnerSet.containsAll(partnerCostSet)){
        errors.reject("error.project.partner.editOrDelete");
    }
}

partnerDeleteValidator
Questa validazione impedisce l'eliminazione di un elemento in partner se il Partner associata è presente in partnerCost (finanziamenti associati a partner). Agisce in caso di eliminazione dei valori del fragment.

if (!Packages.java.lang.Boolean.TRUE.equals(object.getWfItem().getBooleanMap().get("legacy"))){
    var partnerCostSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object.getWfItem(), "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "partnerCost", wfService);
    var partnerCostSetIterator = partnerCostSet.iterator();
    var partnerCostSet = new Packages.java.util.HashSet();
    while (partnerCostSetIterator.hasNext()){
        var element = partnerCostSetIterator.next();
        partnerCostSet.add(element.getOrganizationUnitMap().get("partnerId"));
    }
    if(partnerCostSet.contains(object.getOrganizationUnitMap().get("partnerId"))){
        errors.reject("error.project.partner.editOrDelete");
    }
}

partnerCostEditValidator
Questa validazione impedisce la modifica del Partner in partnerCost (finanziamenti associati a partner) se tale Partner è presente in partner. Agisce in caso di modifica dei valori del fragment.
if (!Packages.java.lang.Boolean.TRUE.equals(object.getWfItem().getBooleanMap().get("legacy"))){
    var partnerSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object.getWfItem(), "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "partner", wfService);
    var partnerSetIterator = partnerSet.iterator();
    var partnerSet = new Packages.java.util.HashSet();
    while (partnerSetIterator.hasNext()){
        var element = partnerSetIterator.next();
        partnerSet.add(element.getOrganizationUnitMap().get("partnerId"));
    }
    if(!partnerSet.contains(object.getOrganizationUnitMap().get("partnerId"))){
        errors.reject("error.project.partnerCost.editOrAdd");
    }
}

notEditableByOwnerValidator
TODO

var reject = 0;
var discriminator = [null, null, "acronym"];
var column = ["wfItemTypeId", "description","string"];
columnLen = column.length;
var value = [];
for (i = 0; i < columnLen; i++) {
    var discriminatorString = discriminator[i] == null ? '' : "." + discriminator[i];
    if(Packages.java.lang.Boolean.TRUE.equals(Packages.it.cilea.wf.util.WfUtil.isAttributeChanged(wfService, wfItem, column[i] + discriminatorString))){
        reject = 1;
        break;
    }
}
if(reject > 0){
    errors.reject("error.project.ownerGeneralData.edit.notAllowed");
}

itemTypeAndTeamEditableValidator
Questa validazione controlla che l'utente corrente NON appartenga al team "Ufficio Erasmus - Progetti con caricamento delle persone abilitate". Questo è un team con agganciata la funzione "Risorsa per Ufficio Erasmus cui negare accesso in modifica ai Progetti" (/ap/project/erasmusTeam/onlyRead.function)
La visione completa consente a chiunque appartenga al team di accedere con i diritti di helpdesk SOLO al flusso per i Progetti di Formazione.
Configurazioni della ValidateLogic:
- ap.project.onlyRead.erasmusItemType
- ap.project.onlyRead.erasmusTeam
itemTypeAndTeamCreatableValidator
Questa validazione controlla che l'utente corrente NON appartenga al team "Ufficio Erasmus - Progetti con caricamento delle persone abilitate". Questo è un team con aganciata la funzione "Risorsa per Ufficio Erasmus cui negare accesso in modifica ai Progetti" (/ap/project/erasmusTeam/onlyRead.function)
La visione completa consente a chiunque appartenga al team di creare una scheda SOLO per il flusso per i Progetti di Formazione.

wfActionLogicGenerateArchiveNumberOnlyOnce
Questa logica effettua la creazione una tantum di un identificativo univoco per il metadato archiveNumber
Configurazioni della ActionLogic:
- ap.prj.archiveNumber.autogenerated.length
- ap.prj.archiveNumber.autogenerated.enabled
function generateArchiveNumber(wfItem) {
    var stringBuilder = new Packages.java.lang.StringBuilder(wfItem.getId().
        toString());
    stringBuilder.append(Packages.java.lang.System.currentTimeMillis());
    var sequence = stringBuilder.toString();
    var configAutogeneratedEnabledLength = Packages.it.cilea.core.configuration.util.
        ConfigurationUtil.getConfigValue("ap.prj.archiveNumber.autogenerated.length");
    var autogeneratedEnabledLength = sequence.length();
    if (configAutogeneratedEnabledLength != null) {
        try {
            if (Integer.parseInt(configAutogeneratedEnabledLength) >= 1 && Integer.parseInt
                (configAutogeneratedEnabledLength) <= 32)
                autogeneratedEnabledLength = Integer.parseInt
                (configAutogeneratedEnabledLength);
        } catch(e) {
            
        }
    }
    sequence = sequence.substring(0, autogeneratedEnabledLength).toUpperCase();
    return sequence;
}

function isArchiveNumberPresent(sequence) {
    var isPresent = true;
    var sql = "select count(*) from AP_ITEM_DATA where AP_ITEM_DATA.
        DISCRIMINATOR='archiveNumber' and AP_ITEM_DATA.STRING_VALUE= " + \\
        "'" + sequence + \\
        "'";
    if (wfService.getCountFromSql(sql) == 0)
        isPresent = false;
    return isPresent;
}

var wfItem = wfService.getWfItem(object.getId());
var configAutogeneratedEnabled = Packages.it.cilea.core.configuration.util.
    ConfigurationUtil.getConfigValue("ap.prj.archiveNumber.autogenerated.enabled");
if (configAutogeneratedEnabled != null && "true".equals(configAutogeneratedEnabled.
    toLowerCase())) {
    if (wfItem.getStringMap().get("archiveNumber") == null) {
        var sequence = "";
        var count = 0;
        do {
            sequence = generateArchiveNumber(wfItem);
            isPresent = isArchiveNumberPresent(sequence);
            count++;
        } while (count <= 10 && isPresent);
        if (isPresent)
            sequence = "NUMERO_ARCHIVIO_NON_VALIDO";
        wfItem.getStringMap().put("archiveNumber", sequence);
        wfService.saveOrUpdate(wfItem);
    }
}

callStartValidator
Questa validazione viene triggerata solo in fase di creazione di un progetto, se la conf 
ap.prj.create-by-call.enabled è valorizzata con true.
In questo caso significa che è stata abilitata per l'utente la doppia scelta
- creazione del progetto specificando il bando
- creazione del progetto specificando il tipo del progetto

Questa scelta viene pilotata tramite il metadato tecnico isCreationByCall
Se l'utente sceglie di creare il progetto a partire dal bando vengono scatenati i seguenti controlli:
- viene verificata l'effettiva selezione di un bando da parte dell'utente
- se la configurazione ap.prj.create-by-call.only-approved-call.enabled è valorizzata a true, viene consentita solo la selezione di bandi in stato ap proved
• viene verificato che il bando selezionato specifichi il tipo del progetto da associare al progetto in fase di creazione. Si tratta del metadato linkedProjectType che deve essere presente nel bando.
   Esempio di un potenziale valore è PRJ.NAZ.PRIN
• se la configurazione ap.prj.create-by-call.grantor.required.enabled è valorizzata a true viene consentita solo la selezione di bandi che specificano l'ente finanziatore promotore da anagrafica degli enti esterni (grantor)

E' necessario anche definire la configurazione ap.prj.create-by-call.default-project-type che deve specificare l'identifier della tipologia di progetto di default nel caso in cui non sia presente tale informazione nel bando (ad esempio PRJ.RESEARCH.INT.HEU)

Al momento attuale, comunque, se l'utente sceglie di creare un progetto a partire dal bando, come spiegato in precedenza, viene SEMPRE verificato che il bando selezionato abbia il tipo di progetto agganciato altrimenti viene bloccato il processo di creazione e di fatto il valore della configurazione ap.prj.create-by-call.default-project-type serve solo per avviare il processo di creazione ma comunque il tipo del progetto effettivo sarà quello del bando

Pertanto basta inserire un tipo di progetto qualsiasi

Configurazioni della ValidateLogic:
• ap.prj.create-by-call.enabled
• ap.prj.create-by-call.only-approved-call.enabled
• ap.prj.create-by-call.grantor.required.enabled
• ap.prj.create-by-call.default-project-type

```
var creationByCallEnabled = ConfigurationUtil.getConfigValue("ap.prj.create-by-call.enabled");
var onlyApprovedCallEnabled = ConfigurationUtil.getConfigValue("ap.prj.create-by-call.only-approved-call.enabled");
var onlyCallWithGrantorEnabled = ConfigurationUtil.getConfigValue("ap.prj.create-by-call.grantor.required.enabled");
var isCreationByCall = wfItem.getBooleanMap().get("isCreationByCall");
if (creationByCallEnabled=="true" && Packages.java.lang.Boolean.TRUE.equals
(isCreationByCall)){
    var callId=wfItem.getIntegerMap().get("callId");
    if (callId==null)
        errors.rejectValue("integerMap[callId]","error.prj.start.callId.
required");
    else {
        var call=wfService.getWfItem(callId);
        if (call==null)
            errors.rejectValue("integerMap[callId]","error.prj.callId.
malformed");
        else {
            if ("true"==onlyApprovedCallEnabled && "approved"!=call.
getWfState().getDescription())
                errors.rejectValue("integerMap[callId]","error.prj.
callId.notApproved");
            else {
                var projectTypeIdentifier=call.getStringMap().get
("linkedProjectType");
                if (projectTypeIdentifier==null)
                    errors.rejectValue("integerMap[callId]","error.
prj.callId.defaultProjectType.required");
                else {
                    var wfItemTypeList=wfService.
getWfItemTypeByIdentifier(projectTypeIdentifier, 0);
                    if (CollectionUtils.isEmpty(wfItemTypeList) ||
wfItemTypeList.size()>1)
                        errors.rejectValue("integerMap[callId]","error.prj.
callId.defaultProjectType.malformed");
                    }
                }
            else {
                var grantorFromCallSet=call.getWfItemElementSet
("grantor");
                if (grantorFromCallSet==null || grantorFromCallSet.size()==0)
                    errors.rejectValue("integerMap[callId]","error.
prj.callId.grantor.required");
                }
            }
        }
    }
```
La logica viene eseguita in fase di creazione di un nuovo progetto se attiva la creazione del progetto a partire dalla selezione del bando.

La logica è abilitata solo se:

- la variabile di configurazione `ap.prj.create-by-call.enabled` è valorizzata a `true`
- l’attributo booleano `isCreationByCall`, specificato in fase di creazione, è valorizzato a `true`

Questa logica effettua il collegamento del Bando al Progetto, aggiunge tutti gli Enti finanziatori del Bando nel Progetto e setta il flag relativo al PNRR (sempre dal Bando al Progetto).

Questa logica va **sempre usata** congiuntamente alla validazione `callStartValidator` (cfr dettaglio delle altre configurazioni disponibili)

**Configurazioni della StartLogic:**

- `ap.prj.create-by-call.enabled`
- `ap.prj.create-by-call.default-project-type`

```java
var isCreationByCall = wfItem.getBooleanMap().get("isCreationByCall");
var creationByCallEnabled = ConfigurationUtil.getConfigValue("ap.prj.create-by-call.enabled");

if (creationByCallEnabled == "true" && Packages.java.lang.Boolean.TRUE.equals(isCreationByCall)) {
    var callId = wfItem.getIntegerMap().get("callId");

    if (callId != null) {
        var call = wfService.getWfItem(callId);
        var projectTypeIdentifier = call.getStringMap().get("linkedProjectType");

        var wfItemTypeList = wfService.getWfItemTypeByentifier(projectTypeIdentifier, 0);

        if (CollectionUtils.isNotEmpty(wfItemTypeList)) {
            var wfItemType = wfItemTypeList.get(0).getId();

            var callProjectLink = new Packages.it.cilea.wf.model.WfItemLink();
            callProjectLink.setDiscriminator("callProjectLink");
            callProjectLink.setParentId(callId);
            callProjectLink.setChildId(wfItem.getId());
            wfService.saveOrUpdate(callProjectLink);
            wfItem.getParentLinkSet().add(callProjectLink);

            var grantorFromCallSet = call.getWfItemElementSet("grantor");
            var grantorFromCallIterator = grantorFromCallSet.iterator();
            while (grantorFromCallIterator.hasNext()) {
                var grantorFromCall = grantorFromCallIterator.next();

                var grantor = new Packages.it.cilea.wf.model.WfItemElement();
                grantor.setDiscriminator("grantor");
                grantor.setWfItemId(wfItem.getId());
                grantor.getOrganizationUnitMap().put("grantorId", grantorFromCall.getOrganizationUnitMap().get("grantorId");

                wfService.saveOrUpdate(grantor);
                wfItem.getWfItemElementSet().add(grantor);
            }

            if (call.getBooleanMap().get("pnrr") != null) {
                wfItem.getBooleanMap().put("pnrr", call.getBooleanMap().get("pnrr");
            }
        }
    }

    wfItem.getIntegerMap().put("callId", null);
    wfItem.getBooleanMap().put("isCreationByCall", null);
    true;
```

Questa logica viene eseguita in fase di creazione di un nuovo Progetto/Bando.

La logica verifica se nel Progetto/Bando è specificato che vengono utilizzati fondi PNRR.

- Se è già specificato (Sì/No) non viene effettuata alcuna modifica
• Se NON è già specificato (Si/No) viene valutata la configurazione `ap.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.pnrr.enabled`, se è presente e settata a `true` viene specificato che per quel Progetto/Bando vengono utilizzati fondi PNRR, in tutti gli altri casi verrà specificato che per quel Progetto/Bando NON vengono utilizzati fondi PNRR

Configurazioni della StartLogic:
• `ap.TIPOLOGIA.pnrr.enabled`

```java
var pnrr = wfItem.getBooleanMap().get("pnrr");
if (pnrr == null){
    wfItem.getBooleanMap().put("pnrr", false);
    var pnrrConfiguration = Packages.it.cilea.wf.util.WfUtil.getParametricConfiguration("ap.<wfItemType>.pnrr.enabled", wfItem.getWfItemType(), null);
    if(pnrrConfiguration != null){
        pnrrConfiguration = pnrrConfiguration.trim();
        if(!pnrrConfiguration.isEmpty()){ 
            if(Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase(pnrrConfiguration, "true")){
                wfItem.getBooleanMap().put("pnrr", true);
            }
        }
    }
}
```

**projectCallValidator**
Questa validazione viene eseguita solo e solamente se sono state abilitate le validazioni e l’invio verso UGOV-PJ.
Questa validazione controlla che esista uno ed un solo **bando di finanziamento** associato a questo progetto nel tab "Dati generali".
Viene verificato inoltre che il bando di finanziamento sia in stato **approvato**
Se viene rilevata una riga anomala, cioè con un Bando vuoto, segnala l’errore
Dal punto di vista del modello dati si tratta degli elementi di tipo **callProjectLink**
Per maggiori dettagli cfr. excel modello dati dell’entità Progetto

```java
var linkSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "callProjectLink", wfService);
if(linkSet != null){
    if (linkSet.size()>1){ 
        errors.rejectAndLocate("error.project.call.moreThanOne", "callProjectLink");
    } else if (linkSet.size()==1){
        var linkSetIterator = linkSet.iterator();
        if( linkSetIterator.hasNext() ){
            var linkElement = linkSetIterator.next();
            var call = linkElement.getParent();
            if(call != null){
                var callState = call.getWfState().getDescription();
                if (!Packages.org.apache.commons.lang.StringUtils.equals("approved", callState)){
                    errors.rejectAndLocate("error.project.call.notApproved", "callProjectLink");
                }else{
                    errors.rejectAndLocate("error.project.call.emptyRow", "callProjectLink");
                }
            }
        }
    }
}
```

**callValidatorProject**
Questa validazione controlla che esista almeno un bando nel tab "Dati generali".
Dal punto di vista del modello dati si tratta degli elementi di tipo **callProjectLink**
Se la variabile di configurazione `ap.prj.call.withoutCallConfirmation.enabled` è valorizzata con true viene consentito di procedere senza bando dopo
avere confermato di volere procedere con dati NON completi. Questo scenario viene consentito per evitare di bloccare la sottomissione della scheda qualora non venga trovato in anagrafica centralizzata il bando. A posteriori è possibile, per operatori interni di Ateneo, filtrare i progetti marcati come bando mancante e procedere alla bonifica dei dati.

**Configurazioni della ValidateLogic:**

- `ap.prj.call.withoutCallConfirmation.enabled`

```javascript
var callProjectLinkSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "callProjectLink", wfService);
var withoutCallConfirmation = ConfigurationUtil.getConfigValue("ap.prj.call.withoutCallConfirmation.enabled");
if (callProjectLinkSet.size()<=1) {
  if (withoutCallConfirmation && withoutCallConfirmation == "true") {
    if (object.getBooleanMap().get("withoutCallConfirmation") == null || Packages.java.lang.Boolean.FALSE.equals(object.getBooleanMap().get("withoutCallConfirmation"))) {
      errors.rejectAndLocate("error.project.call.required", "callProjectLink");
    } else {
      errors.rejectAndLocate("error.project.call.required", "callProjectLink");
    }
  } else {
    errors.rejectAndLocate("error.project.call.required", "callProjectLink");
  }
}
if (callProjectLinkSet.size()>=1) {
  var callProjectLinkSetIterator = callProjectLinkSet.iterator();
  while (callProjectLinkSetIterator.hasNext()) {
    var callProjectLink = callProjectLinkSetIterator.next();
    var call = callProjectLink.getParent();
    if (call == null)
      errors.rejectAndLocate("error.project.call.emptyRow", "callProjectLink");
  }
  if (withoutCallConfirmation && withoutCallConfirmation == "true") {
    if (Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get("withoutCallConfirmation"))) {
      errors.rejectAndLocate("error.project.call.withoutCallConfirmationAndCallMismatch", "callProjectLink");
    } else {
      errors.rejectValue("booleanMap[withoutCallConfirmation]", "error.project.call.withoutCallConfirmation");
    }
  }
}
```

**isAlphaNumeric**

Verifica che il campo `isScoreNumeric` sia a TRUE

```javascript
!object.getBooleanMap().get('isScoreNumeric').booleanValue();
```

**wfStartLogicPublicEngagement**

Questa logica viene eseguita in fase di creazione di un nuovo oggetto. Effettua il setup dei seguenti attributi con i valori tra parentesi

- periodicEvent (false)
- institutionalInitiative (false)
- evaluationEnable (false)
- externalFinancing (0)
- totalBudget (0)
- numTotalDay (1)

Per maggiori dettagli cfr. modello dati dell'entità publicEngagement
if(wfItem.getBooleanMap().get("periodicEvent") == null) {
    wfItem.getBooleanMap().put("periodicEvent", false);
}
if(wfItem.getBooleanMap().get("institutionalInitiative") == null) {
    wfItem.getBooleanMap().put("institutionalInitiative", false);
}
if(wfItem.getBooleanMap().get("evaluationEnable") == null) {
    wfItem.getBooleanMap().put("evaluationEnable", false);
}
if(wfItem.getNumberMap().get("externalFinancing") == null) {
    wfItem.getNumberMap().put("externalFinancing", new java.math.BigDecimal("0"));
}
if(wfItem.getNumberMap().get("totalBudget") == null) {
    wfItem.getNumberMap().put("totalBudget", new java.math.BigDecimal("0"));
}
if(wfItem.getNumberMap().get("numTotalDay") == null) {
    wfItem.getNumberMap().put("numTotalDay", new java.math.BigDecimal("1"));
}
if(wfItem.getBooleanMap().get("personOrDepartment") != null) {
    wfItem.getBooleanMap().put("personOrDepartment", null);
}
true;

wfStartLogicAcademicAreaFromOwner
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
Recupera l’Area Accademica del Responsabile Scientifico (owner) ed estrae la sua Area Accademica, aggiungendo quindi un nuovo elemento "Aree scientifiche coinvolte" (scientificAreasInvolved).
La data alla quale estrarre l’Area Accademica del responsabile è quella contenuta nel metadato con il nome recuperabile dalla configurazione ap.<tipologia>.owner.position.date.
Se questa configurazione

- NON è presente, viene usato la data inserita nel metadato startDate
- è presente ed ha valore uguale a CURRENT viene usata la data corrente
- è presente ed ha valore uguale a CURRENT_OR_LAST viene usata l’ultima Area Accademica disponibile (attiva o cessata)
- è presente ed NON ha un valore tra quelli elencati allora viene usata la data presente nel metadato con il nome specificato

È possibile decidere se settare in automatico il metadato startDate per il responsabile.

Se la configurazione ap.<tipologia>.owner.startDate.enabled:

- NON è presente, allora NON viene abilitato l’automatismo di recupero della startDate
- è uguale a false allora NON viene abilitato l’automatismo di recupero della startDate
- è uguale a true allora viene inserita come startDate lo stesso valore recuperato con le logiche descritte al punto prima
- altrimenti NON viene abilitato l’automatismo di recupero della startDate

È inoltre possibile stabilire quali siano i contesti all’interno dei quali cercare l’Area Accademica per il responsabile scientifico.
Questo viene pilotato dalla configurazione ap.<tipologia>.owner.position.context.
Se questa configurazione

- NON è presente, viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a research allora viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a support allora viene usato il solo contesto di supporto (TA)
- è uguale a ALL allora vengono usati entrambi i contesti

Con <tipologia> viene inteso il codice a tre lettere in minuscolo che indica la tipologia radice dell’oggetto in questione come ad esempio: lab, eqp, inm, prj,
Per maggiori dettagli cfr. modello dati dell’entità publicEngagement
Configurazioni della StartLogic:

- ap.TIPOLOGIA.owner.position.date
- ap.TIPOLOGIA.owner.startDate.enabled
- ap.TIPOLOGIA.owner.position.context

//logica di start che va a popolare l'academicArea (scientificAreasInvolved Aree scientifiche coinvolte)
//con l’area ministeriale estratta dall’owner in start
// lavora in coppia con la logica di save per gli owner (?)
// devo usare per forza il metodo lungo perché l'owner è già stato spostato negli element
var ownerElementSet = wfItem.getWfItemElementSet("owner");
var ownerElementSetIterator = ownerElementSet.iterator();
while(ownerElementSetIterator.hasNext()){
    var ownerElement = ownerElementSetIterator.next();
    var person = gaService.getPerson(ownerElement.getPersonMap().get("ownerId").getId());
    var checkDate = WfUtil.getCheckDate(wfItem, "owner", wfService);
    var positionContext = WfUtil.getPositionContext(wfItem, wfService);
    var positionSet = WfUtil.getPositionSet(person, checkDate, positionContext, "academicArea", gaService);
    // var maxPriority = Packages.it.cilea.ga.util.GaUtil.getMaxPriority
    // (personPositionSet, positionContext, "academicArea");
    var academicAreaOrgUnit = GaUtil.getPriorityOrganizationUnit(personPositionSet, positionContext, "academicArea");
    var personPositionSetIterator = personPositionSet.iterator();
    if(academicAreaOrgUnit != null){
        // log.error("FOUND ONE!");
        var saiElement = new Packages.it.cilea.wf.model.WfItemElement();
        saiElement.setDiscriminator("scientificAreasInvolved");
        saiElement.setWfItemId(wfItem.getId());
        saiElement.getOrganizationUnitMap().put("ouId", academicAreaOrgUnit);
        wfService.saveOrUpdate(saiElement);
        wfItem.getWfItemElementSet().add(saiElement);
    }
}
true;

validatorDepartmentPublicEngagement
Questa validazione verifica in creazione di una nuova iniziativa, se la configurazione "ap.pen.owner.internalOrganizationUnit.mandatory" è settata a true, che la persona abbia un ruolo di ricerca e afferisca ad un dipartimento.
Il valore di default è true.
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

Configurazioni della ValidateLogic:

• ap.TIPOLOGIA.owner.internalOrganizationUnit.mandatory

var personOrDepartment = wfItem.getBooleanMap().get("personOrDepartment");
if (personOrDepartment=true){
    var person=wfItem.getPersonMap().get("owner");
    if(person!=null){
        var mandatory=Packages.it.cilea.wf.util.WfUtil.getMandatory(object,"owner");
        if(mandatory){
            person=gaService.getPerson(person.getId());
            var startDate=wfItem.getDateMap().get("startDate");
            if(startDate!=null){
                // setto il positionSearch in modo da trovare se ho delle position di ricerca per quella data
                var ownerPositionSearch = new Packages.it.cilea.ga.command.PositionSearchCommand();
                ownerPositionSearch.setPersonId(person.getId());
                ownerPositionSearch.setOrganizationUnitTypeDescription("department");
                ownerPositionSearch.setDiscriminator("research");
            }
        }
    }
}
var ownerPositionSet = gaService.getPositionSearchList(ownerPositionSearch, 0);
if(ownerPositionSet.isEmpty()){
    errors.reject("error.publicEngagement.start.organizationUnitMap.required");
} else {
    errors.reject("error.startDate.required");
}
}

ercKeywordValidatorPublicEngagement
Validator per controllare presenza di almeno una classificazione ERC
Dal punto di vista del modello dati si tratta degli attributi ercKeyword
Per maggiori dettagli cfr. modello dati dell'entità in questione

var ercKeywordWfElementSet=FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "ercKeyword", wfService);
if(ercKeywordWfElementSet.isEmpty()){
    errors.rejectAndLocate("error.keywordErc.required", "ercKeyword");
}

ownerStartValidatorPublicEngagement
Questa validazione controlla che esista almeno un Responsabile scientifico e Dipartimento obbligatorio in fase di creazione.
Per maggiori dettagli cfr. modello dati dell'entità

if (object.getId()==null){
    var personOrDepartment =object.getBooleanMap().get("personOrDepartment");
    if  ((personOrDepartment==true)&&(object.getPersonMap().get("owner")==null)){
        errors.rejectValue("personMap[owner]","error.publicEngagement.owner.required");
    }else if ((personOrDepartment==false)&&(object.getOrganizationUnitMap().get("ouId")==null)){
        errors.rejectValue("organizationUnitMap[ouId]","error.publicEngagement.organizationUnitMap.required");
    }
}

checkToolEvaluation
Questa validazione controlla che, se l'attributo evaluationEnable è true, siano presenti gli strumenti di valutazione obbligatori.
Dal punto di vista del modello dati si tratta degli elementi evaluationTool
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

var isToolEnable=object.getBooleanMap().get("evaluationEnable");
var listEvaluation=Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "evaluationTool", wfService);
if ((isToolEnable==true)&&(listEvaluation.size()==0)){
    errors.rejectAndLocate("error.publicEngagement.toolEvaluationEngagement.required", "evaluationTool");
}

suardDepartmentHelpdeskInitiativeValidatorPublicEngagement
Validazione per SUARD (non attiva)
var suardUniversityInitiative =wfItem.getBooleanMap().get("suardUniversityInitiative");
if (suardUniversityInitiative==false){
    var actualDipartment =object.getOrganizationUnitMap().get
    ("suardDepartmentInitiative");
    if(actualDipartment==null){
        errors.reject("error.publicEngagement.organizationUnitMap.required");
    }else {
        var wfItem= wfService.getWfItem(object.getWfItemId());
        var dipartementList=Packages.it.cilea.core.fragment.util.FragmentUtil.
        internalOrganizationUnit*, wfService);
        var departmentWFElementSetIterator=dipartementList.iterator();
        var isPresent=false;
        while(departmentWFElementSetIterator.hasNext()){}
            var department=departmentWFElementSetIterator.next();
            var departmentOwner =department.getOrganizationUnitMap().get
            ("ouId");
                if(actualDipartment.getId().equals(departmentOwner.getId())){
                    isPresent=true;
                    break;
                }
        }
        if(false==isPresent){
            errors.reject("error.publicEngagement.notallowed.
    suardDepartmentInitiative");
        }
    }
}

suardDepartmentInitiativeValidatorPublicEngagement
Viene controllato che siano presenti nella Public Engagement almeno un dipartimento relativo all'iniziativa SUARD.
Dal punto di vista del modello dati si tratta degli elementi internalOrganizationUnit
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

var suardUniversityInitiative =wfItem.getBooleanMap().get("suardUniversityInitiative");
if (suardUniversityInitiative==false){
    var actualDipartment =object.getOrganizationUnitMap().get
    ("suardDepartmentInitiative");
    if(actualDipartment==null){
        errors.reject("error.publicEngagement.organizationUnitMap.required");
    }else {
        var wfItem= wfService.getWfItem(object.getWfItemId());
        var dipartementList=Packages.it.cilea.core.fragment.util.FragmentUtil.
        internalOrganizationUnit*, wfService);
        var departmentWFElementSetIterator=dipartementList.iterator();
        var isPresent=false;
        while(departmentWFElementSetIterator.hasNext()){}
            var department=departmentWFElementSetIterator.next();
            var departmentOwner =department.getOrganizationUnitMap().get
            ("ouId");
                if(actualDipartment.getId().equals(departmentOwner.getId())){
                    isPresent=true;
                    break;
                }
        }
        if(false==isPresent){
            errors.reject("error.publicEngagement.notallowed.
    suardDepartmentInitiative");
        }
    }
}

else { errors.reject("error.publicEngagement.notallowed");}
deleteValidatorParentDetectorPublicEngagement
Questa validazione impedisce l'eliminazione dell'item nel caso in cui sia collegato ad altri publicEngagement (sia loro 'figlio').
Dal punto di vista del modello dati si tratta degli elementi publicEngagementLink
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

var wfItem = wfService.getWfItem(object.getWfItemId());
var identifier = wfItem.getIdentifier();
if(!wfItem.getChildWfItemLinkSet("publicEngagementLink").isEmpty())
  errors.reject("error.publicEngagement.childDetected");

deleteValidatorChildDetectorPublicEngagement
Questa validazione impedisce l'eliminazione dell'item nel caso in cui sia collegato ad altri publicEngagement (sia loro 'padre')
Dal punto di vista del modello dati si tratta degli elementi publicEngagementLink
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

var wfItem = wfService.getWfItem(object.getWfItemId());
var identifier = wfItem.getIdentifier();
if(!wfItem.getParentWfItemLinkSet("publicEngagementLink").isEmpty())
  errors.reject("error.publicEngagement.parentDetected");

periodicityValidatorPublicEngagement
Questa validazione verifica che, se l'evento è periodico (periodicEvent settato a true), sia presente la periodicità (periodicity).
Dal punto di vista del modello dati si tratta degli attributi command.booleanMap[periodicEvent] e command.wfDictionaryMap[periodicity]
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

var wfItem = wfService.getWfItem(object.getWfItemId());
var periodicEvent = wfItem.getBooleanMap().get("periodicEvent");
if (periodicEvent != null & periodicEvent == true){
  if(wfItem.getWfDictionaryMap().get("periodicity") == null)
    errors.rejectValue("wfDictionaryMap[periodicity]","error.
    publicEngagement.periodicEvent.periodicityNotIndicated");
}

scientificAreasInvolvedValidatorPublicEngagement
Questa validazione controlla che esista almeno un'Area scientifica coinvolta nella sezione "Aree scientifiche coinvolte" nel tab "Classificazioni".
Dal punto di vista del modello dati si tratta degli elementi scientificAreasInvolved
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

var scientificAreasInvolvedSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
  getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
  "scientificAreasInvolved", wfService);
if(scientificAreasInvolvedSet.isEmpty()){
  errors.rejectAnd Locate("error.publicEngagement.scientificAreasInvolved.atleast1", "scientificAreasInvolved");
}

impactIndicatorsValidatorPublicEngagement
Questa validazione controlla che non venga caricata la stessa tipologia di impactIndicators più volte
Dal punto di vista del modello dati si tratta degli elementi impactIndicators
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement
**publicInvolvedDuplicatedValidatorPublicEngagement**

Questa validazione controlla che non venga caricata la stessa tipologia di publicInvolved più volte.

Dal punto di vista del modello dati si tratta degli elementi `publicInvolved`

Per maggiori dettagli cfr. modello dati dell’entità `publicEngagement`

```java
publicInvolvedDuplicatedValidatorPublicEngagement
var wfItemElement = object;
var wfItem = wfService.getWfItem(wfItemElement.getWfItemId());
var publicInvolvedType = wfItemElement.getWfDictionaryMap().get("dictionary");
if(publicInvolvedType != null){
    var publicInvolvedSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "publicInvolved", wfService);
    var publicInvolvedSetIterator = publicInvolvedSet.iterator();
    while(publicInvolvedSetIterator.hasNext()){var publicInvolvedElement = publicInvolvedSetIterator.next();
        //controllo per non confrontare lo stesso oggetto
        if(!wfItemElement.getId() .equals( publicInvolvedElement.getId())){
            var dictionary = publicInvolvedElement.getWfDictionaryMap().get("dictionary");
            if(publicInvolvedType.getId() .equals( dictionary.getId())){
                errors.rejectAndLocate("error.publicEngagement.publicInvolved.typeAlreadyPresent", "publicInvolved");
            }
        }
    }
}
```

**financingTotalBudgetValidator**

Questa validazione verifica che il totale del Budget complessivo sia maggiore o uguale al totale dei Finanziamenti esterni.

Per maggiori dettagli cfr. il modello dati dell’entità in questione.

```java
financingTotalBudgetValidator
var totalBudget = wfItem.getNumberMap().get("totalBudget");
var externalFinancing = wfItem.getNumberMap().get("externalFinancing");
if(totalBudget != null && externalFinancing != null) {
    if(totalBudget < externalFinancing) {
        errors.rejectValue("numberMap[totalBudget]", "error.publicEngagement.totalBudget.lower.externalFinancing");
    }
}
```
**ownerTimeReportingValidator**
Vengono valutate le configurazioni `ap.pen.manDay.enabled` e `ap.pen.manHour.enabled`.

- **Configurazione manDay settata a true e configurazione manHour settata a true**
  Viene richiesto che almeno uno dei due attributi manDay o manHour sia compilato
- **Configurazione manDay settata a true e configurazione manHour settata a false o non presente**
  Viene richiesto che il attributo manDay sia compilato
- **Configurazione manDay settata a false o non presente e configurazione manHour settata a true**
  Viene richiesto che l'attributo manHour sia compilato
- **Configurazione manDay settata a false o non presente e configurazione manHour settata a false o non presente**
  Viene sollevato un messaggio di errore in cui viene indicato di settare almeno una delle due configurazioni a true

Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

**Configurazioni della ValidateLogic:**

- `ap.pen.manDay.enabled`
- `ap.pen.manHour.enabled`

```javascript
var wfElementOwnerSetTemp = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
var wfElementOwnerSetIteratorTemp = wfElementOwnerSetTemp.iterator();
var manDayConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.pen.manDay.enabled");
var manHourConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.pen.manHour.enabled");

while (wfElementOwnerSetIteratorTemp.hasNext()) {
    var wfElementTemp = wfElementOwnerSetIteratorTemp.next();
    if ((manDayConf && manDayConf.trim().equalsIgnoreCase("true")) && (manHourConf && manHourConf.trim().equalsIgnoreCase("true"))) {
        if (wfElementTemp.getIntegerMap().get("manDay") == null && wfElementTemp.getIntegerMap().get("manHour") == null) {
            errors.rejectAndLocate("error.publicEngagement.owner.manDayOrManHour.required", "owner", [wfElementTemp.getPersonMap().get("ownerId").getDisplayValue()], null);
        }
    }
    if ((manDayConf && manDayConf.trim().equalsIgnoreCase("true")) || !manHourConf.trim().equalsIgnoreCase("true")) {
        if (wfElementTemp.getIntegerMap().get("manDay") == null) {
            errors.rejectAndLocate("error.publicEngagement.owner.manDay", "owner", [wfElementTemp.getPersonMap().get("ownerId").getDisplayValue()], null);
        }
    }
    if (!manDayConf || manDayConf.trim().equalsIgnoreCase("false") && (manHourConf && manHourConf.trim().equalsIgnoreCase("true"))) {
        if (wfElementTemp.getIntegerMap().get("manHour") == null) {
            errors.rejectAndLocate("error.publicEngagement.owner.manHour", "owner", [wfElementTemp.getPersonMap().get("ownerId").getDisplayValue()], null);
        }
    }
    if (!manDayConf || manDayConf.trim().equalsIgnoreCase("false") && !manHourConf || manHourConf.trim().equalsIgnoreCase("false")) {
        errors.reject("error.publicEngagement.owner.manDayOrManHour.requiredConf");
    }
}
```

**ownerElementTimeReportingValidator**
Vengono valutate le configurazioni `ap.pen.manDay.enabled` e `ap.pen.manHour.enabled`.

- **Configurazione manDay settata a true e configurazione manHour settata a true**
  Viene richiesto che almeno uno dei due attributi manDay o manHour sia compilato
- **Configurazione manDay settata a true e configurazione manHour settata a false o non presente**
  Viene richiesto che l'attributo manDay sia compilato
- Configurazione manDay settata a false o non presente e configurazione manHour settata a true
  Viene richiesto che l'attributo manHour sia compilato
- Configurazione manDay settata a false o non presente e configurazione manHour settata a false o non presente
  Viene sollevato un messaggio di errore in cui viene indicato di settare almeno una delle due configurazioni a true

Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

Configurazioni della ValidateLogic:
- ap.pen.manDay.enabled
- ap.pen.manHour.enabled

```javascript
var wfItemElement = object;
var manDayConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.pen.manDay.enabled");
var manHourConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.pen.manHour.enabled");
if (manDayConf && manDayConf.trim().equalsIgnoreCase("true") && manHourConf && manHourConf.trim().equalsIgnoreCase("true")){
    if(wfItemElement.getIntegerMap().get("manDay") == null && wfItemElement.getIntegerMap().get("manHour") == null){
        errors.reject("error.publicEngagement.ownerElement.manDayOrManHour.required");
    }
}
if (manDayConf && manDayConf.trim().equalsIgnoreCase("true")) && (!manHourConf || manHourConf.trim().equalsIgnoreCase("false")){
    if(wfItemElement.getIntegerMap().get("manDay") == null){
        errors.rejectValue("integerMap[manDay]","error.publicEngagement.ownerElement.manDayOrManHour.required");
    }
}
if (!manDayConf || manDayConf.trim().equalsIgnoreCase("false")) && (manHourConf && manHourConf.trim().equalsIgnoreCase("true")){
    if(wfItemElement.getIntegerMap().get("manHour") == null){
        errors.rejectValue("integerMap[manHour]","error.publicEngagement.ownerElement.manDayOrManHour.required");
    }
}
if (!manDayConf || manDayConf.trim().equalsIgnoreCase("false")) && (!manHourConf || manHourConf.trim().equalsIgnoreCase("false")){
    errors.reject("error.publicEngagement.owner.manDayOrManHour.requiredConf");
}

specificOtherValuePopulatedValidatorPublicEngagement
Questa validazione controlla che venga popolata il campo per specificare l'attributo specificOtherValue nel caso venga selezionata la tipologia Altro (publicInvolvedPublicEngagement.other) per il dizionario publicInvolvedType.
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

```
**publicInvolvedAtLeastOneValidatorPublicEngagement**
Questa validazione controlla che esista almeno un elemento **publicInvolved**, se la tipologia dell'iniziativa è "PEN.FORMAZIONE-COMUNICAZIONE" ("Giornate organizzate di formazione alla comunicazione - rivolta a PTA o docenti")
Dal punto di vista del modello dati si tratta degli elementi **publicInvolved**
Per maggiori dettagli cfr. il modello modello dati dell'entità publicEngagement

```java
var itemTypeIdentifier = object.getWfItemType().getIdentifier();
if (Packages.org.apache.commons.lang.StringUtils.equals("PEN.FORMAZIONE-COMUNICAZIONE",
    itemTypeIdentifier)) {
    var publicInvolvedSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
generateCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
"publicInvolved", wfService);
    if (publicInvolvedSet.isEmpty()) {
        errors.rejectAndLocate("error.publicEngagement.publicInvolved.required",
"publicInvolved");
    } else {
        var publicInvolvedSetIterator = publicInvolvedSet.iterator();
        while (publicInvolvedSetIterator.hasNext()) {
            var publicInvolved = publicInvolvedSetIterator.next();
            var dictionary = publicInvolved.getWfDictionaryMap().get("dictionary");
            var code = dictionary.getStringMap().get("code");
            if (new String("publicInvolvedPublicEngagement.other-professors").toString() != new
String(code).toString() && new String("publicInvolvedPublicEngagement.other-administratives").toString() != new
String(code).toString() && new String("publicInvolvedPublicEngagement.other-unstructored-research-staff").toString() != new
String(code).toString() && new String("publicInvolvedPublicEngagement.other-students-phd").toString() != new
String(code).toString()) {
                errors.rejectAndLocate("error.publicEngagement.publicInvolved.consistent",
"publicInvolved");
            }
        }
    }
}
```

**publicInvolvedValidator**
Questa validazione verifica che esista almeno un Pubblico coinvolto (publicInvolved)
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

```java
var publicInvolvedWfElementSet=FragmentUtil.
generateCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","publicInvolved",
wfService);
if(publicInvolvedWfElementSet.isEmpty()){ errors.rejectAndLocate("error.publicEngagement.publicInvolved.required",
"publicInvolved");
}
```

**addScientificAreaInvolvedFromNewOwnerValidatorPublicEngagement**
Questa "validazione" va a popolare la sezione "Aree scientifiche coinvolte" nel tab "Classificazioni" ogni volta che viene aggiunto un nuovo owner.
Dal punto di vista del modello dati si tratta degli elementi **scientificAreasInvolved**
Per maggiori dettagli cfr. il modello modello dati dell'entità publicEngagement

```java
if(!errors.hasErrors()){
    //inutile, ma mi serve per ricordarmi il contesto
    var wfItemElement =
    object;
    var wfItem = wfService.getWfItem(wfItemElement.getWfItemId());
    var fragmentInfo = Packages.it.cilea.core.fragment.util.FragmentUtil.
generateFragmentableInfo(wfItemElement);
    var newOwnerToAdd = wfItemElement.getPersonMap().get(fragmentInfo.
    getDiscriminator()+"Id");
    var person = gaService.getPerson(newOwnerToAdd.getPersonId());
```
var positionLastSet = person.getPositionLastSet();
var maxPriority = Packages.it.cilea.ga.util.GaUtil.getMaxPriority(positionLastSet, "research", "academicArea");
positionLastSetIterator = positionLastSet.iterator();

while (positionLastSetIterator.hasNext()){
    var position = positionLastSetIterator.next();
    if (maxPriority == null || maxPriority.equals(position.getPriority())) {
        if ("research".equals(position.getDiscriminator())
            & "academicArea".equals(position.getOrganizationUnit().
            getOrganizationUnitType().getDescription())) {
            //log.error("FOUND ONE!");
            //ora che ho trovato l'academicArea, vado a vedere se
            non è già presente
            scientifcAreasInvolvedElementSet = wfItem.
            getWfItemElementSet("scientificAreasInvolved");
            scientificAreasInvolvedElementSetIterator =
            scientificAreasInvolvedElementSet.iterator();
            var alreadyAdded = false;
            while(scientificAreasInvolvedElementSetIterator.
            hasNext()){  

                scientificAreasInvolvedElementSetIterator.next();
                var orgUnit = scientificAreasInvolvedElement.
                getOrganizationUnitMap().get("ouId");
                getOrganizationUnitId());
                if{orgUnit.getOrganizationUnitId() == position.
                getOrganizationUnitId()}{
                    alreadyAdded = true;
                }
            }
            //se alla fine non l'ho trovata, allora la inserisco
            if(!alreadyAdded){
                var saiElement = new Packages.it.cilea.wf.model.
                WfItemElement();
                //"scientificAreasInvolved";
                saiElement.setDiscriminator(position.getOrganizationUnit());
                saiElement.setWfItemId(wfItem.getId());
                saiElement.getOrganizationUnitMap().put("ouId",
                wfservice.saveOrUpdate(saiElement);    
                wfItem.getWfItemElementSet().add(saiElement);
                //UTILIZZARE QUESTO Packages.it.cilea.core.
                break;
            }
        }
    }
}

transitionCommentRequired
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Prevede l'inserimento obbligatorio della motivazione quando l'iniziativa viene inviata nello stato "In validazione" a partire dallo stato "Riaperto".
Configurazioni della ValidateLogic:

- ap.publicEngagement-validation-flow.owner.strictValidation.enable
var penState = object.getWfState().getDescription();
var ownerStrictValidation = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.publicEngagement-validation-flow.owner.strictValidation.enable");
(ownerStrictValidation == "true" && Packages.org.apache.commons.lang.StringUtils.equals("reopened", penState))

secondaryCategoryDifferFromType
Questa validazione va a controllare che tra le Categorie Secondarie inserite non sia presente anche la tipologia del PublicEngagement.

var secondaryCategoryWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","secondaryCategory",wfService);

if(!secondaryCategoryWfElementSet.isEmpty()){
  var wfItemTypeIdentifier = object.getWfItemType().getIdentifier();
  var secondaryCategorySetIterator = secondaryCategoryWfElementSet.iterator();
  while (secondaryCategorySetIterator.hasNext()){
    var secondaryCategoryWfItemElement = secondaryCategorySetIterator.next();
    var secondaryCategoryWfDictionary = secondaryCategoryWfItemElement.getWfDictionaryMap().get("dictionary");
    var sourceId = secondaryCategoryWfDictionary.getStringMap().get("sourceId");
    if(Packages.org.apache.commons.lang.StringUtils.equals(sourceId, wfItemTypeIdentifier))
      errors.rejectAndLocate("error.publicEngagement.secondaryCategory.mustDifferFromType", "secondaryCategory");
  }
}

wfIdentityLogicMultipleOwnerPublicEngagementStatesJs
Questa è una state logic che costruisce dinamicamente gli stati successivi raggiungibili dallo stato "reopened" per i publicEngagement per i responsabili (owner).
Se la variabile di configurazione ap.publicEngagement-validation-flow.owner.strictValidation.enable è valorizzata a true allora viene applicato il flusso con validazione strict.
Questo vuol dire che il responsabile, se riapre un item, dovrà necessariamente ripassare per lo step di validazione (submitted) in carico agli organi dipartimentali o all'ufficio centrale.
Se la variabile invece non è definita o è settata ad un valore diverso allora verrà applicato il flusso di validazione lenient ovvero a seguito di riapertura NON sarà necessario nuovo step di validazione.
Configurazioni della ValidateLogic:
  * ap.publicEngagement-validation-flow.owner.strictValidation.enable

var wfTask = object;
var wfItem = wfService.getWfItem(wfTask.getWfItemId());

var ownerStrictValidationEnabledParam = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.publicEngagement-validation-flow.owner.strictValidation.enable");
var ownerStrictValidationEnabled="true"==ownerStrictValidationEnabledParam)?true:false;
if (ownerStrictValidationEnabled)
  wfTask.setWfNextStates("submitted");
else {
  wfTask.setWfNextStates("approved");
}
Questa è una permission logic che costruisce dinamicamente i permessi per i publicEngagement per i team dipartimentali (headOfDepartment). Assegna i diritti di validazione ai referenti dei dipartimenti censiti. Nel caso di più dipartimenti vengono assegnati questi diritti al primo dipartimento che specifica uno di questi ruoli (nell'ordine specificato):

- **RM - Public Engagement - Ruolo dipartimento (Coordinatore/Organizzatore - ouRolePublicEngagement.manager)**
- **RM - Public Engagement - Ruolo dipartimento (Partecipante - ouRolePublicEngagement.participant)**
- **RM - Public Engagement - Ruolo dipartimento (Afferenza del responsabile/partecipante - ouRolePublicEngagement.afferent)**

```javascript
var wfTask = object;
var wfItem = wfService.getWfItem(wfTask.getWfItemId());
var permissions = "r";
var isOnlyRead = true;

var internalOrganizationUnitWfItemElementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "internalOrganizationUnit", wfService);

//log.error("xxx got ouSet");
//recupero l'unità organizzativa associata (dipartimento) all'identity/team di questo task
var organizationUnitFromIdentity = Packages.it.cilea.ga.util.GaUtil.getOrganizationUnitFromIdentityForDepartment(wfTask.getIdentityId(), gaService, gaAuthorizationService);

//log.error("xxx got organizationUnitFromIdentity: " + organizationUnitFromIdentity.getDescription());

var isManagerPresent = Packages.it.cilea.wf.util.WfUtil.isRoleIdPresent(wfService, "roleId", "ouRolePublicEngagement.manager", internalOrganizationUnitWfItemElementSet);

var hasOrgUnitPowerPermissions = false;
if(Packages.java.lang.Boolean.FALSE.equals(isManagerPresent)){
    var isPartecipantPresent = Packages.it.cilea.wf.util.WfUtil.isRoleIdPresent(wfService, "roleId", "ouRolePublicEngagement.participant", internalOrganizationUnitWfItemElementSet);

    if(Packages.java.lang.Boolean.FALSE.equals(isPartecipantPresent)){
        var isAfferentPresent = Packages.it.cilea.wf.util.WfUtil.isRoleIdPresent(wfService, "roleId", "ouRolePublicEngagement.afferent", internalOrganizationUnitWfItemElementSet);

        if(Packages.java.lang.Boolean.FALSE.equals(isAfferentPresent)){
            //log.error("xxx nothing is present");
            //altri ruoli non previsti
            hasOrgUnitPowerPermissions = false;
        }else{
            //log.error("xxx isAfferentPresent");
            hasOrgUnitPowerPermissions = Packages.it.cilea.wf.util.WfUtil.hasOrgUnitPowerPermissions(wfService, "ouId", organizationUnitFromIdentity.getId(), "roleId", "ouRolePublicEngagement.afferent", internalOrganizationUnitWfItemElementSet);
        }
    }else{
        //log.error("xxx isPartecipantPresent");
        hasOrgUnitPowerPermissions = Packages.it.cilea.wf.util.WfUtil.hasOrgUnitPowerPermissions(wfService, "ouId", organizationUnitFromIdentity.getId(), "roleId", "ouRolePublicEngagement.participant", internalOrganizationUnitWfItemElementSet);
    }
}else{
    //log.error("xxx isManagerPresent");
}
```

hasOrgUnitPowerPermissions = Packages.it.cilea.wf.util.WfUtil.hasOrgUnitPowerPermissions(wfService, "ouId", organizationUnitFromIdentity.getId(), "roleId", "ouRolePublicEngagement.manager", internalOrganizationUnitWfItemElementSet);

//log.error("xxx hasOrgUnitPowerPermissions: " + hasOrgUnitPowerPermissions);

/*
draft        crwfd
approved      rf
submitted     rwf
reopened      rwfd
rejected      rfd
preapproval   r
concluded     r
*/

switch("" + String(wfItem.getWfState().getDescription())){
case "draft":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="crwfd";
    else
        permissions="r";
    break;
case "approved":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="rf";
    else
        permissions="r";
    break;
case "submitted":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="rwf";
    else
        permissions="r";
    break;
case "reopened":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="rwfd";
    else
        permissions="r";
    break;
case "rejected":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="rfd";
    else
        permissions="r";
    break;
    /*
case "preapproval":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="rwf";
    else
        permissions="r";
    break;
case "concluded":
    if(Packages.java.lang.Boolean.TRUE.equals(hasOrgUnitPowerPermissions))
        permissions="rf";
    else
        permissions="r";
    break;
    */
default:
    permissions="r";
L'unità organizzativa associata all'Ateneo viene estratta dalla variabile rm.orgunit.external.myOrganization. Viene infine associato di default a questo partner il ruolo principale (researchCentrePartnerRole.main-seat).

Per maggiori dettagli cfr. modello dati dell'entità in questione.

**Configurazioni della StartLogic:**
- ap.rsc.interuniversity.rootType
- rm.orgunit.external.myOrganization

```java
// se la tipologia selezionata è un qualsiasi figlio di "Centri di ricerca interuniversitari"
// inserire tra i partner l'ateneo con il ruolo sede principale
var parentWfItemTypeSet = wfItem.getWfItemType().getParentWfItemTypeSet();
if (parentWfItemTypeSet != null && !parentWfItemTypeSet.isEmpty()) {
  var parentWfItemType = parentWfItemTypeSet.iterator().next();
  while (parentWfItemTypeSet.iterator().hasNext()) {
    var parentWfItemType = parentWfItemTypeSetIterator.next();
    var interuniversityRootTypeConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.rsc.interuniversity.rootType");
    if (Packages.org.apache.commons.lang.StringUtils.equals(interuniversityRootTypeConf, parentWfItemType.getIdentifier())) {
      var externalOrgUnitRootId = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
      if (externalOrgUnitRootId != null) {
        var externalOrgUnitRoot = gsService.getOrganizationUnit(externalOrgUnitRootId);
        var partnerElement = new Packages.it.cilea.wf.model.WfItemElement();
        partnerElement.setDiscriminator("partner");
        partnerElement.setWfItemId(wfItem.getId());
        partnerElement.getOrganizationUnitMap().put("partnerId", externalOrgUnitRoot);
        var researchCentrePartnerRoleWfDictionaryList = wfService.getWfDictionaryList("researchCentrePartnerRole", null, null);
        var researchCentrePartnerRoleWfDictionaryListIterator = researchCentrePartnerRoleWfDictionaryList.iterator();
        while (researchCentrePartnerRoleWfDictionaryListIterator.hasNext()) {
          var researchCentrePartnerRoleWfDictionary = researchCentrePartnerRoleWfDictionaryListIterator.next();
          //...
researchCentreRenewalValidator
Questa validazione esegue tutta una serie di controlli validi ai fini del Rinnovo del Centro di ricerca/Piattaforma tecnologica, di seguito i controlli:

- Verifica che sia specificato al massimo un collegamento come padre del rinnovo
- Verifica che il collegamento del rinnovo abbia la data di fine settatata e che questa sia minore della attuale data di inizio del corrente centro/piattaforma
- Verifica che il collegamento del rinnovo non abbia già altri collegamenti (figli) come rinnovi

Questa validazione viene eseguita SOLO se la configurazione **ap.rsc.renewal.enabled** è settata a true.
Dal punto di vista del modello dati si tratta degli elementi di tipo **renewalLink**.
Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configurazioni della ValidateLogic:
- **ap.rsc.renewal.enabled**

```java
if(object.getDateMap().get("startDate") != null && Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase("true",ConfigurationUtil.getConfigValue("ap.rsc.renewal.enabled"))){
    var objectLinkSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "renewalLink", wfService);
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("divType", "parent");
    if(objectLinkSet != null){
        if (objectLinkSet.size()>1){
            errors.rejectAndLocate("error.rsc.renewalLink.moreThanOne", "renewalLink", paramMap);
        } else if (objectLinkSet.size()==1){
            var objectLinkSetIterator = objectLinkSet.iterator();
            if(objectLinkSetIterator.hasNext()){
                var objectLink = objectLinkSetIterator.next();
                var itemParentObject = objectLink.getParent();
                if(itemParentObject != null){
                    if(itemParentObject.getDateMap().get("endDate") == null){
                        errors.rejectAndLocate("error.rsc.renewalLink.parent.endDate.notNull", "renewalLink", paramMap);
                    }
                }
            }
        }
    }
    true;
```
researchCentrePartnerValidator

Validator per controllare:
- se la tipologia selezionata è un qualsiasi figlio della tipologia estratta dalla variabile `ap.rsc.interuniversity.rootType`, la presenza dell'Ateneo inserito tra i Partner con un ruolo settato; se diverso da "Sede principale" deve essere compilato il campo Referenti esterni
- se la tipologia selezionata è un qualsiasi figlio della tipologia estratta dalla variabile `ap.rsc.interuniversity.rootType`, la presenza di almeno un altro Partner inserito con un ruolo diverso da "Sede principale"
- la presenza di un solo Partner marcato con ruolo "Sede principale"

Dal punto di vista del modello dati si tratta degli attributi `partner`
Per maggiori dettagli cfr. modello dati dell'entità in questione

Configurazioni della ValidateLogic:
- `ap.rsc.interuniversity.rootType`
- `rm.orgunit.external.myOrganization`

```java
else if (itemParentObject.getDateMap().get("endDate").compareTo(object.getDateMap().get("startDate")) >= 0){
    errors.rejectAndLocate("error.rsc.renewalLink.parent.endDate.inconsistent", "renewalLink", paramMap);
} else {
    var childParentObjectLinkSet = itemParentObject.getChildWfItemLinkSet();
    childParentObjectLinkSetIterator = childParentObjectLinkSet.iterator();
    while(childParentObjectLinkSetIterator.hasNext()){
        var childParentObjectLink = childParentObjectLinkSetIterator.next();
        var itemChildParentObject = childParentObjectLink.getChild();
        if(childParentObjectLink.getDiscriminator().equals('renewalLink') && !itemChildParentObject.getId().equals(object.getId())){
            errors.rejectAndLocate("error.rsc.renewalLink.parent.tooManyChild", "renewalLink", paramMap);
            break;
        }
    }
} else {
    errors.rejectAndLocate("error.rsc.renewalLink.emptyRow", "renewalLink", paramMap);
}
```
researchCentreApprovalOrgUnitValidator

This validation checks the approvals inserted, looking for those with attached structures. In this case, it verifies that all structures are listed in the section Strutture.

From a data model perspective, it deals with the elements of type approval and internalOrganizationUnit.

For more details, see the model of the entity in question.

```javascript
function check(approvalOrgUnitId) {
  var orgUnitSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
      "internalOrganizationUnit", wfService);
  var orgUnitSetIterator=orgUnitSet.iterator();
  while (orgUnitSetIterator.hasNext()){
    var orgUnitElement=orgUnitSetIterator.next();
    if(orgUnitElement.getOrganizationUnitMap().get("ouId").getId() == approvalOrgUnitId){
      return true;
    }
  }
}
```
```java
return false;
}
var errorList = new Packages.java.util.HashSet();
var approvalSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
gCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "approval", wfService);
var approvalSetIterator=approvalSet.iterator();
while (approvalSetIterator.hasNext()){;
var approvalElement=approvalSetIterator.next();
if(approvalElement.getOrganizationUnitMap().get("ouId") != null){
if(!check(approvalElement.getOrganizationUnitMap().get("ouId").getId())){
    errorList.add(approvalElement.getOrganizationUnitMap().get("ouId"));
}
}
}
if(errorList.size() > 0){
    errors.rejectAndLocate("error.rsc.approval.orgUnit.notAllowed", "approval",
[WfUtil.getSelectableListAsHtmlList(new Packages.java.util.ArrayList(errorList))], null);
}
```

**spinoffWFActionLogicOwnerSummary**
Questa logica serve per il conteggio degli elementi di tipo owner.
Viene eseguita ogni volta che viene congelata una versione della Spin-off, eseguendo il conteggio degli elementi per ogni tipologia della Compagine Sociale.
Per ogni tipologia con ALMENO un elemento viene creato il relativo elemento di tipo ownerSummary. Per maggiori dettagli cfr. modello dati dell'entità in oggetto

```java
var itemMostValidateId = wfService.getQueryUniqueResult("select item.
mostValidatedItemId from WfItem item where item.id = " + object.getId());
var wfItemCurrentMostValidated = wfService.getWfItem(itemMostValidateId);
var ownerSummaryLinkSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(wfItemCurrentMostValidated, "it.cilea.wf.model.WfItemElement", "ownerSummary-snapshot", wfService);
if (ownerSummaryLinkSet.isEmpty()) {
    var ownerLinkSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(wfItemCurrentMostValidated, "it.cilea.wf.model.WfItemElement", "owner", wfService);
    if (!ownerLinkSet.isEmpty()) {
        var ownerSetIterator = ownerLinkSet.iterator();
        var ownerSummaryMap = {};
        var versionDate;
        var itemLinkId = wfService.getQueryUniqueResult("select itemLink.id from
WfItemLink itemLink where itemLink.child = " + wfItemCurrentMostValidated.getId() + " and itemLink.
discriminator like 'version'");
        var itemLink = wfService.getWfItemLink(itemLinkId);
        if(itemLink != null) {
            versionDate = itemLink.getDateMap().get("versionDate");
        }
        if (versionDate != null) {
            while (ownerSetIterator.hasNext()) {
                var owner = ownerSetIterator.next();
                if (owner.getDictionary().get("roleId") != null & owner.getDateMap().get("startDate") <= 0 & owner.getDateMap().get("endDate") == null | owner.
getDateTimeMap().get("endDate").compareTo(versionDate) >= 0) {
                    var ownerRoleIdCode = wfService.getWfDictionary(owner.
getWfDictionaryMap().get("roleId").getId()).getInternalSourceIdentifier();
                    if (ownerSummaryMap[ownerRoleIdCode] != null) {
                        ownerSummaryMap[ownerRoleIdCode] = (ownerSummaryMap
[ownerRoleIdCode] + 1);
                    } else {
                        ownerSummaryMap[ownerRoleIdCode] = 1;
                    }
                }
            }
        for (var key in ownerSummaryMap) {
            var ownerSummaryElement = new WfItemElement();
```
spinoffWfActionLogicBalancePreviousYear
Questa logica si occupa di gestire il fatturato e utile degli anni precedenti.
Viene eseguita ogni volta che viene congelata una versione della Spin-off, creando un riepilogo del fatturato e utile dei 3 anni precedenti (se presenti).
Viene creato un elemento sia per il fatturato che per l’utile per ognuno dei 3 anni precedenti alla data di riferimento del monitoraggio, nel caso ci siano più fatturati/utili di riferimento per un singolo anno viene preso l’elemento con la data più recente.
NON viene creato l’elemento se per quel determinato anno NON c’è alcun fatturato/utile.
Viene creato l’elemento SE per quel determinato anno di riferimento è presente un fatturato/utile contenente nella data di fine (attributo dateMap[endDate]) l’anno in questione.
Per maggiori dettagli cfr. modello dati dell’entità in oggetto.

```javascript
function saveNewElement(wfItemCurrentMostValidated, balanceElementSet, valueKey){
  var elementSnapshotLinkSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItemCurrentMostValidated, "it.cilea.wf.model.WfItemElement", valueKey+"PreviousYear-snapshot", wfService);
  if (elementSnapshotLinkSet.isEmpty()) {
    if (!balanceElementSet.isEmpty()) {
      var newMap = {};
      var itemLinkId = wfService.getQueryUniqueResult("select itemLink.id from WfItemLink itemLink where itemLink.child = " + wfItemCurrentMostValidated.getId() + " and itemLink.discriminator like 'version'");
      var itemLink = wfService.getWfItemLink(itemLinkId);
      if(itemLink != null) {
        if(itemLink.getDateMap().get("versionDate") != null){
          var currentYear = new Packages.java.lang.Integer(itemLink.getDateMap().get("versionDate").get(Calendar.YEAR))-1;
          var cicle = 1;
          var balanceElementList = WfUtil.orderWfItemElementByStartDate(balanceElementSet, null);
          var minYear = (currentYear-3);
          var loopYear: for (currentYear; currentYear > minYear; currentYear--){
            var curentYear = currentYear;
            loopElement: for (var i = 0; i < balanceElementList.size(); i++) {
              var currentElement = balanceElementList.get(i);
              if(new Packages.java.lang.Integer(currentElement.getDateMap().get("endDate").get(Calendar.YEAR)).equals(new Packages.java.lang.Integer(currentYear))){
                if(currentElement.getNumberMap().get(valueKey) != null){
                  newMap[currentYear] = currentElement.getNumberMap().get(valueKey);
                }
              }
            }
            for (var key in newMap) {
              var newElement = new WfItemElement();
              newElement.setDiscriminator(valueKey+"PreviousYear-snapshot");
              newElement.setWfItemId(wfItemCurrentMostValidated.getId());
              newElement.getIntegerMap().put("year", new Packages.java.lang.Integer(key));
              wfService.saveOrUpdate(newElement);
              wfItemCurrentMostValidated.getWfItemElementSet().add(newElement);
            }
          }
        }
      }
    }
  }
}
```
newElement.getNumberMap().put("value", Packages.java.math.BigDecimal.valueOf(newMap[key]));
wfService.saveOrUpdate(newElement);
wfItemCurrentMostValidated.getWfItemElementSet().add(newElement);
}
}
}
}
}
)
}
}
}
}
}
}
}

var itemMostValidateId = wfService.getQueryUniqueResult("select item.mostValidatedItemId from WfItem item where item.id = " + object.getId());
var wfItemCurrentMostValidated = wfService.getWfItem(itemMostValidateId);
var balanceElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItemCurrentMostValidated, "it.cilea.wf.model.WfItemElement", "balance", wfService);
saveNewElement(wfItemCurrentMostValidated, balanceElementSet, "sales");
saveNewElement(wfItemCurrentMostValidated, balanceElementSet, "profit");

spinoffWfActionLogicCopyContributorAndExternalOrganizationUnitInsideOwner

Questa logica serve per autopopolare la Compagine sociale con tutti i Proponenti e Membri interni/esterni in fase di inizio Costituzione. Viene eseguita SE l'operatore esprime il consenso a copiare tutti i Proponenti e Membri interni/esterni nella Compagine sociale e se questa non è popolata. In caso di consenso viene riportato anche l'ateneo nella Compagine sociale SOLO SE la tipologia della Spin-off è SPI.UNI-PARTECIPATION. A tutti i partecipanti della Compagine sociale viene impostata come data di inizio validità quella della data di Costituzione SE l'operatore ha espresso il consenso in merito.

Dal punto di vista del modello dati si tratta degli elementi di tipo contributor, externalOrganizationUnit e owner. Per maggiori dettagli cfr. modello dati dell'entità in oggetto.

Configurazioni della ActionLogic:

• rm.orgunit.external.myOrganization

if(object.getBooleanMap().get("copyContributorAndExternalOrganizationUnitInsideOwner") != null && Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get("copyContributorAndExternalOrganizationUnitInsideOwner"))){
  var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
  if(ownerElementSet.isEmpty()){
    var contributorElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "contributor", wfService);
    if(!contributorElementSet.isEmpty()){
      var contributorElementSetIterator = contributorElementSet.iterator();
      while(contributorElementSetIterator.hasNext()){
        var contributorElement = contributorElementSetIterator.next();
        var newElement = new WfItemElement();
        newElement.setDiscriminator("owner");
        newElement.setWfItemId(object.getId());
        if(object.getDateMap().get("constitutionDate") != null && Packages.java.lang.Boolean.TRUE.equals(object.getDateMap().get("copyConstitutionDateInsideOwner"))){
          newElement.getDateMap().put("startDate", object.getDateMap().get("constitutionDate"));
        }
        newElement.getWfDictionaryMap().put("spinoffOwnerType", WfUtil.getWfDictionaryByCode("spinoffOwnerType.structured-internal-person", wfService));
        newElement.getPersonMap().put("ownerId", contributorElement.getPersonMap().get("contributorId"));
        wfService.saveOrUpdate(newElement);
      }
    }
  }
}
throw "Configuration variable rm.orgunit.external.
myOrganization MUST BE DEFINED";

} else {
    myOrganization = gaService.getOrganizationUnit(new Packages.
java.lang.Integer(myOrganization));
}

var externalOrganizationUnitElementSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement",
"externalOrganizationUnit", wfService);

if(!externalOrganizationUnitElementSet.isEmpty()){
    var externalOrganizationUnitElementSetIterator =
    externalOrganizationUnitElementSet.iterator();
    while(externalOrganizationUnitElementSetIterator.hasNext()){  
        var externalOrganizationUnitElement =
        externalOrganizationUnitElementSetIterator.next();
        if(externalOrganizationUnitElement.
getOrganizationUnitMap().get("ouId").getId().equals(myOrganization.getId())){  
            foundMyOrganization = true;
        }
        var newElement = new WfItemElement();
        newElement.setDiscriminator("owner");
        newElement.setWfItemId(object.getId());
        if(object.getDateMap().get("constitutionDate") != null && object.
getBooleanMap().get("copyConstitutionDateInsideOwner") != null && Packages.java.lang.Boolean.TRUE.equals(object.
getBooleanMap().get("copyConstitutionDateInsideOwner"))){
            newElement.getDateMap().put("startDate", object.getDateMap().get
("constitutionDate");
        }
        newElement.getWfDictionaryMap().put("spinoffOwnerType", WfUtil.
getWfDictionaryByCode("spinoffOwnerType.structured-external-organization", wfService));
        newElement.setOrganizationUnitMap().put("ouId",
externalOrganizationUnitElement.getOrganizationUnitMap().get("ouId");
        wfService.saveOrUpdate(newElement);
    }
}

if(!foundMyOrganization && Packages.org.apache.commons.lang.StringUtils.
equals("SPI.UNI-PARTECIPATION", wfItem.getWfItemType().getIdentifier())){
    var newElement = new WfItemElement();
    newElement.setDiscriminator("owner");
    newElement.setWfItemId(object.getId());
    if(object.getDateMap().get("constitutionDate") != null && object.
getBooleanMap().get("copyConstitutionDateInsideOwner") != null && Packages.java.lang.Boolean.TRUE.equals(object.
getBooleanMap().get("copyConstitutionDateInsideOwner"))){
        newElement.getDateMap().put("startDate", object.getDateMap().get
("constitutionDate");
    }
    newElement.getWfDictionaryMap().put("spinoffOwnerType", WfUtil.
getWfDictionaryByCode("spinoffOwnerType.structured-external-organization", wfService));
    newElement.setOrganizationUnitMap().put("ouId", myOrganization);
    wfService.saveOrUpdate(newElement);
}

object.getBooleanMap().put("copyContributorAndExternalOrganizationUnitInsideOwner",
null);
object.getBooleanMap().put("copyConstitutionDateInsideOwner", null);
if(object.getBooleanMap().get("restoreConstitution") != null && Packages.java.lang.
Boolean.TRUE.equals(object.getBooleanMap().get("restoreConstitution"))){
    var childWfItemLinkSet=object.getChildWfItemLinkSet("version");
    var childWfItemLinkSetIterator=childWfItemLinkSet.iterator();
    var wfItemConstitution = null;
    while(childWfItemLinkSetIterator.hasNext()){
        var versionLink=childWfItemLinkSetIterator.next();
        if(versionLink.getStringMap().get("versionComment").equals
(getWithdrawn()){
            wfItemConstitution = versionLink.getChild();
        }
    }
    if(wfItemConstitution != null){
        object.getBooleanMap().put("restoreConstitution", null);
        object.getWfDictionaryMap().put("versionSelector", null);
        object.getDateMap().put("versionDate", null);
        object.getDateMap().put("fictitiousCreateDate", null);
        object.getStringMap().put("versionComment", null);
        object.setMostValidatedItemId(object.getId());
        object.setMostValidatedItem(object);
        wfService.resumeVersionItem(object, wfItemConstitution, false);
        childWfItemLinkSet=object.getChildWfItemLinkSet("version");
        childWfItemLinkSetIterator=childWfItemLinkSet.iterator();
        while(childWfItemLinkSetIterator.hasNext()){
            var versionLink=childWfItemLinkSetIterator.next();
            var wfItemTemp = wfService.getWfItem(versionLink.getChildId());
            wfItemTemp.setWithdrawn(true);
            wfService.saveOrUpdate(wfItemTemp);
        }
    }
}

spinoffActiveValidator
Questa validazione esegue tutta una serie di controlli validi ai fini del Monitoraggio, di seguito i controlli:

- Verifica che tutti i membri nella Compagine sociale hanno la data di inizio validità settata
- Verifica che tutti i membri nella Compagine sociale hanno valorizzato uno dei due campi (e non entrambi) integerMap[ownershipShareNumber] - numberMap[ownershipPercentage]

Dal punto di vista del modello dati si tratta degli elementi di tipo owner.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(object, "it.cilea.wf.model.WfItemElement", "owner", wfService);
if(!ownerElementSet.isEmpty()){
    var ownerElementSetIterator = ownerElementSet.iterator();
    while(ownerElementSetIterator.hasNext()){
        var ownerElement = ownerElementSetIterator.next();
        if (ownerElement.getDateMap().get("startDate") == null)
            errors.rejectAndLocate("error.spi.owner.startDate.required",
            "owner");
        if(ownerElement.getIntegerMap().get("ownershipShareNumber") == null &&
            ownerElement.getNumberMap().get("ownershipPercentage") == null)
            errors.rejectAndLocate("error.spi.owner.
            allOwnershipShareNumberOwnershipPercentage.required", "owner");
        if(ownerElement.getIntegerMap().get("ownershipShareNumber") != null &&
            ownerElement.getNumberMap().get("ownershipPercentage") == null)
            errors.rejectAndLocate("error.spi.owner. allOwnershipShareNumberOwnershipPercentage.required", "owner");
        }
spinoffCheckCopyContributorAndExternalOrganizationUnitInsideOwnerValidator
Questa validazione chiede all'operatore se vuole portare i Proponenti, Membri interni/esterni e l'ateneo (SE la tipologia della Spin-off è SPI.UNI-PARTECIPATION) nella Compagine.
Da interfaccia verrà proposta questa possibilità tramite flag booleano.
Questa operazione viene proposta SOLO se NON è già stata censita la Compagine sociale.
Dal punto di vista del modello dati si tratta degli elementi di tipo booleanMap[copyContributorAndExternalOrganizationUnitInsideOwner], booleanMap[copyConstitutionDateInsideOwner], dateMap[constitutionDate].
Per maggiori dettagli cfr. modello dati dell'entità in questione.

    if (!errors.hasErrors()){
      var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
      (object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
      if(ownerElementSet.isEmpty()){
        if(object.getBooleanMap().get("copyContributorAndExternalOrganizationUnitInsideOwner") == null){
          errors.reject("error.spinoff.
          copyContributorAndExternalOrganizationUnitInsideOwner.generic");
          errors.rejectValueAndNotLocate("booleanMap
          [copyContributorAndExternalOrganizationUnitInsideOwner]", "error.spinoff.
          copyContributorAndExternalOrganizationUnitInsideOwner", null, null);
        } else {
          if(Packages.java.lang.Boolean.TRUE.equals(object.
          getBooleanMap().get("copyContributorAndExternalOrganizationUnitInsideOwner"))){
            if(object.getBooleanMap().get("copyConstitutionDateInsideOwner") != null){
              if(Packages.java.lang.Boolean.TRUE.equals(object.
              getBooleanMap().get("copyConstitutionDateInsideOwner"))){
                if(object.getDateMap().get("constitutionDate") == null){
                    errors.rejectValueAndNotLocate
                    ("booleanMap[copyContributorAndExternalOrganizationUnitInsideOwner]", "error.spinoff.
                    copyContributorAndExternalOrganizationUnitInsideOwner.copyConstitutionDateInsideOwner",
                    constitutionDate");
                  } else {
                    errors.reject("error.spinoff.
                    copyContributorAndExternalOrganizationUnitInsideOwner.generic"); errors.rejectValueAndNotLocate
                    ("booleanMap[copyContributorAndExternalOrganizationUnitInsideOwner]", "error.spinoff.
                    copyContributorAndExternalOrganizationUnitInsideOwner.copyConstitutionDateInsideOwner",
                    constitutionDate");
                  } else {
                    errors.rejectValue("dateMap
                    [constitutionDate]", "error.spinoff.copyContributorAndExternalOrganizationUnitInsideOwner.
                    constitutionDate");
                  }
                } else {
                  errors.reject("error.spinoff.
                  copyContributorAndExternalOrganizationUnitInsideOwner.generic");
                  errors.rejectValueAndNotLocate("booleanMap
                  [copyContributorAndExternalOrganizationUnitInsideOwner]", "error.spinoff.
                  copyContributorAndExternalOrganizationUnitInsideOwner.copyConstitutionDateInsideOwner",
                  constitutionDate");
                }
          } else {
            errors.reject("error.spinoff.
            copyContributorAndExternalOrganizationUnitInsideOwner.generic");
            errors.rejectValueAndNotLocate("booleanMap
            [copyContributorAndExternalOrganizationUnitInsideOwner]", "error.spinoff.
            copyContributorAndExternalOrganizationUnitInsideOwner.copyConstitutionDateInsideOwner",
            constitutionDate");
          }
        }
      } else {
        errors.rejectValueAndNotLocate("booleanMap
        [copyContributorAndExternalOrganizationUnitInsideOwner]", "error.spinoff.
        copyContributorAndExternalOrganizationUnitInsideOwner.copyConstitutionDateInsideOwner",
        constitutionDate");
      }
    }

spinoffOwnershipElementValidator
Questa validazione controlla che in fase di salvataggio del fragment owner sia valorizzato uno dei seguenti valori (e non entrambi):
integerMap[ownershipShareNumber]
numberMap[ownershipPercentage]
Dal punto di vista del modello dati si tratta degli elementi di tipo owner.
Per maggiori dettagli cfr. modello dati dell'entità in questione.
if(object.getIntegerMap().get("ownershipShareNumber") == null && object.getNumberMap().get("ownershipPercentage") == null){
errors.rejectValue("integerMap[ownershipShareNumber]", "error.spi.owner.ownershipShareNumberOwnershipPercentage.required", null, null);
errors.rejectValue("numberMap[ownershipPercentage]", "error.spi.owner.ownershipShareNumberOwnershipPercentage.required", null, null);
errors.reject("error.spi.owner.ownershipShareNumberOwnershipPercentage.required");
}
if(object.getIntegerMap().get("ownershipShareNumber") != null && object.getNumberMap().get("ownershipPercentage") != null){
errors.rejectValue("integerMap[ownershipShareNumber]", "error.spi.owner.ownershipShareNumberOwnershipPercentage.notCompliant", null, null);
errors.rejectValue("numberMap[ownershipPercentage]", "error.spi.owner.ownershipShareNumberOwnershipPercentage.notCompliant", null, null);
errors.reject("error.spi.owner.ownershipShareNumberOwnershipPercentage.notCompliant");
}

spinoffOwnerTypeElementValidator
Questa validazione controlla che in fase di salvataggio del fragment owner sia valorizzato uno dei seguenti valori:
personMap[ownerId]
organizationUnitMap[ouId]
stringMap[otherOwner]
Dal punto di vista del modello dati si tratta degli elementi di tipo owner.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

if(object.getPersonMap().get("ownerId") == null && object.getOrganizationUnitMap().get("ouId") == null && object.getStringMap().get("otherOwner") == null)
errors.reject("error.spi.spinoffOwnerType.required");

spinoffBoardUniversityMemberRemunerationElementValidator
Questa validazione controlla che in fase di salvataggio del fragment boardUniversityMember sia presente almeno un elemento dal fragment remuneration SOLO SE l'utente ha selezionato Sì sul campo Con remunerazione?.
Viceversa SE l'utente ha selezionato No sul campo Con remunerazione? la validazione controlla che NON sia presente alcun elemento dal fragment remuneration.
Dal punto di vista del modello dati si tratta degli elementi di tipo boardUniversityMember.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

var remunerationSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "remuneration", wfService);  
if(Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get("withRemuneration"))){
if (remunerationSet.isEmpty()){
errors.reject("error.spi.boardUniversityMember.remuneration.required");
}
} else if(Packages.java.lang.Boolean.FALSE.equals(object.getBooleanMap().get("withRemuneration"))){
if (!remunerationSet.isEmpty()){
errors.reject("error.spi.boardUniversityMember.remuneration.notCompliant");
}
}

spinoffCompanyPositionPersonSpinOffElementValidator
Questa validazione controlla che in fase di salvataggio del fragment companyPosition sia valorizzato uno dei seguenti valori:
personMap[personId]
stringMap[externalPerson]
Dal punto di vista del modello dati si tratta degli elementi di tipo companyPosition.
Per maggiori dettagli cfr. modello dati dell'entità in questione.
if(object.getPersonMap().get("personId") == null && object.getStringMap().get("externalPerson") == null)
errors.reject("error.spi.companyPositionPerson.required");

spinoffIntellectualPropertyTypeCheckConsistencyElementValidator
Questa validazione controlla che in fase di salvataggio del fragment Proprietà Intellettuale NON sia presente o che si tenti di aggiungere la tipologia N/A (non applicabile) insieme ad altre tipologie.
Dal punto di vista del modello dati si tratta degli elementi di tipo intellectualPropertyType.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

spinoffCheckElementDateBetweenMonitoringDateElementValidator
Questa validazione controlla la coerenza delle date di Inizio e Fine in fase di salvataggio dei fragment Personale, Gender equality, Bilancio e Proprietà Intellettuali (conteggi).
Se NON è presente alcun monitoraggio pregresso (la Costituzione NON viene valutata) verifica che la data di Inizio dell'elemento sia maggiore o uguale della data data di Costituzione
Se NON è presente alcun monitoraggio pregresso (la Costituzione NON viene valutata) verifica che la data di Inizio dell'elemento sia maggiore della data dell'ultimo Monitoraggio, in caso di rettifica verifica che sia maggiore della data del penultimo Monitoraggio.
Verifica che la data di Fine dell'elemento sia minore o uguale della data del corrente Monitoraggio.
Dal punto di vista del modello dati si tratta degli attributi dateMap[startDate] e dateMap[endDate].
Dal punto di vista del modello dati si tratta degli elementi di tipo staff, genderEquality, balance e intellectualPropertySummary.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

spinoffIntellectualPropertyTypeCurrent = wfItemElement.getWfDictionaryMap().get("dictionary");
var found = false;
var count = 1;
var spinoffIntellectualPropertyTypeNaId = WfUtil.getDictionaryByCode(wfService, "spinoffIntellectualPropertyType.na");
if(spinoffIntellectualPropertyTypeCurrent != null && spinoffIntellectualPropertyTypeCurrent.getId().equals(spinoffIntellectualPropertyTypeNaId)){
    found = true;
}

intellectualPropertyTypeSetIterator = intellectualPropertyTypeSet.iterator();
while(intellectualPropertyTypeSetIterator.hasNext()){
    var intellectualPropertyType = intellectualPropertyTypeSetIterator.next();
    count++;
    if(intellectualPropertyType.getWfDictionaryMap().get("dictionary") != null && intellectualPropertyType.getWfDictionaryMap().get("dictionary").getId().equals(spinoffIntellectualPropertyTypeNaId)){
        found = true;
    }
}
if(count > 1 && found){
    errors.reject("error.spi.intellectualPropertyType.notCompliant",
"intellectualPropertyType");
}

spinoffCheckElementDateBetweenMonitoringDateElementValidator
Questa validazione controlla la coerenza delle date di Inizio e Fine in fase di salvataggio dei fragment Personale, Gender equality, Bilancio e Proprietà Intellettuali (conteggi).
Se NON è presente alcun monitoraggio pregresso (la Costituzione NON viene valutata) verifica che la data di Inizio dell'elemento sia maggiore o uguale della data data di Costituzione
Se NON è presente alcun monitoraggio pregresso (la Costituzione NON viene valutata) verifica che la data di Inizio dell'elemento sia maggiore della data dell'ultimo Monitoraggio, in caso di rettifica verifica che sia maggiore della data del penultimo Monitoraggio.
Verifica che la data di Fine dell'elemento sia minore o uguale della data del corrente Monitoraggio.
Dal punto di vista del modello dati si tratta degli attributi dateMap[startDate] e dateMap[endDate].
Dal punto di vista del modello dati si tratta degli elementi di tipo staff, genderEquality, balance e intellectualPropertySummary.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

if(!object.getPersonMap().get("personId") || object.getStringMap().get("externalPerson") || object.getPersonMap().get("personId") == null && object.getStringMap().get("externalPerson") == null)
errors.reject("error.spi.companyPositionPerson.required");

var wfItemElement = object;
var wfItem = wfService.getWfItem(wfItemElement.getWfItemId());
var wfDictionaryMap = wfItemElement.getWfDictionaryMap().get("dictionary");
var found = false;
var count = 1;
var spinoffIntellectualPropertyTypeNaId = WfUtil.getDictionaryByCode(wfService, "spinoffIntellectualPropertyType.na");
if(spinoffIntellectualPropertyTypeCurrent != null && spinoffIntellectualPropertyTypeCurrent.getId().equals(spinoffIntellectualPropertyTypeNaId)){
    found = true;
}

var intellectualPropertyTypeSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "intellectualPropertyType", wfService);
var intellectualPropertyTypeSetIterator = intellectualPropertyTypeSet.iterator();
while(intellectualPropertyTypeSetIterator.hasNext()){
    var intellectualPropertyType = intellectualPropertyTypeSetIterator.next();
    count++;
    if(intellectualPropertyType.getWfDictionaryMap().get("dictionary") != null && intellectualPropertyType.getWfDictionaryMap().get("dictionary").getId().equals(spinoffIntellectualPropertyTypeNaId)){
        found = true;
    }
}
if(count > 1 && found){
    errors.reject("error.spi.intellectualPropertyType.notCompliant",
"intellectualPropertyType");
}
var versionSelectorDictionaryCode = wfService.getWfDictionary(wfItem.getWfDictionaryMap().get("versionSelector").getId()).getInternalSourceIdentifier();
if(versionSelectorDictionaryCode.equals("versionSelector.amendLastVersion")){
    var masterItemId=wfService.getWfItem(wfItem.getMasterItemId());
    var childWfItemLinkSet=masterItemId.getChildWfItemLinkSet("version");
    var childWfItemLinkSetIterator=childWfItemLinkSet.iterator();
    var previousItemLinkDate = null;
    while(childWfItemLinkSetIterator.hasNext()){;
        var versionLink=childWfItemLinkSetIterator.next();
        if(versionLink.getStringMap().get("versionComment") != "Costituzione"){
            if(Packages.java.lang.Boolean.FALSE.equals(versionLink.getChild().getWithdrawn())){
                if(versionLink.getDateMap().get("versionDate") == 0){
                    if(previousItemLinkDate != null){
                        lastMonitoringDate=previousItemLinkDate;
                    }
                    break;
                }
            }
        }
    }
}

if(versionSelectorDictionaryCode.equals("versionSelector.newVersion")){
    var lastMonitoring=wfService.getWfItem(wfItem.getMostValidatedItemId());
    var parentWfItemLinkSet=lastMonitoring.getParentWfItemLinkSet("version");
    var parentWfItemLinkSetIterator=parentWfItemLinkSet.iterator();
    while(parentWfItemLinkSetIterator.hasNext()){;
        var versionLink=parentWfItemLinkSetIterator.next();
        if (versionLink.getStringMap().get("versionComment") != "Costituzione"){
            if(Packages.java.lang.Boolean.FALSE.equals(versionLink.getChild().getWithdrawn())){
                lastMonitoringDate=versionLink.getDateMap().get("versionDate");
            }
        }
    }
}

var formatter = Packages.org.apache.commons.lang.time.FastDateFormat.getInstance("dd/MM/yyyy");
if(wfItemElementStartDate != null && (lastMonitoringDate != null || constitutionDate != null) && wfItemElementEndDate != null && currentMonitoringDate != null){
    if(lastMonitoringDate != null){
        if(wfItemElementStartDate.compareTo(lastMonitoringDate) <= 0 ||
            wfItemElementStartDate.compareTo(currentMonitoringDate) > 0){
            errors.rejectValue("dateMap[startDate]", "error.spi.element.startDate.monitoring", [formatter.format(lastMonitoringDate.getTime()),formatter.format(currentMonitoringDate.getTime())], null);
        }
    } else {
        if(wfItemElementStartDate.compareTo(constitutionDate) < 0 ||
            wfItemElementStartDate.compareTo(currentMonitoringDate) > 0){
            errors.rejectValue("dateMap[startDate]", "error.spi.element.startDate.constitution", [formatter.format(constitutionDate.getTime()),formatter.format(currentMonitoringDate.getTime())], null);
        }
    }
}

if(wfItemElementEndDate.compareTo(currentMonitoringDate) > 0){
    errors.rejectValue("dateMap[endDate]", "error.spi.element.endDate.monitoring", [formatter.format(currentMonitoringDate.getTime())], null);
}
spinoffRequirementCheckDateElementValidator
Questa validazione controlla che in fase di salvataggio del fragment requirement la data di verifica sia compilata se l'operatore ha specificato che il requisito è stato verificato.
Dal punto di vista del modello dati si tratta degli elementi di tipo requirement.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
if(Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get("validated"))){
    if(object.getDateMap().get("checkDate") == null){
        errors.rejectValue("dateMap[checkDate]", "error.notNull", [messageUtil.findMessage("label.required")], null);
    }
}
```

spinoffItemLinkParentElementValidator
Questa validazione controlla che in fase di salvataggio dei fragment di collegamento ad altre entità l'oggetto che si sta cercando di collegare abbia la data di inizio settata e che ci sia un minimo di sovrapposizione di date tra lo Spin-off e l'oggetto collegato.
Dal punto di vista del modello dati si tratta degli elementi di tipo projectSpinoffLink, contractSpinoffLink e agreementSpinoffLink.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
if (object.getParentId() != null) {
    var wfItemParent = wfService.getWfItem(object.getParentId());
    var wfItemChild = wfService.getWfItem(object.getChildId());
    if (wfItemParent.getDateMap().get("startDate") == null) {
        errors.reject("error.spi." + StringUtils.substringBefore(object.getDiscriminator(), ".") + ".parent.notCompliant");
    } else {
        var minDate = wfItemChild.getDateMap().get("accreditationDate");
        if (minDate != null) {
            if (wfItemChild.getDateMap().get("constitutionDate") != null && minDate.compareTo(wfItemChild.getDateMap().get("constitutionDate")) < 0) {
                minDate = wfItemChild.getDateMap().get("constitutionDate");
            }
        }
        if (wfItemParent.getDateMap().get("endDate") != null && minDate != null && minDate.after(wfItemParent.getDateMap().get("endDate"))) {
            errors.reject("error.spi." +.StringUtils.substringBefore(object.getDiscriminator(), ".") + ".parent.date.notCompliant");
        }
    }
}
```

spinoffIntellectualPropertyValidator
Questa validazione controlla che sia compilata la sezione delle Proprietà intellettuali.
Controlla che ALMENO uno dei due fragment intellectualPropertySpinoffLink e intellectualPropertyType sia valorizzato con ALMENO un valore.
Dal punto di vista del modello dati si tratta degli elementi di tipo intellectualPropertySpinoffLink e intellectualPropertyType.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
var intellectualPropertyConfig = ConfigurationUtil.getConfigValue("ap.spi.intellectualProperty.structured.enabled");
if (intellectualPropertyConfig == null || intellectualPropertyConfig.equalsIgnoreCase("true")) {
    var intellectualPropertySpinoffLinkSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "intellectualPropertySpinoffLink", wfService);
    if(intellectualPropertySpinoffLinkSet.isEmpty())
        errors.rejectAndLocate("error.spi.intellectualProperty.required",
```
**spinoffRequirementValidator**
Questa validazione controlla che sia compilata la sezione delle Validazione Requisiti. Controlla che sia presente ALMENO un valore nel fragment requirement e che ogni requisito sia stato validato, quindi che ogni elemento abbia settato a true il campo validated.
Dal punto di vista del modello dati si tratta degli elementi di tipo requirement. Per maggiori dettagli cfr. modello dati dell'entità in questione.

```javascript
var requirementElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "requirement", wfService);
if(requirementElementSet.isEmpty())
    errors.rejectAndLocate("error.spi.requirement.required", "requirement");
else {
    var requirementElementSetIterator = requirementElementSet.iterator();
    while(requirementElementSetIterator.hasNext()){
        var requirementElement = requirementElementSetIterator.next();
        if(requirementElement.getBooleanMap().get("validated") == null || requirementElement.getBooleanMap().get("validated") == false)
            errors.rejectAndLocate("error.spi.requirement.allElementValidated", "requirement");
    }
}
```

**spinoffHeadquartersValidator**
Questa validazione controlla che sia compilata la sezione Sedi legali e operative. Controlla che sia presente ALMENO un valore nel fragment headquarters e che almeno uno di questi sia marcato come Sede Legale. Dal punto di vista del modello dati si tratta degli elementi di tipo headquarters. Per maggiori dettagli cfr. modello dati dell'entità in questione.

```javascript
var headquartersElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "headquarters", wfService);
if(headquartersElementSet.isEmpty())
    errors.rejectAndLocate("error.spi.headquarters.required", "headquarters");
else {
    var found = false;
    var spinoffHeadquartersTypeLegalId = WfUtil.getDictionaryByCode(wfService, "spinoffHeadquartersType.legal");
    var headquartersElementSetIterator = headquartersElementSet.iterator();
    while(headquartersElementSetIterator.hasNext()){
        var headquartersElement = headquartersElementSetIterator.next();
        if(headquartersElement.getWfDictionaryMap().get("type").getId().equals(spinoffHeadquartersTypeLegalId)){
            found = true;
            break;
        }
    }
    if(!found){
        errors.rejectAndLocate("error.spi.headquarters.typeLegal.required", "headquarters");
    }
}
```

**spinoffVatAndCfValidator**
Questa validazione controlla la consistenza degli attributi Partita IVA e Codice Fiscale.
Verifica che i valori inseriti siano di 11 carattari SOLO numerici.
Dal punto di vista del modello dati si tratta degli attributi `stringMap[vatNumber]` e `stringMap[cf]`.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
if(object.getStringMap().get("vatNumber") != null){
    if(object.getStringMap().get("vatNumber").length() != 11 || !object.getStringMap().get("vatNumber").matches("\[0-9\]+")){
        errors.rejectValue("stringMap[vatNumber]", "error.spi.vatNumber.inconsistent", null, null);
    }
}

if(object.getStringMap().get("cf") != null){
    if(object.getStringMap().get("cf").length() != 11 || !object.getStringMap().get("cf").matches("\[0-9\]+")){
        errors.rejectValue("stringMap[cf]", "error.spi.cf.inconsistent", null, null);
    }
}
```

**spinoffVersionDateAndVersionSelectorValidator**
Questa logica verifica inizialmente che i campi Operazione monitoraggio e Data monitoraggio siano valorizzati.
Successivamente in base alla scelta dell'Operazione monitoraggio verifica che:

* Scelta Nuovo: viene verificato che la data di Monitoraggio corrente sia successiva all'ultima data di Monitoraggio se presente, se non presente viene verificato che sia successiva alla data di Costituzione.
* Scelta Rettifica ultimo presente: viene verificato che effettivamente esista un monitoraggio precedente (ATTENZIONE, non viene considerata la Costituzione) e viene verificato che la data di Monitoraggio sia uguale alla data del Monitoraggio da rettificare.

Dal punto di vista del modello dati si tratta degli attributi `dateMap[versionDate]` e `dateMap[constitutionDate]` e `wfDictionaryMap[versionSelector]`.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
//check necessario, per non far scattare la validazione in fase di costituzione e in caso di ritorno In costituzione (perche si tratta del riavvio del processo di costituzione)
var spiNextState = WfUtil.getWfNextState(request);
    if(object.getWfDictionaryMap().get("versionSelector") != null){
        var versionSelectorDictionaryCode = wfService.getWfDictionary(object.getWfDictionaryMap().get("versionSelector").getId()).getInternalSourceIdentifier();
        if(object.getDateMap().get("versionDate") != null){
            var versionSelectorDictionaryCode.equals("versionSelector.newVersion"){
                var lastDate=null;
                var lastMonitoringItem=wfService.getWfItem(object);
                getNextMonitoringItemId();
                var parentWfItemLinkSet=lastMonitoringItem.getParentWfItemLinkSet("version");
                var parentWfItemLinkSetIterator=parentWfItemLinkSet.iterator();
                while(parentWfItemLinkSetIterator.hasNext()){var versionLink=parentWfItemLinkSetIterator.next();
                    if(Packages.java.lang.Boolean.FALSE.equals(versionLink.getChild().getWithdrawn())){
                        lastDate=versionLink.getDateMap().get("versionDate");
                    }
                }
                if (lastDate==null){
                    lastDate=object.getDateMap().get("constitutionDate");
                }
            }
        }
    }

    var firstRightDate = lastDate;
```
var cal=Calendar.getInstance();
cal.setTime(firstRightDate.getTime());
cal.add(Calendar.DATE, 1);
firstRightDate=cal;

if(object.getDateMap().get("versionDate").before

if(object.getDateMap().get("versionDate").before

errors.rejectValueAndNotLocate("dateMap[versionDate]", "error.spi.versionDate.newVersion.notCompliant", [formatter.format(lastDate.getTime())], null);
}

} else if(versionSelectorDictionaryCode.equals("versionSelector.amendLastVersion")){

var lastMonitoringDate = null;
var lastMonitoringItem=wfService.getWfItem(object.

getParentWfItemLinkSet("version");

if(!parentWfItemLinkSet.isEmpty()){

var found = false;
var

parentWfItemLinkSetIterator=parentWfItemLinkSet.iterator();
while(parentWfItemLinkSetIterator.hasNext()){

var versionLink =

parentWfItemLinkSetIterator.next();
if(versionLink.getStringMap().get

"versionComment") != null && !versionLink.getStringMap().get("versionComment").equals("Costituzione"){

if(Packages.java.lang.Boolean.

FALSE.equals(versionLink.getChild().getWithdrawn())){

lastMonitoringDate =

versionLink.getDateMap().get("versionDate");

} 

}

if(lastMonitoringDate == null){

errors.rejectValueAndNotLocate

("wfDictionaryMap[versionSelector]", "error.spi.versionSelector.amendNthVersion.notAllowed.oldMonitorig", null, null);
} else {

if(object.getDateMap().get

apache.commons.lang.time.FastDateFormat.getInstance("dd/MM /yyyy");

errors.rejectValueAndNotLocate

("dateMap[versionDate]", "error.spi.versionDate.amendLastVersion.notCompliant", [formatter.format

(lastMonitoringDate.getTime())], null);
}
} else {

errors.rejectValueAndNotLocate("wfDictionaryMap[versionSelector]", "error.spi.versionSelector.amendNthVersion.notAllowed", null, null);
}
} else {

errors.rejectValueAndNotLocate("dateMap[versionDate]", "error.

notNull.fieldLabelKey", [messageUtil.findMessage("label.spinoff.versionDate")],
null);
} else {

errors.rejectValueAndNotLocate("wfDictionaryMap[versionSelector]", "error.

notNull.fieldLabelKey", [messageUtil.findMessage("label.spinoff.versionSelector")], null);
}
} else {

errors.rejectValueAndNotLocate("dateMap[versionDate]", "error.

notNull.fieldLabelKey", [messageUtil.findMessage("label.spinoff.versionDate")],
null);
} else {

errors.rejectValueAndNotLocate("wfDictionaryMap[versionSelector]", "error.

notNull.fieldLabelKey", [messageUtil.findMessage("label.spinoff.versionSelector")], null);
}
spinoffVersionMoreInformationValidator
Questa validazione esegue una serie di operazioni al fine della Costituzione, del Monitoraggio, della Cessassione e della Liquidazione.
In caso di Costituzione:
- La data di monitoraggio viene settata con la data di Costituzione
- Il commento del monitoraggio viene settato con la dicitura "Costituzione"

In caso di Monitoraggio:
- La data di monitoraggio viene settata con la data scelta dall'utente
- Il commento del monitoraggio viene settato con la dicitura "Monitoraggio alla data + DATA DI MONITORAGGIO"

In caso di Cessazione
- La data di monitoraggio viene settata con la data scelta dall'utente
- La data di cessione viene settata dall'utente
- Il commento del monitoraggio viene settato con la dicitura "Monitoraggio alla data + DATA DI MONITORAGGIO"

In caso di Liquidazione
- La data di monitoraggio viene settata con la data scelta dall'utente
- La data di liquidazione viene settata dall'utente
- Il commento del monitoraggio viene settato con la dicitura "Monitoraggio alla data + DATA DI MONITORAGGIO"

In caso di Scadenza
- La data di monitoraggio viene settata con la data scelta dall'utente
- La data di scadenza viene settata dall'utente
- Il commento del monitoraggio viene settato con la dicitura "Monitoraggio alla data + DATA DI MONITORAGGIO"

Dal punto di vista del modello dati si tratta degli attributi \texttt{wfDictionaryMap[versionSelector]}, \texttt{dateMap[versionDate]}. Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
if (!errors.hasErrors()){
    var spiCurrentState = object.getWfState().getDescription();
    if(Packages.org.apache.commons.lang.StringUtils.equals("monitoring", spiCurrentState)){
        var formatter = Packages.org.apache.commons.lang.time.FastDateFormat.getInstance("dd/MM/yyyy");
        var versionDateFormatted = (object.getDateMap().get("versionDate") == null ? "DATA NON DISPONIBILE" : formatter.format(object.getDateMap().get("versionDate").getTime()));
        object.getStringMap().put("versionComment", "Monitoraggio alla data + " + versionDateFormatted);
    } else {
        var versionSelectorId = WfUtil.getDictionaryByCode(wfService, "versionSelector.newVersion");
        if (versionSelectorId != null){
            object.getWfDictionaryMap().put("versionSelector", wfService.getWfDictionary(new Packages.java.lang.Integer(versionSelectorId)));
        }
        var constitutionDate = object.getDateMap().get("constitutionDate");
        if (constitutionDate != null){
            object.getDateMap().put("versionDate", constitutionDate);
        }
        object.getStringMap().put("versionComment", "Costituzione");
    }
}
```

spinoffFullTransferQuoteValidator
Questa validazione controlla che l'ateneo non abbia più quote di finanziamento associate SE la tipologia è \textit{spinOff non partecipata dall'Ateneo} o se si sta tentando di portare la Spin-off nello stato Ceduto (\textit{sold}).
Dal punto di vista del modello dati si tratta degli elementi di tipo \textit{owner}.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

Configurazioni della ValidateLogic:
- \texttt{rm.orgunit.external.myOrganization}

```java
var myOrganization = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
```
if (myOrganization == null) {
    throw "Configuration variables ap.project.partner.role.main and rm.orgunit.
external.myOrganization MUST BE DEFINED";
} else {
    myOrganization = new Packages.java.lang.Integer(myOrganization);
}

var checkDate = (wfItem.getDateMap().get("versionDate") != null ? wfItem.getDateMap().get("versionDate") : wfItem.getDateMap().get("constitutionDate"));

var wftimeTypeNoPartecipation = wfService.getWfItemTypeByIdentifier("SPI.NO-UNI-PARTECIPATION");
var spiNextState = WfUtil.getWfNextState(request);

if(checkDate != null && (Packages.org.apache.commons.lang.StringUtils.equals(wftimeTypeNoPartecipation.getId(), object.getWfItemTypeId()) || Packages.org.apache.commons.lang.StringUtils.equals("sold", spiNextState))){
    var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "owner", wfService);
    if(!ownerElementSet.isEmpty()){ 
        var ownerElementSetIterator = ownerElementSet.iterator();
        while(ownerElementSetIterator.hasNext()){ 
            var ownerElement = ownerElementSetIterator.next();
            var orgUnit = ownerElement.getOrganizationUnitMap().get("ouId");
            if (orgUnit != null && myOrganization.equals(orgUnit.getId())) {
                if (ownerElement.getDateMap().get("startDate") != null && ((ownerElement.getDateMap().get("startDate").before(checkDate) && (ownerElement.getDateMap().get("endDate").equals(BigDecimal.ZERO)))) { 
                    errors.rejectAndLocate("error.spi.owner.
fullTransferQuote", "owner");
                } 
                } 
                } 
                } 
                }
}

spinoffMonitoringIntervalDatesValidator

QUESTA VALIDAZIONE NON DEVE ESSERE DISABILITATA

Questa validazione controlla la coerenza degli elementi inseriti nelle sezioni Personale, Gender equality, Bilancio e Proprietà Intellettuali (conteggi).

Verifica che ogni elemento inserito abbia le date di inizio e fine valorizzate.

Verifica che NON ci siano sovrapposizioni di date, per ogni tipologia SE PRESENTE.

Se presente una configurazione di questo tipo ap.spi.DISCRIMINATOR_ELEMENT.endDateMonitoring.required settata a true (es: ap.spi.balance.endDateMonitoring.required = true) verifica che ci sia almeno un elemento con la data di fine settata uguale alla data del corrente Monitoraggio.

Dal punto di vista del modello dati si tratta degli attributi dateMap[startDate] e dateMap[endDate].

Per maggiori dettagli cfr. modello dati dell'entità in questione.

Configurazioni della ValidateLogic:

- ap.spi.DISCRIMINATOR_ELEMENT.endDateMonitoring.required

function check(elementList, discriminatorElement, isRequiredForMonitoring, versionDate) {
    var found = false;
    //questo check è necessario nel caso in cui si tenta di concludere il
    monitoraggio dalla pagina di lista degli Spin-off
    //perché la validazione spinoffVersionDateAndVersionSelectorValidator non
    scatterebbe
    for (var i = 0; i < elementList.size(); i++)
    {
        var currentElement = elementList.get(i);
        if(currentElement.getDateMap().get("endDate") == null ||
        currentElement.getDateMap().get("endDate") != null)
        {
            errors.rejectAndLocate("error.spi."+discriminatorElement+".
startAndDateRequired", discriminatorElement);
            break;
        } 
    }
}
if (versionDate.equals(currentElement.getDateMap().get("endDate"))) {
    found = true;
}

if (i + 1 == elementList.size() || elementList.size() == 1) {
    break;
}
var nextElement = elementList.get(i + 1);

if (versionDate.equals(nextElement.getDateMap().get("endDate"))) {
    found = true;
}

var differenceInMillis = nextElement.getDateMap().get("startDate").getTimeInMillis() - currentElement.getDateMap().get("endDate").getTimeInMillis();
var differenceInDays = (differenceInMillis / (1000 * 60 * 60 * 24));
if (differenceInDays < 0) {
    errors.rejectAndLocate("error.spi." + discriminatorElement + ".overlap", discriminatorElement);
}

if (!found && isRequiredForMonitoring) {
    errors.rejectAndLocate("error.spi." + discriminatorElement + ".endDateMonitoring", discriminatorElement);
}
}

var versionDate = object.getDateMap().get("versionDate");
if (Packages.org.apache.commons.lang.StringUtils.equals("monitoring", object.getWfState().getDescription()) && versionDate != null) {
    var allDiscriminatorElement = ["staff", "genderEquality", "balance", "intellectualPropertySummary"];
    for (var d = 0; d < allDiscriminatorElement.length; d++) {
        var discriminatorElement = allDiscriminatorElement[d];
        var isRequiredForMonitoring = WfUtil.getLenientRootConfiguration("ap.<wfItemType>." + discriminatorElement + ".endDateMonitoring.required", object, false);
        var elementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", discriminatorElement, wfService);
        if (!elementSet.isEmpty()) {
            var dictionaryList = wfService.getWfDictionaryList("spinoff" + discriminatorElement.charAt(0).toUpperCase() + discriminatorElement.slice(1) + "Type", false, null);
            if (dictionaryList.isEmpty()) {
                var elementList = WfUtil.orderWfItemElementByStartDate(elementSet, null);
                check(elementList, discriminatorElement, isRequiredForMonitoring, versionDate);
            } else {
                for (var i = 0; i < dictionaryList.size(); i++) {
                    var currentDictionary = dictionaryList.get(i);
                    var elementList = WfUtil.orderWfItemElementByStartDate(elementSet, null);
                    var elementToCheckList = new Packages.java.util.ArrayList();
                    for (var e = 0; e < elementList.size(); e++) {
                        var currentElement = elementList.get(e);
                        if (currentDictionary.getInternalSourceIdentifier().equals(currentElement.getWfDictionaryMap().get(discriminatorElement + "Type").getInternalSourceIdentifier())) {
                            elementToCheckList.add((currentElement);
                        }
                    }
                    check(elementToCheckList, discriminatorElement, isRequiredForMonitoring, versionDate);
                }
            }
        } else if (isRequiredForMonitoring) {
            errors.rejectAndLocate("error.spi." + discriminatorElement + ".endDateMonitoring", discriminatorElement);
        }
    }
}
spinoffMemberAuthorizationCheckValidator
Questa validazione controlla le autorizzazioni.
Verifica che tutte le persone interne della Compagine sociale con ruolo Socio operativo abbiano l’autorizzazione per il periodo corrente nella sezione Autorizzazioni.
Verifica che tutte le persone con Cariche Societarie abbiano l’autorizzazione per il periodo corrente nella sezione Autorizzazioni.
Per periodo corrente si intende che le startDate e endDate di operatività della persona rientrino nell’intervallo delle startDate e endDate della autorizzazione.
Nel caso in cui la persona non abbia la data di fine settata, mentre l’autorizzazione si, viene verificato che la data di fine dell’autorizzazione sia successiva o uguale alla data corrente.
Dal punto di vista del modello dati si tratta degli elementi di tipo memberAuthorization, owner, companyPosition.
Per maggiori dettagli cfr. modello dati dell’entità in questione.

function check(wfItemElement, personDiscriminator) {
  var found = false
  var memberAuthorizationElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "memberAuthorization", wfService);
    if(!memberAuthorizationElementSet.isEmpty()){
        var memberAuthorizationElementSetIterator = memberAuthorizationElementSet.iterator();
            while(memberAuthorizationElementSetIterator.hasNext()){
                var memberAuthorizationElement = memberAuthorizationElementSetIterator.next();
                if(memberAuthorizationElement.getPersonMap().get("personId") != null && wfItemElement.getPersonMap().get(personDiscriminator) != null){
                    if(memberAuthorizationElement.getPersonMap().get("personId") == wfItemElement.getPersonMap().get(personDiscriminator)){
                        if(memberAuthorizationElement.getDateMap().get("startDate") != null && wfItemElement.getDateMap().get("startDate") != null){
                            if(memberAuthorizationElement.getDateMap().get("startDate").compareTo(wfItemElement.getDateMap().get("startDate")) <= 0){
                                var currentDate=Calendar.getInstance();
                                currentDate.set(Calendar.HOUR_OF_DAY, 0);
                                currentDate.set(Calendar.MINUTE, 0);
                                currentDate.set(Calendar.SECOND, 0);
                                currentDate.set(Calendar.MILLISECOND, 0);
                                currentDate.setTime(currentDate.getTime());
                                if((memberAuthorizationElement.getDateMap().get("endDate") == null) || (wfItemElement.getDateMap().get("endDate") != null && memberAuthorizationElement.getDateMap().get("endDate").compareTo(wfItemElement.getDateMap().get("endDate")) >= 0) || (wfItemElement.getDateMap().get("endDate") == null && memberAuthorizationElement.getDateMap().get("endDate").compareTo(currentDate) >= 0)){
                                    found = true;
                                break;
                                }
                            }
                        }
                    }
                }
            }
    }
  return found;
}

var personElementErrorList = new Packages.java.util.HashSet();

var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "owner", wfService);
if(!ownerElementSet.isEmpty()){  
var spinoffOwnerRoleOperativeAssociateId = WfUtil.getDictionaryByCode(wfService, "spinoffOwnerRole.operate-associate");  
var ownerElementSetIterator = ownerElementSet.iterator();  
while(ownerElementSetIterator.hasNext()){  
var ownerElement = ownerElementSetIterator.next();  
if(ownerElement.getPersonMap().get("ownerId") != null && ownerElement.getWfDictionaryMap().get("roleId").getId().equals(spinoffOwnerRoleOperativeAssociateId)){  
if(!check(ownerElement, "ownerId")){  
personElementErrorList.add(ownerElement.getPersonMap().get("ownerId"));  
}
}
}
}

var companyPositionElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "companyPosition", wfService);  
if(!companyPositionElementSet.isEmpty()){  
var companyPositionElementSetIterator = companyPositionElementSet.iterator();  
while(companyPositionElementSetIterator.hasNext()){  
var companyPositionElement = companyPositionElementSetIterator.next();  
if(companyPositionElement.getPersonMap().get("personId") != null){  
if(!check(companyPositionElement, "personId")){  
personElementErrorList.add(companyPositionElement.getPersonMap().get("personId"));  
}
}
}
}

if(personElementErrorList.size() > 0){  
errors.rejectAndLocate("error.spi.ownerAndCompanyPosition.personAuthorization", "memberAuthorization", [WfUtil.getSelectableListAsHtmlList(new Packages.java.util.ArrayList(personElementErrorList))], null);
}

spinoffPartecipationCheckValidator
Questa validazione controlla in base alla tipologia della Spin-off se l'ateneo è presente nella Compagine sociale

- Se la tipologia è spinOff partecipata dall'Ateneo viene verificato che l'ateneo sia presente nella Compagine sociale
- Se la tipologia è spinOff non partecipata dall'Ateneo viene verificato che l'ateneo NON sia presente nella Compagine sociale

Dal punto di vista del modello dati si tratta degli elementi di tipo owner.
Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configurazioni della ValidateLogic:

- rm.orgunit.external.myOrganization

var checkDate = (wfItem.getDateMap().get("versionDate") != null ? wfItem.getDateMap().get("versionDate") : wfItem.getDateMap().get("constitutionDate"));

var myOrganization = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
if (myOrganization == null){
    throw "Configuration variable rm.orgunit.external.myOrganization MUST BE DEFINED";
}

if(checkDate != null){
    var isPresent = false;
    var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);  
if(!ownerElementSet.isEmpty()){  
    var ownerElementSetIterator = ownerElementSet.iterator();  
    while(ownerElementSetIterator.hasNext()){  

spinoffOwnershipValidator
Questa validazione controlla in base alla Forma giuridica della Spin-off, se questa NON è di tipo SPA.
In tal caso verifica che venga raggiunto il 100% di Misura della partecipazione (in %) tra i membri della Compagine sociale attivi alla corrente data di Costituzione o ultima data di monitoraggio.
Dal punto di vista del modello dati si tratta degli attributi numberMap[ownershipPercentage] e wfDictionaryMap[companyType].
Dal punto di vista del modello dati si tratta degli elementi di tipo owner.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```javascript
var spinoffCompanyTypeSpaId = WfUtil.getDictionaryByCode(wfService, "spinoffCompanyType.spa");
var spinoffCompanyType = object.getWFDictionaryMap().get("companyType");
var checkDate = (wfItem.getDateMap().get("versionDate") != null ? wfItem.getDateMap().get("versionDate") : wfItem.getDateMap().get("constitutionDate"));
if(checkDate != null && spinoffCompanyTypeSpaId != null && spinoffCompanyType != null && !spinoffCompanyTypeSpaId.equals(spinoffCompanyType.getId())){
    var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "owner", wfService);
    if(!ownerElementSet.isEmpty()){
        var ownerElementSetIterator = ownerElementSet.iterator();
        while(ownerElementSetIterator.hasNext()){
            var ownerElement = ownerElementSetIterator.next();
            if (ownerElement.getDateMap().get("startDate") != null) {
                if (ownerElement.getDateMap().get("endDate") != null) {
                    if (ownerElement.getDateMap().get("startDate").compareTo(checkDate) <= 0 && ownerElement.getDateMap().get("endDate").compareTo(checkDate) >= 0)) {
                        isPresent = true;
                        break;
                    }
                } else {
                    if (ownerElement.getDateMap().get("startDate").compareTo(checkDate) <= 0) {
                        sum = sum.add(ownerElement.getNumberMap().get("ownershipPercentage") != null ? BigDecimal.valueOf(ownerElement.getNumberMap().get("ownershipPercentage")) : BigDecimal.ZERO);
                    }
                }
            }
        }
    }
    if(!isPresent) {
        errors.rejectAndLocate("error.spi.owner.myOrganization.notPresent", "owner", null, null);
    }
}
```
spinoffRestoreConstitutionValidator
Questa validazione ha il compito di chiedere all'utente la conferma di voler riavviare tutto il processo costitutivo. In caso di conferma la Spin-off verrà riportata allo stato **In costituzione**, verranno annullati TUTTI i Monitoraggi e la Costituzione. Dal punto di vista del modello dati si tratta degli attributi **booleanMap[restoreConstitution]**. Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
if(Packages.org.apache.commons.lang.StringUtils.equals("monitoring", object.getWfState().getDescription())){
    if(object.getBooleanMap().get("restoreConstitution") == null){
        errors.rejectValueAndNotLocate("booleanMap[restoreConstitution]", "error.spinoff.restoreConstitution.confirm", null, null);
    } else if(Packages.java.lang.Boolean.FALSE.equals(object.getBooleanMap().get("restoreConstitution"))){
        errors.rejectValueAndNotLocate("booleanMap[restoreConstitution]", "error.spinoff.restoreConstitution.notAllowed", null, null);
    }
}
```

spinoffElementPrepopulateLogic
Questa logica effettua l'inizializzazione delle date di inizio e fine (se presente la configurazione) per gli elementi degli oggetti spinoff dove è iniettata:

- historicalDescription
- headquarters
- owner
- companyPosition
- memberAuthorization
- publiclyControlled
- body
- boardUniversityMember
- staff
- genderEquality
- balance
- intellectualPropertySummary
- agreement
- project
- contract
- universityInfrastructureUseApproval

La data di inizio viene valorizzata alternativamente con:

- Data dell'ultimo monitoraggio+1: se disponibile monitoraggio precedente
- Data di costituzione: se non ci sono monitoraggi precedenti
- nessun valore se non viene trovata nessuna delle due informazioni precedenti

La data di fine dell'elemento viene valorizzata con la data di riferimento del **monitoraggio corrente SE** la configurazione **ap.spi.monitoring.Discriminat OR.endDate.prepopulate** è valorizzata con true. Il segnaposto DISCRIMINATOR deve essere sostituito con il nome dell'elemento.

Ad esempio **ap.spi.monitoring.genderEquality.endDate.prepopulate** valorizzata a true causa l'inizializzazione anche della data di fine con la data di riferimento del monitoraggio corrente.

Configurazioni della ValidateLogic:

- **ap.spi.ELEMENTO.validator.date.start.discriminator**
- **ap.spi.monitoring.ELEMENTO.endDate.prepopulate**
function getSpinoffElementStartDate(wfItem, startDateDiscriminator) {
  var startDate = null;
  if (WfUtil.isAmendLastVersion(wfService, wfItem)) {
    var masterItemId = wfService.getWfItem(wfItem.getMasterItemId());
    var childWfItemLinkSet = masterItemId.getChildWfItemLinkSet("version");
    var childWfItemLinkSetIterator = childWfItemLinkSet.iterator();
    var previousItemLinkDate = null;
    while (childWfItemLinkSetIterator.hasNext()) {
      var versionLink = childWfItemLinkSetIterator.next();
      if (versionLink.getStringMap().get("versionComment") !== "Costituzione") {
        if (versionLink.getStringMap().get("versionComment") !== "") {
          if (versionLink.getDateMap().get("versionDate").compareTo(wfItem.getDateMap().get("versionDate")) == 0) {
            if (previousItemLinkDate != null) {
              startDate = previousItemLinkDate;
              break;
            }
          }
          previousItemLinkDate = versionLink.getDateMap().get("versionDate");
        }
      }
    }
  } else {
    var lastMonitoring = wfService.getWfItem(wfItem.getMostValidatedItemId());
    var parentWfItemLinkSet = lastMonitoring.getParentWfItemLinkSet("version");
    var parentWfItemLinkSetIterator = parentWfItemLinkSet.iterator();
    while (parentWfItemLinkSetIterator.hasNext()) {
      var versionLink = parentWfItemLinkSetIterator.next();
      if (versionLink.getStringMap().get("versionComment") !== "Costituzione") {
        if (versionLink.getStringMap().get("versionComment") !== "") {
          startDate = versionLink.getDateMap().get("versionDate");
        }
      }
    }
  }
  if (startDate == null) {
    startDate = wfItem.getDateMap().get(startDateDiscriminator);
  } else {
    var cal = Calendar.getInstance();
    cal.setTime(startDate.getTime());
    cal.add(Calendar.DATE, 1);
    startDate = cal;
  }
  return startDate;
}

var element = object;
var wfItem = wfService.getWfItem(element.getWfItemId());

var elementStatus = FragmentUtil.getFragmentableInfo(element).getStatus().name();
if (elementStatus == "NEW") {
  var elementDiscriminator = FragmentUtil.getFragmentableInfo(element).getDiscriminator();
  var startDateDiscriminator = ConfigurationUtil.getConfigValue("ap.spi." + elementDiscriminator + ".validator.date.start.discriminator");
  if (!startDateDiscriminator) {
    startDateDiscriminator = "constitutionDate";
  }
  startDate = wfItem.getDateMap().put("startDate", getSpinoffElementStartDate(wfItem, startDateDiscriminator));
}
var elementDiscriminator=FragmentUtil.getFragmentableInfo(element).getDiscriminator();
if (ConfigurationUtil.getLenientBoolean("ap.spi.monitoring."+elementDiscriminator+".endDate.prepopulate", false))

    element.getDateMap().put("endDate", wfItem.getDateMap().get("versionDate");
}

isNotCancelMonitoring
Questa è un’applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera l’attributo cancelMonitoring, tipicamente con etichetta “Annulla monitoraggio”
Per tutti gli oggetti che hanno questo attributo valorizzato a true vengono disattivate le validazioni.

(l!Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get("cancelMonitoring")))

laboratoryValidatorWorkgroup
Questa validazione controlla che se per ogni laboratorio collegato al gruppo di lavoro, questo sia in stato “Approved” o in stato “Reopened”, altrimenti solleva un errore.
Dal punto di vista del modello dati si tratta degli elementi di tipo workgroupLaboratoryLink
Per maggiori dettagli cfr. il modello dati dell’entità Gruppi di Lavoro

//ricavo dal fragment dei laboratory
var laboratorySet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getChildWfItemLinkSet", "it.cilea.wf.model.WfItemLink",
"workgroupLaboratoryLink", wfService);
//controllo banale, controllo che ci sia almeno un link, sennò non vado avanti
if(laboratorySet.size() > 0){
    //ricavo l'iteratore lo uso per ciclare il set
    var laboratorySetIterator = laboratorySet.iterator();
    while( laboratorySetIterator.hasNext() ){
        //ricavo il singolo link e prendo il child che è il laboratorio
        var laboratoryLinkElement = laboratorySetIterator.next();
        var laboratory = laboratoryLinkElement.getChild();
        //poi ricavo il suo stato e controllo che sia in approved
        var laboratoryState = laboratory.getWfState().getDescription();
        if(!(Packages.org.apache.commons.lang.StringUtils.equals("approved",
laboratoryState) ||
          Packages.org.apache.commons.lang.StringUtils.equals("reopened", laboratoryState)){
            //se non è approved o reopened allora lancia l'errore
            errors.rejectAndLocate("error.workgroup.laboratory.
              mustBeApproved", "workgroupLaboratoryLink");
        }
    }
}

equipmentValidatorWorkgroup
Validazione che controlla che se esiste almeno una Grande Attrezzatura collegata al gruppo di lavoro, questa Attrezzatura deve essere in stato “Approved” o in stato “Reopened”, altrimenti lancia un errore.
Dal punto di vista del modello dati si tratta degli elementi di tipo workgroupEquipmentLink
Per maggiori dettagli cfr. modello dati dell’entità Gruppi di Lavoro

//ricavo dal fragment dei equipment
var equipmentSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getChildWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "workgroupEquipmentLink", wfService);

//controllo banale, controllo che ci sia almeno un link, sennò non vado avanti
if(equipmentSet.size()>0){

    //ricavo l'iteratore lo uso per ciclare il set
    var equipmentSetIterator = equipmentSet.iterator();
    while( equipmentSetIterator.hasNext() ){

        //ricavo il singolo link e prendo il child che è il equipment
        var equipmentLinkElement = equipmentSetIterator.next();
        var equipment = equipmentLinkElement.getChild();
        //poi ricavo il suo stato e controllo che sia in approved
        var equipmentState = equipment.getWfState().getDescription();

            //se non è approved o reopened allora lancia l'errore
            errors.rejectAndLocate("error.workgroup.equipment.mustBeApproved", "workgroupEquipmentLink");
        }
    }
}

approvedWorkgroupValidator
Validazione che controlla che, mentre viene collegato ad altri gruppi di lavoro in un rapporto bidirezionale, il Gruppo non venga collegato a Gruppi che non siano in stato "approved" o "reopened".
Dal punto di vista del modello dati si tratta degli elementi di tipo workgroupJointLink
Per maggiori dettagli cfr. modello dati dell'entità Gruppi di Lavoro

//ricavo dal fragment dei workgroup
var workgroupSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getChildWfItemLinkSet", "it.cilea.wf.model.WfItemLink", "workgroupJointLink", wfService);

//controllo banale, controllo che ci sia almeno un link, sennò non vado avanti
if(workgroupSet.size()>0){

    //ricavo l'iteratore lo uso per ciclare il set
    var workgroupSetIterator = workgroupSet.iterator();
    while( workgroupSetIterator.hasNext() ){

        //ricavo il singolo link e prendo il child che è il workgroup collegato
        var workgroupLinkElement = workgroupSetIterator.next();
        var workgroup = workgroupLinkElement.getChild();
        //poi ricavo il suo stato e controllo che sia in approved o reopened
        var workgroupState = workgroup.getWfState().getDescription();

            //se non è approved o reopened allora lancia l'errore
            errors.rejectAndLocate("error.workgroup.workgroupJointLink.mustBeApproved", "workgroupJointLink");
        }
    }
}

projectValidatorWorkgroup
Questa validazione controlla che per ogni progetto collegato al gruppo di lavoro, questo sia in uno stato che è contenuto nella variabile ap.wkg, workgroupProjectLink.csvAllowedProjectState altrimenti solleva un errore.
Questa variabile di configurazione deve essere popolata con elenco (separato da virgola) degli stati accettabili per i progetti agganciabili.
Gli stati configurati di default sono operative, concluded
Dal punto di vista del modello dati si tratta degli elementi di tipo workgroupProjectLink
Per maggiori dettagli cfr. modello dati dell'entità Gruppi di Lavoro
Configurazioni della ValidateLogic:

- ap.wkg.workgroupProjectLink.csvAllowedProjectState

```java
//ricavo dal fragment dei project
var projectSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getChildWfItemLinkSet", "it.cilea.wf.model.WfItemLink",
"workgroupProjectLink", wfService);

//controllo banale, controllo che ci sia almeno un link, senzò non vado avanti
if(projectSet.size()>0){

//variabile che mi servirà più tardi
var configStatesArray=[];

//controllo per vedere se è definita la variabile di configurazione con gli
stati accettati
var csvAllowedProjectState=Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue("ap.wkg.workgroupProjectLink.csvAllowedProjectState");
if (!csvAllowedProjectState){
//nel caso non sia definita
throw "Configuration variable ap.wkg.workgroupProjectLink.
csvAllowedProjectState MUST BE DEFINED";
}else{
    //vado a immettere in una lista tutti gli stati ricavati
    configStatesArray = csvAllowedProjectState.split(',
    );
}

//ricavo l'iteratore lo uso per ciclare il set
var projectSetIterator = projectSet.iterator();
while( projectSetIterator.hasNext() ){

    //ricavo il singolo link e prendo il child che è il project
    var projectLinkElement = projectSetIterator.next();
    var project = projectLinkElement.getChild();
    //poi ricavo il suo stato e controllo che sia in uno degli stati
    ricavati dalla config
    var projectState = project.getWfState().getDescription();
    //vado a confrontare lo stato del progetto con tutti quelli ricavati
    //se non è tra quelli in configurazione, scatto il messaggio di errore
    //(variabile per controllo, viene rigenerata per ogni progetto collegato
    var isPermited=false;
    for(var i=0; i<configStatesArray.length; i++){
        if(Packages.org.apache.commons.lang.StringUtils.equals
            (configStatesArray[i], projectState))
            isPermited=true;
    }

    if(isPermited==false){
        //uso questo stratagemma visto che non conosco quanti stati
        possano essere accettati
        var arrayStatesNamesAllArray = [];
        var arrayStatesNamesAllString="";
        //ciclo i nomi degli stati permessi e vado a ricavare il loro
        nome 'normale'
        for(var i=0; i<configStatesArray.length; i++){
            if(i>0)
                arrayStatesNamesAllString +=", ";
            arrayStatesNamesAllString += Packages.it.cilea.core.
util.MessageUtilConstant.messageUtil.findMessage("wfState." +
configStatesArray[i]);
        }
        arrayStatesNamesAllArray.push(arrayStatesNamesAllString);
        errors.rejectAndLocate("error.workgroup.project.
stateMustBeAccepted", "workgroupProjectLink", arrayStatesNamesAllArray, null);
    }
}
```
wfActionLogicEnterSetForwardInConfirmedDate

```java
if (wfItem.getDateMap().get('forwardInConfirmedDate')==null) wfItem.getDateMap().put('forwardInConfirmedDate', java.util.Calendar.getInstance());
```

wfActionLogicEnterSetForwardInCompliantDate

```java
if (wfItem.getDateMap().get('forwardInCompliantDate')==null) wfItem.getDateMap().put('forwardInCompliantDate', java.util.Calendar.getInstance());
```

WfActionLogicSaveContextVision

Questa action logic serve per generare i task necessari per rendere visibile l'oggetto in visione dipartimentale. I task vengono generati in automatico in fase di transizione da uno stato ad un altro. Nei casi in cui le informazioni vengono modificate senza effettuare nessuna transizione è necessario procedere ad una rigenerazione dei task. Ad esempio se viene aggiunto un nuovo dipartimento ad un progetto e viene effettuato un salvataggio (senza transizione di stato), questa logica, agganciata in salvataggio, genera i task per rendere visibile l'oggetto anche in visione dipartimentale del nuovo dipartimento.

```java
it.cilea.wf.logic.action.save.WfActionLogicSaveContextVision
```

WfActionLogicSavePersonalVision

Questa action logic serve per generare i task necessari per rendere visibile l'oggetto nella visione personale. I task vengono generati in automatico in fase di transizione da uno stato ad un altro. Nei casi in cui le informazioni vengono modificate senza effettuare nessuna transizione è necessario procedere ad una rigenerazione dei task.

```java
it.cilea.wf.logic.action.save.WfActionLogicSavePersonalVision
```

WfActionLogicVersionize

Questa action logica crea una snapshot per l'oggetto in questione e lo conserva tra versioni precedentis

```java
it.cilea.wf.logic.WfActionLogicVersionize
```

WfActionLogicEnterMailSender

Questa action logic effettua l'invio di segnalazioni via mail ai vari attori del flusso. L'invio delle segnalazioni viene pilotato dalle configurazioni abilitate. Cfr. Configurare le notifiche di RM

WfStartLogicIdentifier

Questa logica viene eseguita in fase di creazione di un nuovo oggetto. Viene effettuata la costruzione dell'identificativo utilizzando la logica definita nella variabile di configurazione `ap.<wfItemType>.identifier.logic configuration`. Se questa non è definita viene supposto l'utilizzo di `wfStartLogicStandardIdentifier`

Le logiche utilizzabili al momento sono:

- `wfStartLogicStandardIdentifier`
- `wfStartLogicYearFromProposalStartDateIdentifier`
- `wfStartLogicOwnerDependentItemTypeFirstIdentifier`
- `wfStartLogicOwnerDependentLastNameFirstIdentifier`
- `wfStartLogicOwnerDependentLastNameFirstIdentifierAndAcronym`
- `wfStartLogicDepartmentAndOwnerDependentIdentifier`
- `wfStartLogicProjectWithYearCallIdentifier`

Configurazioni della StartLogic:

- `ap.TIPOLOGIA.identifier.logic`
var sourceItemType = wfService.getWfItemType(wfItem.getWfItemTypeId());

var identifyLogic = Packages.it.cilea.wf.util.WfUtil.getParametricConfiguration("ap.<wfItemType>.identifier.logic", sourceItemType, null);

if(identifyLogic != null) {
    var startLogic = Packages.it.cilea.wf.WfConstant.START_LOGIC_MAP.get(identifyLogic);
    startLogic.start(wfItem, request);
} else {
    throw "To use wfStartLogicIdentifier you MUST specify ap.<wfItemType>.identifier.logic configuration";
}

wfStartLogicStandardIdentifier
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
Viene effettuata la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:

* CODE_TIPOLOGIA: come da campo CODE da tassonomia scaricata dalla sezione "Alberatura tipologie RM/AP": se si applica questa logica di costruzione identificativo evitare che il CODE abbia il carattere -, non può essere creata su DB la sequenza corrispondente per il progressivo
* PROGRESSIVO: numero progressivo

Esempio di costruzione dell'identificativo PEN-0003.
Per maggiori dettagli cfr. modello dati dell'entità Progetto e l'alberatura delle tipologie.

function buildIdentifier(counter, identifierCode) {
    var counterString;
    if (counter < 10)
        counterString = "000" + counter;
    else if (counter < 100)
        counterString = "00" + counter;
    else if (counter < 1000)
        counterString = "0" + counter;
    else
        counterString = "" + counter;
    identifierCode += "-" + counterString;
    return identifierCode;
}

var identifierCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();
var identifier;
if (wfItem.getIdentifier() == null ||
    (wfItem.getIdentifier().lastIndexOf("-")) >= 0 &&
    !wfItem.getIdentifier().substring(0, wfItem.getIdentifier().lastIndexOf("-"))
    .equals(identifierCode)) {
    var sequenceName = "item_" + identifierCode + ".seq";
    var counter = wfService.updateSequence(sequenceName);
    identifier = buildIdentifier(counter, identifierCode);
    var isPresent = wfService.identifierIsPresent(identifier);
    if (Packages.java.lang.Boolean.TRUE.equals(isPresent)) {
        var maxValSequence = wfService.getMaxSequencesValueFromIdentifier(identifierCode + "-"));
        maxValSequence++;
        counter = wfService.updateSequenceFromNumber(sequenceName, maxValSequence);
        identifier = buildIdentifier(maxValSequence, identifierCode);
    }
    wfItem.setIdentifier(identifier);
}

wfStartLogicYearFromProposalStartDateIdentifier
Questa logica viene eseguita in fase di creazione di un nuovo progetto.
Viene effettuata la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:

* ANNO: estratto dalla Data di avvio pratica (proposalStartDate)
Esempio di costruzione dell'identificativo 2021-AL-0002.
Per maggiori dettagli cfr. modello dati dell'entità Progetto e l'alberatura delle tipologie.

```javascript
function buildIdentifier(counter, identifierCode) {
    var counterString;
    if (counter < 10)
        counterString = "000" + counter;
    else if (counter < 100)
        counterString = "00" + counter;
    else if (counter < 1000)
        counterString = "0" + counter;
    else
        counterString = "" + counter;
    identifierCode += "-" + counterString;
    return identifierCode;
}

var startDate = wfItem.getDateMap().get("proposalStartDate");
var year = startDate.get(Packages.java.util.Calendar.YEAR);
var yearString = new Packages.java.lang.Integer(year).toString();
var sourceItemTypeCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();
var stringBuilder = new Packages.java.lang.StringBuilder(yearString);
stringBuilder.append("-");
stringBuilder.append(sourceItemTypeCode);
var identifierCode = stringBuilder.toString();
var identifier;
if(wfItem.getIdentifier() == null || !(wfItem.getIdentifier().substring(0, wfItem.getIdentifier().lastIndexOf("-"))).equals(identifierCode)){
    var sequenceName = "item_" + sourceItemTypeCode + "_" + year + "_seq";
    var counter = wfService.updateSequence(sequenceName);
    identifier = buildIdentifier(counter, identifierCode);
    var isPresent = wfService.identifierIsPresent(identifier);
    if (Packages.java.lang.Boolean.TRUE.equals(isPresent)) {
        var maxNumber = wfService.getMaxSequencesValueFromIdentifier(identifierCode + "-");
        maxNumber++;
        counter = wfService.updateSequenceFromNumber(sequenceName, maxNumber);
        identifier = buildIdentifier(counter, identifierCode);
    }
    wfItem.setIdentifier(identifier);
}
```

**wfStartLogicYearFromStartDate**
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
Viene recuperato l'anno dalla configurazione fatta nel seguente modo: `ap.-"rootWfItemType"-owner.position.date` e viene salvato nel metadato `year`
Esempio: `ap.prj.owner.position.date`

**Configurazioni della StartLogic:**
- `ap.TIPOLOGIA.owner.position.date`

```javascript
var myDate=WfUtil.getCheckDate(wfItem, "owner", wfService);
if(myDate == null)
    throw "The provided date MUST NOT be null";
var calendar = Calendar.getInstance();
calendar.setTime(myDate);
wfItem.setYear(new Packages.java.lang.Integer(calendar.get(Calendar.YEAR)));
```
Questa logica effettua la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:

- **CODE_TIPOLOGIA**: come da campo CODE da tassonomia scaricata dalla sezione "Alberatura tipologie RM/AP"
- **ANNO**: anno in due cifre estratto dalla data di inizio
- **PRIMA_LETTERA_NOME**: prima lettera del nome del responsabile
- **PRIME_QUATTRO_LETTERE_COGNOME**: prime quattro lettere del cognome del responsabile
- **PROGRESSIVO**: numero progressivo

Esempio di costruzione dell'identificativo **H202019IABBA_01**.
Per maggiori dettagli cfr. modello dati dell'entità Progetto e l'alberatura delle tipologie.

```java
var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem,
  "it.cilea.wf.model.WfItemElement","owner", wfService);
  var person = Packages.it.cilea.wf.util.WfUtil.getOwnerForIdentifier(ownerWfElementSet,
    wfService);
  var startDate = wfItem.getDateMap().get("proposalStartDate");
  var year = startDate.get(Packages.java.util.Calendar.YEAR);
  var yearString = new Packages.java.lang.Integer(year).toString();
  yearString = yearString.substring(2);
  var sourceItemTypeCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();
  var lastName = person.getLastName().replaceAll("\s", "");
  lastName = lastName.replaceAll("'", "");
  lastName = (lastName.length() >= 4) ? lastName.substring(4,4) : lastName.substring(0,
    lastName.length());
  lastName = lastName.toUpperCase();
  var firstName = person.getFirstName().substring(0,1);
  firstName = firstName.toUpperCase();
  var stringBuilder = new Packages.java.lang.StringBuilder(sourceItemTypeCode);
  stringBuilder.append(yearString);
  stringBuilder.append(firstName);
  stringBuilder.append(lastName);
  var identifier = stringBuilder.toString();
  if(wfItem.getIdentifier() == null || !(wfItem.getIdentifier().substring(0, wfItem.
    getIdentifier().lastIndexOf("_"))).equals(identifier)){
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("identifier", identifier);
    paramMap.put("excludeIdentifier", identifier+"M");
    var count = searchBuilderTemplate.getSingleObject
      ("getMaxItemCountForGivenIdentifier", paramMap);
    var countString = "0";
    if (count < 9)
      countString += "0";
    countString += (count.intValue()+1);
    identifier+= "_" + countString;
    wfItem.setIdentifier(identifier);
  }
```
var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","owner", wfService);
var person = Packages.it.cilea.wf.util.WfUtil.getOwnerForIdentifier(ownerWfElementSet, wfService);

var lastName = person.getLastName().replaceAll("\s", "");
lastName = lastName.replaceAll("", "");
lastName = (lastName.length() >= 3) ? lastName.substring(0,4) : lastName.substring(0, lastName.length());
lastName = lastName.toUpperCase();

var firstName = person.getFirstName().substring(0,1);
firstName = firstName.toUpperCase();

var dateDiscriminators = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.identifier.logic.date.discriminators");
var discriminator = "proposalStartDate";
if(dateDiscriminators != null){
    var discriminatorArray = dateDiscriminators.split(",");
    for(i in discriminatorArray){
        if(wfItem.getDateMap().get(discriminatorArray[i]) != null){
            discriminator = discriminatorArray[i];
            break;
        }
    }
}

var startDate = wfItem.getDateMap().get(discriminator);
var year = startDate.get(Packages.java.util.Calendar.YEAR);
var yearString = new Packages.java.lang.Integer(year).toString();
yearString = yearString.substring(2);

var sourceItemTypeCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();
var acronym = wfItem.getStringMap().get("acronym");

var stringBuilder = new Packages.java.lang.StringBuilder(lastName);
stringBuilder.append("_");
stringBuilder.append(firstName);
stringBuilder.append("_");
stringBuilder.append(yearString);
stringBuilder.append("_");
stringBuilder.append(sourceItemTypeCode);
if(acronym != null){
    acronym = acronym.replaceAll("\s", "");
    acronym = acronym.toUpperCase();
    stringBuilder.append("_");
    stringBuilder.append(acronym);
}

var identifier = stringBuilder.toString();
identifier = (identifier.length() > 46) ? identifier.substring(0,46) : identifier.substring(0,identifier.length());

if(wfItem.getIdentifier() == null || !(wfItem.getIdentifier().substring(0, wfItem.getIdentifier().lastIndexOf("_"))) .equals(identifier)){
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("identifier", identifier);
    paramMap.put("excludeIdentifier", identifier + "M");
    var count = searchBuilderTemplate.getSingleObject("getMaxItemCountForGivenIdentifier", paramMap);
    var countString = ";
    if (count < 9)
        countString += "0"
    countString += (count.intValue()+1);
    identifier += ";" + countString;
    wfItem.setIdentifier(identifier);
}
Questa logica effettua la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:

<VALORE_PERSONALIZZABILE>-<CODICE_TIPO_PROGETTO>-<ULTIME_DUE_CIFRE_ANNO>-<PRIME_5_LETTERE_COGNOME>_<PROGRESSIVO>

- **VALORE_PERSONALIZZABILE:**
  - CODICE_CSA_DEL_DIPARTIMENTO: codice del dipartimento recuperato nel metadato `orgUnit.stringMap[idCsa]`
  - CODICE_IDAB_DEL_DIPARTIMENTO: codice del dipartimento recuperato nel metadato `orgUnit.stringMap[idAb]`
  - ID_DEL_DIPARTIMENTO: identificativo univoco del dipartimento recuperato nel metadato `orgUnit.id`
  - CODICE_ACRONYM_DEL_DIPARTIMENTO: codice del dipartimento recuperato nel metadato `orgUnit.stringMap[acronym]`

- **CODICE_TIPO_PROGETTO:** come da campo CODE da tassonomia scaricata dalla sezione "Alberatura tipologie RM/AP"

- **ULTIME_DUE_CIFRE_ANNO:** anno in due cifre estratto dalla data di inizio

- **PRIME_5_LETTERE_COGNOME:** prime cinque lettere del cognome del responsabile

- **PROGRESSIVO:** numero progressivo

Per poter personalizzare il valore occorre valorizzare (o creare) la variabile di configurazione `ap.prj.identifier.logic.department-id`. Se la variabile di configurazione è valorizzata con:

- `csa` verrà utilizzato il metadato `orgUnit.stringMap[idCsa]`
- `ugov` verrà utilizzato il metadato `orgUnit.stringMap[idAb]`
- `iris` verrà utilizzato il metadato `orgUnit.id` del dipartimento
- `acronym` verrà utilizzato il metadato `orgUnit.stringMap[acronym]`

Se la configurazione non è valorizzata o è valorizzata con un valore diverso da quelli previsti, verrà usato `stringMap[idCsa]` estratto dal dipartimento.

**Configurazioni della StartLogic:**

- `ap.prj.identifier.logic.department-id`

```javascript
var departmentIdConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.identifier.logic.department-id");
departmentIdConf = (!departmentIdConf) ? "csa" : departmentIdConf;
var departmentId = null;
var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "owner", wfService);
var person = Packages.it.cilea.wf.util.WfUtil.getOwnerForIdentifier(ownerWfElementSet, wfService);
if (person != null) {
    var checkDate = WfUtil.getCheckDate(wfItem, "owner", wfService);
    var positionContext = WfUtil.getPositionContext(wfItem, wfService);
    var ownerPositionSet = WfUtil.getPositionSet(person, checkDate, positionContext, "department", gaService);
    var department = GaUtil.getPriorityOrganizationUnit(ownerPositionSet, positionContext, "department");
    if (department == null) {
        throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier you MUST specify department for that person ";
    } else {
        throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier you MUST specify department";
    }
} else {
    throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier you MUST specify department";
}
switch("" + String(departmentIdConf)){
    case "csa":
        departmentId = department.getStringMap().get("idCsa");
        if (departmentId == null) {
            throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier you MUST specify code department [idCsa]";
        }
        break;
    case "ugov":
        departmentId = department.getStringMap().get("idAb");
        if (departmentId == null) {
            throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier you MUST specify code department [idAb]";
        }
        break;
    case "iris":
        departmentId = String(department.getId());
        break;
    default:
        break;
}
```
break;
case "acronym":
    departmentId = department.getStringMap().get("acronym");
    if (departmentId == null) {
        throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier
you MUST specify code department [acronym]");
    } break;
default:
    departmentId = department.getStringMap().get("idCsa");
    if (departmentId == null) {
        throw "To use wfStartLogicDepartmentAndOwnerDependentIdentifier
you MUST specify code department [idCsa]");
    }
}

var lastName = person.getLastName().replaceAll("\s", "");
lastName = lastName.replaceAll("'", "");
lastName = (lastName.length() >= 4) ? lastName.substring(0, 5) : lastName.substring(0, lastName.length());
lastName = lastName.toUpperCase();

var dateDiscriminators = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.identifier.logic.date.discriminators");
var discriminator = "proposalStartDate";
if (dateDiscriminators != null) {
    var discriminatorArray = dateDiscriminators.split(",");
    for (i in discriminatorArray) {
        if (wfItem.getDateMap().get(discriminatorArray[i]) != null) {
            discriminator = discriminatorArray[i];
            break;
        }
    }
}

var startDate = wfItem.getDateMap().get(discriminator);
var year = startDate.get(Packages.java.util.Calendar.YEAR);
var yearString = new Packages.java.lang.Integer(year).toString();
yearString = yearString.substring(2);
var sourceItemTypeCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();

var stringBuilder = new Packages.java.lang.StringBuilder(departmentId);
stringBuilder.append(sourceItemTypeCode);
stringBuilder.append(yearString);
stringBuilder.append(lastName);
var identifier = stringBuilder.toString();
if (wfItem.getIdentifier() == null || !wfItem.getIdentifier().substring(0, wfItem.getIdentifier().lastIndexOf("_"))).equals(identifier)) {
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("identifier", identifier);
    paramMap.put("excludeIdentifier", identifier+"M");
    var count = searchBuilderTemplate.getSingleObject("getMaxItemCountForGivenIdentifier", paramMap);
    var countString = "";
    if (count < 9)
        countString += "0";
    countString += (count.intValue()+1);
    identifier += "_" + countString;
    wfItem.setIdentifier(identifier);
}

**wfStartLogicOwnerDependentLastNameFirstIdentifier**
Questa logica effettua la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:

<CODE_TIPOLOGIA>-<ANNO>-<PRIMA_LETTERA_NOME>-<PRIME_QUATTRO_LETTERE_COGNOME>-<PROGRESSIVO>

- **PRIME_QUATTRO_LETTERE_COGNOME**: prime quattro lettere del cognome del responsabile
- **PRIMA_LETTERA_NOME**: prima lettera del nome del responsabile
Esempio di costruzione dell'identificativo **ROSS_P_21_PRJ_21_01**.
Per maggiori dettagli cfr. modello dati dell'entità Progetto e l'alberatura delle tipologie.

```java
var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","owner", wfService);
var person = Packages.it.cilea.wf.util.WfUtil.getOwnerForIdentifier(ownerWfElementSet, wfService);

var dateDiscriminators = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.identifier.logic.date.discriminators");
var discriminator = "proposalStartDate";
if(dateDiscriminators != null){
    var discriminatorArray = dateDiscriminators.split(",");
    for(i in discriminatorArray){
        if(wfItem.getDateMap().get(discriminatorArray[i]) != null){
            discriminator = discriminatorArray[i];
            break;
        }
    }
}

var startDate = wfItem.getDateMap().get(discriminator);
var year = startDate.get(Packages.java.util.Calendar.YEAR);
var yearString = new Packages.java.lang.Integer(year).toString();
yearString = yearString.substring(2);
var sourceItemTypeCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();
var lastName = person.getLastName().replaceAll("\s", "");
lastName = lastName.replaceAll("'", "");
lastName = (lastName.length() >= 3) ? lastName.substring(0,3) : lastName.substring(0,lastName.length());
lastName = lastName.toUpperCase();
var firstName = person.getFirstName().substring(0,1);
firstName = firstName.toUpperCase();
var stringBuilder = new Packages.java.lang.StringBuilder(lastName);
stringBuilder.append(firstName);
stringBuilder.append('_');
stringBuilder.append(sourceItemTypeCode);
stringBuilder.append('_');
stringBuilder.append(yearString);
var identifier = stringBuilder.toString();

if(wfItem.getIdentifier() == null | !(wfItem.getIdentifier().substring(0, wfItem.getIdentifier().lastIndexOf('_'))).equals(identifier)){
    var paramMap = new Packages.java.util.HashMap();
    paramMap.put("identifier", identifier);
    paramMap.put("excludeIdentifier", identifier+"M");
    var count = searchBuilderTemplate.getSingleObject("getMaxItemCountForGivenIdentifier", paramMap);
    var countString = "";
    if (count < 9){
        countString += "0";
    }
    countString += (count.intValue()+1);
    identifier += "_" + countString;
    wfItem.setIdentifier(identifier);
}
```

**wfStartLogicProjectWithYearCallIdentifier**

Questa logica effettua la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:

```
<CODE_TIPOLOGIA>-<ANNO>-<ACRONIMO>-<PRIME_SEI_LETTERE_COGNOME>-<PRIMA_LETTERA_NOME>-<PROGRESSIVO>
```

- **CODE_TIPOLOGIA**: come da campo CODE da tassonomia scaricata dalla sezione "Alberatura tipologie RM/AP" recuperata a partire dal Bando di finanziamento o direttamente da selezione in fase di creazione
Esempio di costruzione dell'identificativo **SCAVI_2022_TEST_ROSSI_M_13**.
Per maggiori dettagli cfr. modello dati dell'entità Progetto e l'alberatura delle tipologie.

```java
//IMPORTANTE: questa startLogic deve essere posizionata PRIMA di
wfStartLogicProjectTypeFromCall
    var ownerWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
    (wfItem, "it.cilea.wf.model.WfItemElement","owner", wfService);
    var person = Packages.it.cilea.wf.util.WfUtil.getOwnerForIdentifier(ownerWfElementSet,
    wfService);
    var callProjectLinkWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
    (wfItem, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink",
    "callProjectLink", wfService);
    var callProjectLinkSetIterator = callProjectLinkWfElementSet.iterator();
    var sourceItemTypeCode = wfService.getWfItemType(wfItem.getWfItemTypeId()).getCode();
    var yearString = "XX";
    var callId = wfItem.getIntegerMap().get("callId");
    // cerco di recuperare prima integerMap["callId"] perché in fase di creazione l'element
callProjectLink non restituisce alcun valore
    if(callId != null){
        var call=wfService.getWfItem(callId);
        if (call==null)
            throw "The wfStartLogicProjectWithYearCallIdentifier encountered an
error in use the element integerMap["callId"]";
        var startDate = call.getDateMap().get("startDate");
        var year = startDate.get(Packages.java.util.Calendar.YEAR);
        yearString = new Packages.java.lang.Integer(year).toString();
    } else {
        if(callProjectLinkSetIterator.hasNext()){ 
            var callProjectLink = callProjectLinkSetIterator.next();
            var call = callProjectLink.getParent();
            if (call==null)
                throw "The wfStartLogicProjectWithYearCallIdentifier
encountered an error in use the element callProjectLink";
            var startDate = call.getDateMap().get("startDate");
            var year = startDate.get(Packages.java.util.Calendar.YEAR);
            yearString = new Packages.java.lang.Integer(year).toString();
            var projectTypeIdentifier=call.getStringMap().get("linkedProjectType");
            var wfItemTypeList=wfService.getWfItemTypeByIdentifier
            (projectTypeIdentifier, 0);
            if (CollectionUtils.isNotEmpty(wfItemTypeList)){
                sourceItemTypeCode = wfItemTypeList.get(0).getCode();
            } 
        } else {
            if(callProjectLinkSetIterator.hasNext()){
                var callProjectLink = callProjectLinkSetIterator.next();
                var call = callProjectLink.getParent();
                if (call==null)
                    throw "The wfStartLogicProjectWithYearCallIdentifier
encountered an error in use the element callProjectLink";
                var startDate = call.getDateMap().get("startDate");
                var year = startDate.get(Packages.java.util.Calendar.YEAR);
                yearString = new Packages.java.lang.Integer(year).toString();
                var projectTypeIdentifier=call.getStringMap().get("linkedProjectType");
                var wfItemTypeList=wfService.getWfItemTypeByIdentifier
                (projectTypeIdentifier, 0);
                if (CollectionUtils.isNotEmpty(wfItemTypeList)){
                    sourceItemTypeCode = wfItemTypeList.get(0).getCode();
                } 
            } 
        } 
    } 
    var acronym = wfItem.getStringMap().get("acronym");
    var lastName = person.getLastName().replaceAll("\s", "");
    lastName = lastName.replaceAll("'", "");
    lastName = (lastName.length() >= 6) ? lastName.substring(0,6) : lastName.substring(0,
    lastName.length());
    lastName = lastName.toUpperCase();
    var firstName = person.getFirstName().substring(0,1);
    firstName = firstName.toUpperCase();
    var stringBuilder = new Packages.java.lang.StringBuilder(sourceItemTypeCode);
```
wfStartLogicAcademicField2000
Questa logica viene eseguita in fase di creazione di un nuovo oggetto. Recupera il responsabile scientifico (owner), estrae il SSD relativo all’ultimo rapporto di lavoro valido prioritario relativo al contesto ricerca ed aggancia questa informazione all’oggetto radice.

```java
var person=null;
person=wfItem.getPersonMap().get("owner");
var person2 = gaService.getPerson(person.getId());
var positionLastSet = person2.getPositionLastSet();
var maxPriority2 = Packages.it.cilea.ga.util.GaUtil.getMaxPriority(positionLastSet, "research", "academicField2000");
positionLastSetIterator=positionLastSet.iterator();
while (positionLastSetIterator.hasNext()){ 
    var position=positionLastSetIterator.next();
    if (maxPriority2 == null || maxPriority2.equals(position.getPriority())){
        if ("research").equals(position.getDiscriminator()) &
    
    getOrganizationUnitType().getDescription())

    WfItemElement();
{"academicField2000"};
{"organizationUnit", position.getOrganizationUnit()};

    ssdElement.setDiscriminator
    ssdElement.setWfItemId(wfItem.getId());
    ssdElement.getOrganizationUnitMap().put
   _wfService.saveOrUpdate(ssdElement);
    wfItem.getWfItemElementSet().add(ssdElement);
    break;
```
**wfStartLogicMultipleOwners**

Questa logica viene eseguita in fase di creazione di un nuovo oggetto.

SE l'attributo "Struttura capofila diversa da afferenza del responsabile?" (booleanMap[customizedInternalOrganization]) è settato a **true** la logica recupera la Struttura presente nel relativo campo (organizationUnitMap[internalOrganizationUnit]) e l'aggiunge alle Strutture interne (all'oggetto radice *internalOrganization*).

SE l'attributo "Struttura capofila diversa da afferenza del responsabile?" (booleanMap[customizedInternalOrganization]) è settato a **false** la logica effettua il setup del responsabile scientifico (*owner*) ed estrae la sua afferenza dipartimentale (se presente) che aggancia alle Strutture interne (all'oggetto radice *internalOrganization*).

Il ruolo assegnato alla Struttura interna viene valorizzato di default, e viene preso dalla configurazione ap.<tipologia oggetto radice>.<tipologia oggetto fragment>.role.main.

La data alla quale estrarre l'afferenza dipartimentale del responsabile è quella contenuta nel metadato con il nome recuperabile dalla configurazione ap.<tipologia>.owner.position.date.

E' possibile decidere se settare in automatico il metadato startDate per il responsabile.

Se la configurazione ap.<tipologia>.owner.startDate.enabled:

- NON è presente, allora NON viene abilitato l'automatismo di recupero della startDate
- è uguale a **true** allora NON viene abilitato l'automatismo di recupero della startDate
- altrimenti NON viene abilitato l'automatismo di recupero della startDate

E' inoltre possibile stabilire quali siano i contesti all'interno dei quali cercare l'afferenza dipartimentale per il responsabile scientifico.

Questo viene pilotato dalla configurazione ap.<tipologia>.owner.position.context.

Se questa configurazione

- NON è presente, viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a research allora viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a support allora viene usato il solo contesto di supporto (TA)
- è uguale a ALL allora vengono usati entrambi i contesti

E' possibile definire la variabile ap.<tipologia>.internalOrganizationUnit.inferredChidCsv.

Questa variabile, se definita, contiene l'elenco, separato da virgola, degli elementi (recuperabili dal modello dati) da creare agganciati alla struttura estratta dal responsabile.

Consideriamo questo esempio.

Se la variabile ap.prj.internalOrganizationUnit.inferredChidCsv è valorizzata con *internalOrganizationUnitCost* questo vorrà dire che verrà creato un elemento di questo tipo associato alla struttura responsabile scientifico.

Fare attenzione che per usare questa funzione l'elemento inferito (nell'esempio *internalOrganizationUnitCost*) deve prevedere nel suo modello dati l'attributo *ouId* che è l'attributo che lega questo elemento alla struttura del responsabile.

Con <tipologia> viene inteso il codice a tre lettere in minuscolo che indica la tipologia radice dell'oggetto in questione come ad esempio: lab, eqp, inm, prj, ...

Per un elenco esaustivo fare riferimento all’"Alberatura tipologie RM/AP"

**Configurazioni della StartLogic:**

- ap.prj.internalOrganizationUnit.inferredChidCsv
- ap.prj.owner.customizableAutomaticInternalOrganizationExtractionInCreationPhase.enabled
- ap.TIPOLOGIA.FRAGMENT.role.main
- ap.TIPOLOGIA.owner.department.enabled
- ap.TIPOLOGIA.owner.position.context

```javascript
var person = wfItem.getPersonMap().get("owner");
if (person != null) {
  var checkDate = WfUtil.getCheckDate(wfItem, "owner", wfService);
  var positionContext = WfUtil.getPositionContext(wfItem, wfService);
  var ownerPositionSet = WfUtil.getPositionSet(person, checkDate,
positionContext, "department", gaService);
  var department = GaUtil.getPriorityOrganizationUnit(ownerPositionSet,
  positionContext, "department");
  var departmentCustom = wfItem.getOrganizationUnitMap().get
```
var customizedInternalOrganization = wfItem.getBooleanMap().get("customizedInternalOrganization");

var wfItemType = wfService.getWfItemType(wfItem.getWfItemTypeId());
var rootWfItemType = wfItemType.getRootWfItemType();
var departmentSpecifiedInOwnerElement = ConfigurationUtil.getConfigValue("ap." + rootWfItemType.getIdentifier().toLowerCase() + ".owner.department.enabled");

var ownerElement = new WfItemElement();
ownerElement.setDiscriminator("owner");
ownerElement.setWfItemId(wfItem.getId());
ownerElement.getPersonMap().put("ownerId", person);

if (WfUtil.isElementStartDateAutomaticPopulationEnabled(wfItem, "owner", wfService) && checkDate != null) {
    var calendar=Calendar.getInstance();
    calendar.setTime(checkDate);
    ownerElement.getDateMap().put("startDate", calendar);
} else if (departmentSpecifiedInOwnerElement=="true" && department!=null) {
    ownerElement.getOrganizationUnitMap().put("ouId", department);
} else if (departmentSpecifiedInOwnerElement=="true" && departmentCustom!=null && customizedInternalOrganization!=true) {
    ownerElement.getOrganizationUnitMap().put("ouId", departmentCustom);
}

wfService.saveOrUpdate(ownerElement);
wfItem.getWfItemElementSet().add(ownerElement);
wfItem.getPersonMap().put("owner", null);

if (department!=null || departmentCustom!=null) {
    var internalOrganizationUnitElement=new WfItemElement();
    internalOrganizationUnitElement.setDiscriminator("internalOrganizationUnit");
    internalOrganizationUnitElement.setWfItemId(wfItem.getId());
    if (department!=null && customizedInternalOrganization!=true) {
        internalOrganizationUnitElement.getOrganizationUnitMap().put("ouId", department);
    } else if (departmentCustom!=null && customizedInternalOrganization==true) {
        internalOrganizationUnitElement.getOrganizationUnitMap().put("ouId", departmentCustom);
    }
}

if (WfUtil.isElementStartDateAutomaticPopulationEnabled(wfItem, "internalOrganizationUnit", wfService)) {
    var checkDate = WfUtil.getCheckDate(wfItem, "internalOrganizationUnit", wfService);
    if (checkDate != null) {
        var calendar=Calendar.getInstance();
        calendar.setTime(checkDate);
        internalOrganizationUnitElement.getDateMap().put("startDate", calendar);
    }
}

wfService.saveOrUpdate(internalOrganizationUnitElement);
wfItem.getWfItemElementSet().add(internalOrganizationUnitElement);
WfUtil.addElementInferredFromOrgUnitElement(internalOrganizationUnitElement, wfService, gaService, null);
**wfStartLogicPartner**

Questa logica viene eseguita in fase di creazione di un nuovo progetto.

Viene eseguito l'inserimento dell'Ateneo tra i partner.

La variabile di configurazione `ap.prj.partner.role.autopopulate.enabled`, attiva di default, gestisce l'inserimento del ruolo associato al Partner.

La voce di dizionario che definisce il ruolo da inserire è specificata nella variabile `ap.prj.partner.role.autopopulate.code`.

Per maggiori dettagli cfr. excel modello dati dell'entità di riferimento

**Configurazioni della StartLogic:**

- `ap.prj.partner.role.autopopulate.enabled`
- `ap.prj.partner.role.autopopulate.code`

```java
var myOrganization = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("rm.orgunit.external.myOrganization");
if (myOrganization == null){
    throw "Configuration variable rm.orgunit.external.myOrganization MUST BE DEFINED";
} else {
    var partnerElement = new Packages.it.cilea.wf.model.WfItemElement();
    partnerElement.setDiscriminator("partner");
    partnerElement.setWfItemId(wfItem.getId());
    var partnerRoleAutopopulate = ConfigurationUtil.getConfigValue("ap.prj.partner.role.autopopulate.enabled");
    var partnerRoleCode = ConfigurationUtil.getConfigValue("ap.prj.partner.role.autopopulate.code");
    if (!partnerRoleAutopopulate || "true" == partnerRoleAutopopulate){
        if (partnerRoleCode){
            var uniquePartnerRoleId = WfUtil.getDictionaryByCode(wfService, partnerRoleCode);
            if (uniquePartnerRoleId == null){
                throw "Dictionary code " + partnerRoleCode + " MUST BE DEFINED";
            } else {
                uniquePartnerRoleId = wfService.getWfDictionary(new Packages.java.lang.Integer(uniquePartnerRoleId));
            }
        } else {
            throw "Configuration variable ap.prj.partner.role.autopopulate.code MUST BE DEFINED";
        }
    }
    myOrganization = gaService.getOrganizationUnit(new Packages.java.lang.Integer(myOrganization));
    partnerElement.getOrganizationUnitMap().put("partnerId", myOrganization);
    wfService.saveOrUpdate(partnerElement);
}
```

**wfStartLogicMultipleInternalOrganizationUnits**

Questa logica viene eseguita in fase di creazione di un nuovo oggetto.

Aggiunge l’unità organizzativa (internalOrganizationUnit) all’oggetto che si sta creando utilizzando l’unità organizzativa specificata in fase di creazione (organizationUnitMap) e l’unità organizzativa (internalOrganizationUnit) all’oggetto che si sta creando utilizzando l’unità organizzativa specificata in fase di creazione (organizationUnitMap).

Aggiunge anche, se previsto il ruolo di default dell’unità organizzativa, se viene trovata la configurazione `ap.<tipologia>.internalOrganizationUnit.role.main`, dove `tipologia` indica il codice della tipologia, che è navigabile fino alla radice.
Ad esempio se la configurazione ap.pen.formazione-comunicazione.internalOrganizationUnit.role.main è settata a 15, vuol dire che il ruolo con id 15 è quello che verrà associato all'unità organizzativa selezionata nel caso di iniziative di public engagement di formazione e comunicazione.

Se la configurazione fosse ap.pen.internalOrganizationUnit.role.main (e non esistesse quella più specifica presentata sopra) e fosse settata a 15, vuol dire che il ruolo con id 15 è quello che verrà associato all'unità organizzativa selezionata nel caso di tutte le iniziative di public engagement.

Configurazioni della StartLogic:
- ap.TIPOLOGIA.internalOrganizationUnit.role.main

```java
// caricamento internalOrganizationUnit
var internalOrganizationUnit = wfItem.getOrganizationUnitMap().get("ouId");
if (internalOrganizationUnit != null)
    internalOrganizationUnit = gaService.getOrganizationUnit(internalOrganizationUnit.getId());
var internalOrganizationUnitElement = new Packages.it.cilea.wf.model.WfItemElement();
internalOrganizationUnitElement.setDiscriminator("internalOrganizationUnit");
internalOrganizationUnitElement.setWfItemId(wfItem.getId());
internalOrganizationUnitElement.getOrganizationUnitMap().put("ouId", internalOrganizationUnit);
Packages.it.cilea.wf.util.WfUtil.addRoleInElementIfDefined(wfItem, internalOrganizationUnitElement, wfService);
wfService.saveOrUpdate(internalOrganizationUnitElement);
wfItem.getWfItemElementSet().add(internalOrganizationUnitElement);
wfItem.getOrganizationUnitMap().put("ouId", null);
```

wfStartLogicAdministrativeOwnerFromUser
Questa logica viene eseguita in fase di creazione di un nuovo oggetto. Aggiunge il referente amministrativo all'oggetto che si sta creando.

Precisamente:
- Viene recuperata la macrotipologia dell'oggetto che si sta creando (macrotype)
- Viene recuperata la variabile di configurazione ap.<macrotype>.administrativeOwner.inheritFromLoggedUserIfResourcePresent
- Questa configurazione, se presente, contiene le risorse relative ai profili per i quali inserire come referente amministrativo dell'oggetto, l'utente corrente.

Ad esempio supponiamo la variabile di conf ap.prj.administrativeOwner.inheritFromLoggedUserIfResourcePresent=/DEPARTMENT.profile, /RESEARCH_DIVISION.profile.
Questo vuol dire che se un utente che appartiene a un team con profili dipartimentali o divisione ricerca effettua la creazione di un oggetto, verrà inserito lui stesso come referente amministrativo. Per maggiori dettagli cfr. modello dati dell'entità in questione.

Configurazioni della StartLogic:
- ap.TIPOLOGIA.administrativeOwner.inheritFromLoggedUserIfResourcePresent

```java
var typeForConfig;
var itemType = wfService.getWfItemType(wfItem.getWfItemTypeId());
if (!itemType.getParentWfItemTypeSet().isEmpty()) {
    var parentSetIterator = itemType.getParentWfItemTypeSet().iterator();
    while (parentSetIterator.hasNext()){
        var parent = parentSetIterator.next();
        if (!parent.getParentWfItemTypeSet().isEmpty()) {
            var parentParentSetIterator = parent.getParentWfItemTypeSet().iterator();
            while (parentParentSetIterator.hasNext()){ [continue]
                var parentParent = parentParentSetIterator.next();
                typeForConfig = parentParent; [break]
                break;
            }
        }
        break;
    }
}
```

var profiles = '';
if(typeForConfig.getIdentifier() == 'CN' || typeForConfig.getIdentifier() == 'PRJ'){
    profiles = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap."+ typeForConfig.getIdentifier().toLowerCase() +".administrativeOwner.inheritFromLoggedUserIfResourcePresent");
}
var profileArray = profiles.split(",");
var gaUserDetail = Packages.it.cilea.ga.authorization.context.
GaAuthorizationUserHolder.getUser();
for (i in profileArray) {
  if (gaUserDetail.hasAuthorities(profileArray[i])) {
    var administrativeOwnerElement = new Packages.it.cilea.wf.
    model.WfItemElement();
    administrativeOwnerElement.setDiscriminator("administrativeOwner");
    administrativeOwnerElement.setWfItemId(wfItem.getId());
    var person = gaService.getPerson(gaUserDetail.getPersonId())
    administrativeOwnerElement.getPersonMap().put("administrativeOwner", person);
    wfService.saveOrUpdate(administrativeOwnerElement);
    wfItem.getWfItemElementSet().add(administrativeOwnerElement);
    break;
  }
}
true;

wfStartLogicInternalRepresentativeFromUser
Questa logica viene eseguita in fase di creazione di un nuovo oggetto e aggiunge l'elemento delegato (internalRepresentative) all'oggetto che si sta creando.
La logica è abilitata solo se
- la variabile di configurazione \texttt{ap.<macrotype>.personal-view.create-for-other-people.enabled} è valorizzata a \texttt{true}, dove \texttt{<macrotype>} indica la macrotipologia dell'oggetto che si sta creando.
- l'attributo booleano \texttt{isCreatorOwner}, specificato in fase di creazione, è valorizzato a \texttt{false}
Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configuration della StartLogic:
- \texttt{ap.TIPOLOGIA.personal-view.create-for-other-people.enabled}

var isCreatorOwner = wfItem.getBooleanMap().get("isCreatorOwner");
var wfItemType = wfService.getWfItemType(wfItem.getWfItemType());
var rootWfItemType = wfItemType.getRootWfItemType();
var createForOtherPeopleEnabled = ConfigurationUtil.getConfigValue("ap." +
  rootWfItemType.getIdentifier().toLowerCase() + ".personal-view.
  create-for-other-people.enabled");
if (createForOtherPeopleEnabled == "true" && Packages.
  java.lang.Boolean.FALSE.equals(isCreatorOwner)) {
  var gaUserDetail = Packages.it.cilea.ga.authorization.context.
  GaAuthorizationUserHolder.getUser();
  var internalRepresentativeElement = new Packages.it.cilea.wf.
  model.WfItemElement();
  internalRepresentativeElement.setDiscriminator("internalRepresentative");
  internalRepresentativeElement.setWfItemId(wfItem.getId());
  var person = gaService.getPerson(gaUserDetail.getPersonId())
  internalRepresentativeElement.getPersonMap().put("internalRepresentative",
  person);
  wfItem.getBooleanMap().put("isCreatorOwner", null);
  wfService.saveOrUpdate(internalRepresentativeElement);
  wfItem.getWfItemElementSet().add(internalRepresentativeElement);
} true;

wfStartLogicLegacy
Questa logica viene eseguita in fase di creazione di un nuovo oggetto e setta l'attributo legacy a false (Legacy - disabilitare validazioni ed invio ad UGOV)
Settandolo di default a false NON verrà disattivata alcuna validazione/invio. Dal punto di vista del modello dati si tratta dell'attributo booleanMap[legacy]. Per maggiori dettagli cfr. modello dati dell'entità in questione.

```java
wfItem.getBooleanMap().put("legacy", false);
true;
```

**wfStartLogicVisibleOnPortal**
Questa logica viene eseguita in fase di creazione di un nuovo oggetto. Viene effettuata la valorizzazione dell'attributo "Visibile su portale pubblico"(booleanMap[visibleOnPortal]) utilizzando il valore definita nella variabile di configurazione ap.<wfItemType>.visibleOnPortal.default.
Per esempio per utilizzarla nei Gruppi di Ricerca (workgroup) la configurazione dovrà essere "ap.wkg.visibleOnPortal.default". Questa configurazione è trasversale su tutti i flussi di quella entità.
I possibili casi sono:

- Se la config è popolata è ha valore true (indipendentemente dal case) allora viene presettato il valore booleano a true
- Se la config è popolata è ha valore false (indipendentemente dal case) allora viene presettato il valore booleano a false
- Se la config non è presente o non è popolata o il valore è diverso da true/false allora non viene fatto nessun preset

Nel caso in cui si voglia attivare questa logica successivamente alla prima attivazione del flusso, è necessario procedere ad integrare il pregresso con l'attributo visibleOnPortal
Si riporta una query di esempio che integra gli item di tipo PRJ con il metadato visibleOnPortal settato a true per i soli item che non hanno il metadato: la query è quindi safe e può essere rilanciata.

```sql
insert into ap_item_data (fk_item, discriminator, boolean_value)
select ai.id, 'visibleOnPortal', 1
from ap_item ai
join ap_item_type ait on ait.id=ai.fk_item_type
left join ap_item_data aid on aid.fk_item=ai.id and aid.discriminator='visibleOnPortal'
where ait.identifier like 'PRJ.%'
and aid.fk_item is null
```

Dopo l’avvenuta modifica su DB è necessario procedere ad una resync su SOLR con la relativa procedura (DS6 - Core Entities SOLR)

**Configurazioni della StartLogic:**

- ap.TIPOLOGIA.visibleOnPortal.default

```java
var sourceItemType = wfService.getWfItemType(wfItem.getWfItemTypeId()).getRootWfItemType();

var visibleOnPortalConfiguration = Packages.it.cilea.wf.util.WfUtil.getParametricConfiguration("ap.<wfItemType>.visibleOnPortal.default", sourceItemType, null);
if(visibleOnPortalConfiguration != null){
    visibleOnPortalConfiguration = visibleOnPortalConfiguration.trim();
    if(!visibleOnPortalConfiguration.isEmpty()){  
        if (Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase(visibleOnPortalConfiguration, "true")){
            wfItem.getBooleanMap().put("visibleOnPortal", true);
        }else if (Packages.org.apache.commons.lang.StringUtils.equalsIgnoreCase(visibleOnPortalConfiguration, "false")){
            wfItem.getBooleanMap().put("visibleOnPortal", false);
        }
    }
}
```

//if else, do nothing
dateCoverageAndOverlapValidator
Questa validazione controlla la coerenza delle date inserite per i possibili elementi: owner, contributor, internalRepresentative, internalOrganizationUnit, historicalDescription, publiclyControlled.
Vengono valutati gli attributi element.dateMap[startDate], element.dateMapendDate, dateMap[startDate], dateMap[endDate].
Vengono valutate le seguenti configurazioni (che di default NON sono presenti), se non è presente ALCUNA configurazione la logica NON effettuerà alcun controllo.

- **Overlap**: ap.<macrotype>.<elementDiscriminator>.overlap
  - Se settata a true viene verificato che per gli elementi con quel discriminator non ci sia ALCUNA sovrapposizione di date.
  - Se settata con qualsiasi altro valore o non presente non viene fatto alcun controllo.
  - E` possibile specificare i ruoli degli elementi a cui effettuare i controlli sopra indicati, per farlo bisogna introdurre la configurazione ap.<macrotype>.<elementDiscriminator>.role.overlap.csv in cui bisogna specificare l'elenco dei code dei ruoli (quello che si vede da interfaccia dei dizionari alla colonna Codici).

- **Coverage**: ap.<macrotype>.<elementDiscriminator>.coverage
  - Se settata a true viene verificato che per gli elementi con quel discriminator esista un intervallo in cui sia stato un elemento con la data di inizio dell'oggetto.
  - Se settata a true viene verificato che per gli elementi con quel discriminator esista un intervallo in cui sia stato un elemento con la data di fine dell'oggetto.
  - E` possibile specificare i ruoli degli elementi a cui effettuare i controlli sopra indicati, per farlo bisogna introdurre la configurazione ap.<macrotype>.<elementDiscriminator>.role.coverage.csv in cui bisogna specificare l'elenco dei code dei ruoli (quello che si vede da interfaccia dei dizionari alla colonna Codici).

E` possibile specificare che tutti gli elementi abbiano la data di fine valorizzata, settando a true la configurazione ap.<macrotype>.<elementDiscriminator>.endDate.required. Questa viene valutata SOLO se la configurazione ap.<macrotype>.<elementDiscriminator>.coverage è settata a true.

Alcuni esempi:

- configurazione ap.prj.owner.overlap = true e configurazione ap.prj.owner.role.overlap.csv = ownerRole.administrative, si sta indicando che per i Responsabili scientifici marcati con il ruolo Amministrativo NON ci devono essere sovrapposizioni di intervalli
- configurazione ap.prj.owner.coverage = true e configurazione ap.prj.owner.role.coverage.csv NON presente, si sta indicando che ci deve essere una copertura completa per tutto il periodo di attività dell'oggetto da parte dei Responsabili scientifici e che vengono valutati TUTTI
- configurazione ap.prj.owner.overlap = true e configurazione ap.prj.owner.coverage = true, si sta indicando che ci deve essere una copertura completa per tutto il periodo di attività dell'oggetto, che NON ci siano sovrapposizioni di intervalli da parte dei Responsabili scientifici e che vengono valutati TUTTI

Per maggiori dettagli cfr. modello dati dell'entità in questione

**Configurazioni della ValidateLogic:**

- ap.TIPOLOGIA.xxxx.overlap
- ap.TIPOLOGIA.xxxx.coverage
- ap.TIPOLOGIA.xxxx.role.coverage.csv
- ap.TIPOLOGIA.xxxx.role.overlap.csv

```javascript
var wfItemType = wfService.getWfItemType(wfItem.getWfItemType());
var allDiscriminatorElement = ["owner", "contributor", "internalRepresentative", "internalOrganizationUnit", "historicalDescription", "publiclyControlled"];
for(var i=0; i<allDiscriminatorElement.length; i++) {
    var discriminatorElement = allDiscriminatorElement[i];
    var overlapConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".overlap", null);
    var overlapRoleCsvConfig = null;
    if(overlapConfig != null)
        overlapRoleCsvConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".role.overlap.csv", wfItemType, null);
    var coverageConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".coverage", wfItemType, null);
    var coverageRoleCsvConfig = null;
    if(coverageConfig != null)
        coverageRoleCsvConfig = WfUtil.getParametricConfiguration("ap.
wfIdentityLogicContextVisionContractPermissionsJs

Questa è una permissions logic che costruisce dinamicamente i permessi per i contratti per i team dipartimentali.

Se la variabile di configurazione `ap.contract-centralized-flow.department.enable` è settata a **true** o non è valorizzata allora vengono assegnati ai team dipartimentale i seguenti permessi:

- draft: crwfd
- validated: rwf
- signed: rwf
- archived: rwfd
- closed: rwf

Se la variabile di configurazione `ap.contract-centralized-flow.department.enable` è settata a **false** allora vengono assegnati ai team dipartimentale i seguenti permessi:

- draft:
- validated:
- signed: r
- archived:
- closed: r

Se la variabile di configurazione `ap.contract-centralized-flow.department.enable` è settata a **true** o non è valorizzata allora viene consentita la creazione di nuovi progetti ai team dipartimentali.

Se la variabile di configurazione `ap.contract-centralized-flow.department.enable` è settata a **false** allora **NON** viene consentita la creazione di nuovi progetti ai team dipartimentali.

In caso di modifica di queste variabili di configurazione procedere con un reload delle Configurazioni ed effettuare un nuovo upload della tassonomia per forzare anche il reload delle risorse che pilotano la visibilità dei bottoni di creazione dell'oggetto.

**Configurazioni della ValidateLogic:**

- `ap.contract-centralized-flow.department.enable`

```javascript
var wfTask=object;
var wfItem=wfService.getWfItem(wfTask.getWfItemId());
var permissions=wfTask.getPermissions();
var departmentContractCentralizedEnableParam = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.contract-centralized-flow.department.enable");
var departmentContractCentralizedEnable=("true" ==departmentContractCentralizedEnableParam||!departmentContractCentralizedEnableParam)?true:false;

switch("" + String(wfItem.getWfState().getDescription())){
    case "draft":
        if(departmentContractCentralizedEnable)
            permissions="crwfd"
        else
            permissions="";
        break;
    case "validated":
        if(departmentContractCentralizedEnable)
            permissions="rwf"
        else
            permissions="";
        break;
    case "signed":
        if(departmentContractCentralizedEnable)
            permissions="rwf"
        else
            permissions="r";
        break;
    case "archived":
```
if (overridesCentralizedEnable) permissions="rwfd"
else permissions="";
break;

case "closed":
if (overridesCentralizedEnable) permissions="rwf"
else permissions="r";
break;
}

wfTask.setPermissions(permissions);

**multipleOwnerValidator**
Questa validazione verifica che esista almeno un responsabile (owner)
Per maggiori dettagli cfr. il modello modello dati dell’entità in questione.

if (object.getId()==null){
    if (object.getPersonMap().get("owner")==null)
        errors.rejectValue("personMap[owner]","error.owner.required");
} else {
    if (object.getPersonMap().get("owner")==null)
        var ownerSet = FragmentUtil.GetCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","owner", wfService);
        if(ownerSet.isEmpty()){
            errors.rejectAndLocate("error.owner.required", "owner");
        }
}

**administrativeOwnerValidator**
Questa validazione controlla che esista almeno un referente amministrativo nel tab "Soggetti interni". Dal punto di vista del modello dati si tratta degli elementi di tipo administrativeOwner
Per maggiori dettagli cfr. excel modello dati dell’entità in questione.

var administrativeOwner = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "administrativeOwner", wfService);
if (administrativeOwner.size()<1)
    errors.rejectAndLocate("error.administrativeOwner.required", "administrativeOwner");

**multipleContributorValidator**
Questa validazione verifica che esista almeno un Docente interno (contributor)
Per maggiori dettagli cfr. il modello modello dati dell’entità in questione.

if (object.getId()==null){
    if (object.getPersonMap().get("contributor")==null)
        errors.rejectValue("personMap[contributor]","error.contributor.required");
} else {
if (object.getPersonMap().get("contributor") == null)
{
    var contributorWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","contributor", wfService);
    if(contributorWfElementSet.isEmpty())
    {
        errors.rejectAndLocate("error.contributor.required","contributor");
    }
}

startDateAndEndDateValidator
Questa validazione controlla che la data di inizio preceda la data di fine
Dal punto di vista del modello dati si tratta degli attributi startDate e endDate
Per maggiori dettagli cfr. modello dati dell'entità in questione

if(object.getDateMap().get("endDate") != null){
    if(object.getDateMap().get("startDate") != null){
        if(object.getDateMap().get("startDate").getTime().after(object.getDateMap().get("endDate").getTime())){
            errors.rejectValue("dateMap[startDate]","error.date.startDateEndDateNotCompatible");
            errors.rejectValue("dateMap[endDate]","error.date.startDateEndDateNotCompatible");
        } else { errors.rejectValue("dateMap[startDate]","error.date.startDateEndDateNotCompatible");
    }
} else { errors.rejectValue("dateMap[startDate]","error.date.startDateEndDateNotCompatible");
}

wfIdentityLogicOwnerSimplePermissionsJs
Questa è una permissions logic che costruisce dinamicamente i permessi per gli item per gli owner.
Se la variabile di configurazione ap.xxxx-flow.owner.create è settata a true o non è valorizzata allora vengono assegnati all'owner i seguenti permessi:

- crwd

Se la variabile di configurazione ap.xxxx-flow.owner.create è settata a false allora vengono assegnati all'owner i seguenti permessi:

- rwfd

Se la variabile di configurazione ap.xxxx-flow.owner.create dove xxxx è il nome del flusso è settata a true o non è valorizzata allora viene consentita la creazione di nuovi item agli owner.
Se la variabile di configurazione ap.xxxx-flow.owner.create dove xxxx è il nome del flusso è settata a false allora NON viene consentita la creazione di nuovi item agli owner.
In caso di modifica di queste variabili di configurazione procedere, nell'ordine, con

- reload delle configurazioni
- nuovo upload della tassonomia anche se non è stato modificato nulla (serve per forzare aggiornamento delle risorse su DB)
- reload delle autorizzazioni

Configurazioni della ValidateLogic:

- ap.xxxx-flow.owner.create

var wfTask=object;
var wfItem=wfService.getWfItem(wfTask.getWfItemId());
var permissions=wfTask.getPermissions();
var definition = wfItem.getWfItemType().getWfDefaultDefinition().getDescription();
var ownerCreateConf = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.*\+definition\+.owner.create");
var ownerCreateConfBoolean="true"==ownerCreateConf||!ownerCreateConf?true:false;

switch("" + String(wfItem.getWfState().getDescription())){
    case "draft":
        if(ownerCreateConfBoolean)
            permissions="crwfd"
        else
            permissions="rwf";
        break;
    case "reopened":
        if(ownerCreateConfBoolean)
            permissions="rwfd"
        else
            permissions="rwf";
        break;
}
wfTask.setPermissions(permissions);

checkElementDateBetweenItemDateValidator
Questa validazione controlla la coerenza delle date di inizio e fine in fase di salvataggio del fragment.
Verificare la coerenza degli intervalli tra le date dell'elemento corrente (wfItemElement) e quelle dell'oggetto (wfItem).
Come date dell'elemento corrente vengono recuperate le due date presenti negli attributi dateMap[startDate] e dateMap[endDate].
Come data di inizio dell'oggetto viene recuperata la data presente nell'attributo element.dateMap[discriminator].
Il discriminator viene recuperato dalla configurazione specifica ap.<sourceItemType>.<discriminatorElement>.validator.date.start.discriminator, se non presente viene valutata la configurazione più generale ap.<sourceItemType>.contributorAndOwner.validator.date.start.discriminator.
Come data di fine dell'oggetto viene recuperata la data presente nell'attributo element.dateMap[discriminator].
Il discriminator viene recuperato dalla configurazione specifica ap.<sourceItemType>.<discriminatorElement>.validator.date.end.discriminator, se non presente viene valutata la configurazione più generale ap.<sourceItemType>.contributorAndOwner.validator.date.end.discriminator.
Vengono effettuati i seguenti controlli se presenti le relative configurazione:

- Se la data di inizio dell'oggetto è presente e se la data di inizio dell'elemento corrente è presente
  Viene verificato che la data di inizio dell'elemento corrente sia successiva alla data di inizio dell'oggetto.
- Se la data di fine dell'oggetto è presente e se la data di fine dell'elemento corrente è presente
  Viene verificato che la data di fine dell'elemento corrente sia precedente alla data di fine dell'oggetto.

Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.xxxx.validator.date.start.discriminator
- ap.TIPOLOGIA.contributorAndOwner.validator.date.start.discriminator

var wfItemElementStartDate = object.getDateMap().get("startDate");
var wfItemElementEndDate = object.getDateMap().get("endDate");
var sourceItemType = wfService.getWfItemType(object.getWfItem().getWfItemTypeId());
var discriminatorElement = StringUtils.substringBefore(object.getDiscriminator(), "");
discriminatorElement = new String(discriminatorElement).toString();
var startDateDiscriminator = WfUtil.getParametricConfiguration("ap.<wfItemType>." +
discriminatorElement + ".validator.date.start.discriminator", sourceItemType, null);
if(startDateDiscriminator == null){
    startDateDiscriminator = WfUtil.getParametricConfiguration("ap.<wfItemType>." +
contributorAndOwner.validator.date.start.discriminator", sourceItemType, null);
}

var endDateDiscriminator = WfUtil.getParametricConfiguration("ap.<wfItemType>." +
discriminatorElement + ".validator.date.end.discriminator", sourceItemType, null);
if(endDateDiscriminator == null){
    endDateDiscriminator = WfUtil.getParametricConfiguration("ap.<wfItemType>." +
contributorAndOwner.validator.date.end.discriminator", sourceItemType, null);
}

var wfItemStartDate = object.getWfItem().getDateMap().get(startDateDiscriminator);
var wfItemEndDate = object.getWfItem().getDateMap().get(endDateDiscriminator);
var formatter = Packages.org.apache.commons.lang.time.FastDateFormat.getInstance("dd/MM/yyyy");
if( wfItemElementStartDate!= null && wfItemStartDate && (wfItemElementStartDate.before (wfItemStartDate) || (wfItemEndDate!=null && wfItemElementStartDate.after(wfItemEndDate))) ){
  var suffixMessage=" - ";
  if (wfItemEndDate!=null){
    suffixMessage+=formatter.format(wfItemEndDate.getTime());
  } else {
    suffixMessage+="QUALSIASI DATA MAGGIORE DELLA DATA DI INIZIO";
  }
  errors.rejectValue("dateMap[startDate]", "error.date.
  startDateOfElementAfterStartDateOfItemValidator", [formatter.format(wfItemStartDate.getTime()) +
  suffixMessage], null);
}

if( wfItemElementEndDate != null && wfItemStartDate && (wfItemElementEndDate.before (wfItemStartDate) || (wfItemEndDate != null && wfItemElementEndDate.after(wfItemEndDate))) ) {
  var suffixMessage=" - ";
  if (wfItemEndDate != null) {
    suffixMessage += formatter.format(wfItemEndDate.getTime());
  } else {
    suffixMessage += "QUALSIASI DATA MAGGIORE DELLA DATA DI INIZIO";
  }
  errors.rejectValue("dateMap[endDate]", "error.date.
  checkElementDateBetweenItemDateValidator.endDate", [formatter.format(wfItemStartDate.getTime()) +
  suffixMessage], null);
}

internalOrganizationUnitValidator
Questa validazione controlla che esista almeno una unità organizzativa responsabile (internalOrganizationUnit)
Per maggiori dettagli cfr. modello dati dell'entità in questione.

    var internalOrganizationUnitWfElementSet=FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement","internalOrganizationUnit","", wfService);
    if(internalOrganizationUnitWfElementSet.isEmpty()){
      errors.rejectAndLocate("error.internalOrganizationUnit.required","internalOrganizationUnit");
    }

contributorValidator
Validator per controllare presenza di almeno un partecipante (contributor)
Dal punto di vista del modello dati si tratta degli attributi contributor
Per maggiori dettagli cfr. modello dati dell'entità in questione

    var contributorWfElementSet=FragmentUtil.get_currentFragmentSetByParentAndDiscriminator
    (object, "it.cilea.wf.model.WfItemElement","contributor", wfService);
    if(contributorWfElementSet.isEmpty()){
      errors.rejectAndLocate("error.contributor.required", "contributor");
    }

keywordErcValidator
Validator per controllare presenza di almeno una classificazione ERC
Dal punto di vista del modello dati si tratta degli attributi keywordErc
Per maggiori dettagli cfr. modello dati dell'entità in questione

    var keywordErcWfElementSet=FragmentUtil.get_currentFragmentSetByParentAndDiscriminator
    (object, "it.cilea.wf.model.WfItemElement","keywordErc", wfService);
    if(keywordErcWfElementSet.isEmpty()){
academicField2000Validator
Validator per controllare presenza di almeno un SSD (academicField2000)
Dal punto di vista del modello dati si tratta degli attributi academicField2000
Per maggiori dettagli cfr. modello dati dell’entità in questione

keywordSdgValidator
Validator per controllare presenza di almeno un SDG (sustainableDevelopmentGoal)
Dal punto di vista del modello dati si tratta degli attributi sustainableDevelopmentGoal
Per maggiori dettagli cfr. modello dati dell’entità in questione

esfriClassificationValidator
Validator per controllare presenza di almeno una classificazione ESFRI
Dal punto di vista del modello dati si tratta degli attributi esfriClassification
Per maggiori dettagli cfr. modello dati dell’entità in questione

keywordAtecoValidator
Validator per controllare presenza di almeno una classificazione ATECO
Dal punto di vista del modello dati si tratta degli attributi keywordAteco
Per maggiori dettagli cfr. modello dati dell’entità in questione
**keywordUserDefinedValidator**
Questa validazione controlla la presenza di almeno una parola chiave libera.
Dal punto di vista del modello dati si tratta degli attributi **keywordUserDefined**
Per maggiori dettagli cfr. modello dati dell'entità in questione

```javascript
var keywordUserDefinedWfElementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "keywordUserDefined", wfService);
if(keywordUserDefinedWfElementSet.isEmpty()){
    errors.rejectAndLocate("error.keywordUserDefined.required", "keywordUserDefined");
}
```

**researchLineValidator**
Questa validazione presenza di almeno una linea di ricerca
Dal punto di vista del modello dati si tratta degli attributi **researchLine**
Per maggiori dettagli cfr. modello dati dell'entità in questione

```javascript
var researchLineWfElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "it.cilea.wf.model.WfItemElement", "researchLine", wfService);
if(researchLineWfElementSet.isEmpty()){
    errors.rejectAndLocate("error.researchLine.required", "researchLine");
}
```

**checkDateExtensionValidator**
Questa validazione controlla la coerenza delle date inserite.
Precisamente, viene controllato che:
- startDate minore o uguale a endDate relativi all'oggetto radice
- durationInMonth maggiore o uguale a 0 relativo all'oggetto radice
- gli elementi di tipo approval abbiano una data maggiore della startDate dell'oggetto radice
- gli elementi di tipo extension abbiano una data maggiore della startDate dell'oggetto radice

Per maggiori dettagli cfr. modello dati dell'entità in questione.
Questa validazione viene attivata SOLO se **ap.checkDateExtensionValidator.enabled=true**
Inoltre è disponibile la configurazione **ap.(PRJ-CON).approval.approvalDateAfterStartDate.enabled** (una per i progetti e una per i contratti), queste due configurazioni permettono l'inserimento di una data di approvazione successiva alla data effettiva di inizio del progetto/contratto.

In particolare per la configurazione **ap.(PRJ-CON).approval.approvalDateAfterStartDate.enabled**:
- se viene impostata con valore **true**, la logica di validazione è ATTIVATA e quindi verrà scatenato l'errore sulla coerenza delle date
- se viene impostata con valore **false**, la logica di validazione è DISATTIVATA e non verrà effettuato il controllo

Quest'ultima configurazione disabilita il controllo del validator relativo alla data di approvazione se è maggiore della data di inizio, può essere utile come nel caso in cui per qualche motivo interno il progetto è partito perché ufficiosamente è stato approvato ma l'approvazione ufficiale è arrivata dopo (ad esempio con decreto rettorale).
Nel caso di inserimento di data di approvazione successiva all'inizio del progetto/contratto, verrà visualizzato un warning NON bloccante in ogni caso

**Configurazione della ValidateLogic:**
- **ap.checkDateExtensionValidator.enabled**
- **ap.prj.approval.approvalDateAfterStartDate.enabled**
- **ap.con.approval.approvalDateAfterStartDate.enabled**

```javascript
var enable = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.checkDateExtensionValidator.enabled");
var rootIdentifierCode = wfService.getType(object, "rootWfItemType");
rootIdentifierCode = rootIdentifierCode.getIdentifier();
var approvalDateAfterStartDateEnabled = "\";
if(rootIdentifierCode.equalsIgnoreCase("prj")){
    approvalDateAfterStartDateEnabled = Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.prj.approval.approvalDateAfterStartDate.enabled");
}
```
else {
    approvalDateAfterStartDateEnabled = Packages.it.cilea.core.configuration.util.
    ConfigurationUtil.getConfigValue("ap.con.approval.approvalDateAfterStartDate.enabled");
}

if(approvalDateAfterStartDateEnabled == null){
    approvalDateAfterStartDateEnabled = "true";
}

if(enable.equalsIgnoreCase("true")){
    var startDate = object.getDateMap().get("startDate");
    var endDate = object.getDateMap().get("endDate");
    if(startDate == null){
        errors.rejectValue("dateMap[startDate]", "error.notNull.fieldLabelKey",
        [messageUtil.findMessage("label.project.startDate")], null);
    }
    if(endDate == null){
        errors.rejectValue("dateMap[endDate]", "error.notNull.fieldLabelKey",
        [messageUtil.findMessage("label.project.endDate")], null);
    }
    if(endDate != null && startDate != null){
        if(startDate.getTime().after(endDate.getTime())){
            errors.rejectValue("dateMap[startDate]", "error.notNull.
            fieldLabelKey", [messageUtil.findMessage("label.project.startDate")], null);
            errors.rejectValue("dateMap[endDate]", "error.notNull.
            fieldLabelKey", [messageUtil.findMessage("label.project.endDate")],
            null);
        }
    }
}

var durationInMonth = object.getIntegerMap().get("durationInMonth");
if(durationInMonth == null || durationInMonth < 0){
    errors.rejectValue("integerMap[durationInMonth]", "error.project.
    durationInMonth");
}
if(endDate != null && startDate != null){
    var extensionSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
    "extension", wfService);
    var extensionSetIterator = extensionSet.iterator();
    while(extensionSetIterator.hasNext()){
        var extensionElement = extensionSetIterator.next();
        if (endDate.compareTo(extensionElement.getDateMap().get
        ("currentEndDate"))<0 ||
        durationInMonth<extensionElement.getIntegerMap().get
        ("currentDurationInMonth")
        errors.rejectAndLocate("error.project.extension.date",
        "extension");
    }
    if(extensionElement.getDateMap().get("previousEndDate").
    getTime().after(extensionElement.getDateMap().get("currentEndDate").
    getTime()){ errors.rejectAndLocate("error.date.
    startDateEndDateNotCompatible", extension);
    }
}
    approvalSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
    getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement",
    "approval", wfService);
    var approvalSetIterator = approvalSet.iterator();
    while(approvalSetIterator.hasNext()){
        var approvalElement = approvalSetIterator.next();
        var approvalDate = approvalElement.getDateMap().get
        ("approvalDate");
if (approvalDate != null){
    if(!approvalDateAfterStartDateEnabled.equalsIgnoreCase("false")){
        if(approvalDate.getTime().after(startDate.getTime())){
            errors.rejectAndLocate("error.project.approval.date", "approval");
        }
    }
}
}

customizedInternalOrganizationValidator
Questa validazione viene triggerata SOLO in fase di creazione dell'oggetto, e se l'utente ha selezionato si nel campo Struttura capofila diversa da afferenza del responsabile?.
In tal caso verifica che sia valorizzato il campo della Struttura.
Dal punto di vista del modello dati si tratta degli elementi di tipo organizationUnitMap[internalOrganizationUnit].

if (object.getId()==null){
    if(Packages.java.lang.Boolean.TRUE.equals(object.getBooleanMap().get("customizedInternalOrganization"))){
        if(object.getOrganizationUnitMap().get("internalOrganizationUnit") == null){
            errors.rejectValue("organizationUnitMap[internalOrganizationUnit]", "error.notNull", [messageUtil.findMessage("label.required")], null);
        }
    }
}

ownerPositionStartValidator
Questa validazione viene triggerata SOLO se la configurazione ap.<tipologia>.owner.internalOrganizationUnit.mandatory è settata a true.
Se questa configurazione non viene trovata la validazione NON viene eseguita.
Verifica che l'owner (responsabile scientifico) inserito in fase di creazione abbia una afferenza dipartimentale.
La data alla quale estrarre l'afferenza dipartimentale del responsabile è quella contenuta nel metadato con il nome recuperabile dalla configurazione ap.<tipologia>.owner.position.date.
Se questa configurazione

- NON è presente, viene usato la data inserita nel metadato startDate
- è presente ed ha valore uguale a CURRENT viene usata la data corrente
- è presente ed ha valore uguale a CURRENT_OR_LAST viene usata l'ultima afferenza dipartimentale disponibile (attiva o cessata)
- è presente ed NON ha un valore tra quelli elencati allora viene usata la data presente nel metadato con il nome specificato

E' inoltre possibile stabilire quali siano i contesti all'interno dei quali cercare l'afferenza dipartimentale per il responsabile scientifico.
Questo viene pilotato dalla configurazione ap.<tipologia>.owner.position.context.
Se questa configurazione

- NON è presente, viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a research allora viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a support allora viene usato il solo contesto di supporto (TA)
- è uguale a ALL allora vengono usati entrambi i contesti

Con <tipologia> viene inteso il codice a tre lettere in minuscolo che indica la tipologia radice dell'oggetto in questione come ad esempio : lab, eqp, inm, prj, ...
Per un elenco esaustivo fare riferimento all'"Alberatura tipologie RM/AP"

Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.owner.internalOrganizationUnit.mandatory
- ap.TIPOLOGIA.owner.position.date
- ap.TIPOLOGIA.owner.position.context

if (!errors.hasErrors() && object.getId()==null){
var isInternalOrganizationMandatory=WfUtil.getMandatory(object,"owner");
if (isInternalOrganizationMandatory){
    var checkDate = WfUtil.getCheckDate(object, "owner", wfService);
    var positionContext = WfUtil.getPositionContext(object,
    positionContext, "department", gaService);
    var owner = object.getPersonMap().get("owner");
    var ownerPositionSet = WfUtil.getPositionSet(owner, checkDate,
    [positionContextDescription,checkDateDescription], null);
    if(ownerPositionSet.isEmpty()){errors.rejectValue("personMap[owner]", "error.owner.position", [positionContextDescription,checkDateDescription], null);}
}

getYearFromStartDateValidator
Logica lanciata in fase di salvataggio che ricava l'anno dalla data di inizio e lo salva nel metadato year.

var startDate=object.getDateMap().get("startDate");
//log.error("startDate: "+startDate.getTime());
if(startDate!=null){object.year=startDate.get(startDate.YEAR);
}else{
    errors.reject("error.startDate.required");
}

addOrgUnitFromNewOwnerValidator
Questa logica, ogni volta che viene aggiunto o aggiornato un owner (responsabile scientifico) o contributor(partecipante), ne ricava il dipartimento e lo aggiunge alle strutture di afferenza, se non già presente.
La data alla quale estrarre l'afferenza dipartimentale del responsabile è quella contenuta nel metadato con il nome recuperabile dalla configurazione ap.<tipologia>.owner.position.date.
Se questa configurazione

- NON è presente, viene usata la data inserita nel metadato startDate
- è presente ed ha valore uguale a CURRENT viene usata la data corrente
- è presente ed ha valore uguale a CURRENT_OR_LAST viene usata l'ultima afferenza dipartimentale disponibile (attiva o cessata)
- è presente ed NON ha un valore tra quelli elencati allora viene usata la data presente nel metadato con il nome specificato

E' inoltre possibile stabilire quali siano i contesti all'interno dei quali cercare l'afferenza dipartimentale per il responsabile scientifico.
Questo viene pilotato dalla configurazione ap.<tipologia>.owner.position.context.
Se questa configurazione

- NON è presente, viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a research allora viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
- è uguale a support allora viene usato il solo contesto di supporto (TA)
- è uguale a ALL allora vengono usati entrambi i contesti

E' possibile infine definire le mappature tra ruolo dell'owner|contributor e il ruolo del dipartimento che deve essere aggiunto.
Questo viene pilotato dalla variabile di configurazione ap.<tipologia>.[owner|contributor].role.<codice_dizionario>.inheritedInternalOrganizationUnit.role.
Il valore di questa configurazione indica il code del dizionario (visibile da interfaccia di gestione dei dizionari) associato al ruolo del dipartimento per quello specifico ruolo dell'[owner|contributor].
Tipicamente queste mappature vengono gestite a livello di sistema.

Accanto al meccanismo descritto di estrazione del dipartimento dalle afferenze delle persone, è possibile avere anche un'altra configurazione (al momento
SOLO per le entità PRJ.
Per i PRJ, specificando la variabile `ap.prj.owner.department.enabled` è possibile richiedere all'utente, direttamente nell'elemento dell'owner, di specificare il dipartimento associato.
Questo dipartimento NON è necessario che sia quello di afferenza ma potrebbe essere una struttura qualsiasi con la quale la persona sta svolgendo il progetto.
Se questa variabile è presente, la struttura che verrà riportata nella sezione delle strutture interne sarà quella specificata direttamente nell'elemento owner.

E' possibile, infine, definire la variabile `ap.<tipologia>.internalOrganizationUnit.fullyInferredFromPeopleTypeCsv`
Questa variabile abilita il ricalcolo complesso delle unità organizzative a partire dalle strutture di afferenza delle persone.
Questa variabile deve contenere l'elenco, separato da virgola, degli elementi del modello associati alle persone (owner, contributor, ....) dalle estrarrre le info delle strutture.
E' comunque sempre possibile operare in edit sulle strutture inferiori.
Questo significa che

1. se viene settata a true la variabile `ap.<tipologia>.internalOrganizationUnit.fullyInferredFromPeopleTypeCsv`
2. se viene modificata/eliminata/aggiunta direttamente una struttura nella sezione delle strutture
3. se viene modificata/eliminata/aggiunta una persona

verrà forzato il ricalcolo delle strutture a partire dalle persone presenti e quindi verranno perse le modifiche manuali fatte direttamente sulle strutture.
Nel caso dei PRJ è presente un ulteriore meccanismo che, se la variabile `ap.<tipologia>.internalOrganizationUnit.fullyInferredFromPeopleTypeCsv` è definita, impedisce l'edit diretto delle strutture a tutti tranne agli utenti che hanno la visione completa e i diritti di edit sulle strutture.

Questa validazione considera anche la configurazione `ap.<tipologia>.internalOrganizationUnit.inferredChidCsv`
Questa configurazione consente di creare automaticamente un'altro elemento di modello associato alla struttura estratta dalla persona.
Per i dettagli della configurazione fare riferimento alla descrizione presente in `deleteOrgUnitAssociatedToOldOwnerValidator`
Qualora venga fatto un edit di un elemento struttura (direttamente a partire dall'elemento struttura o a partire dagli elementi delle persone), tale edit verrà consentito se sarà possibile eliminare gli eventuali elementi inferiti agganciati.

Con «tipologia» viene inteso il codice a tre lettere in minuscolo che indica la tipologia radice dell'oggetto in questione ad esempio : lab, eqp, inm, prj, ...
Per un elenco esaustivo fare riferimento all'"Alberatura tipologie RM/AP"

**Configurazioni della ValidateLogic:**

- `ap.TIPOLOGIA.owner.position.context`
- `ap.TIPOLOGIA.[owner|contributor].role.COD_DIZIONARIO.inheritedInternalOrganizationUnit.role`
- `ap.TIPOLOGIA.owner.department.enabled`
- `ap.TIPOLOGIA.internalOrganizationUnit.fullyInferredFromPeopleTypeCsv`
- `ap.TIPOLOGIA;.internalOrganizationUnit.inferredChidCsv`

```java
if (!errors.hasErrors()) {
    WfUtil.addOrUpdateOrgUnitInferredFromPersonElement(object, wfService, gaService, errors);
}
```

**deleteOrgUnitAssociatedToOldOwnerValidator**
Questa logica, ogni volta che viene ELIMINATO una persona ne ricava il dipartimento ed elimina anche questo dalle strutture interne (se non esiste un'altra persona con lo stesso dipartimento)
Qualora sia definita anche la variabile `ap.<tipologia>.internalOrganizationUnit.inferredChidCsv` viene effettuata anche la cancellazione degli elementi elencati come associati all'unità organizzativa.

Ad esempio nel caso di progetti esiste nel modello l'elemento `internalOrganizationUnitCost`.
Se la variabile ha questo valore, verrà eliminato anche l'elemento associato alla struttura che si sta eliminando.
Gli unici elementi di modello che possono essere inseriti in questa variabile sono quelli che prevedono nel modello dati l'attributo `ouId`.
Continuando con l'esempio `internalOrganizationUnitCost` nei progetti, è possibile definire anche la variabile `ap.<tipologia>.internalOrganizationUnit.inferredChid.<elemento>.jsValidator e cioè ap.prj.internalOrganizationUnit.inferredChid.internalOrganizationUnitCost.jsValidator`
Questa configurazione deve contenere un javascript che deve ritornare true o false in base al fatto che l'elemento possa o meno essere cancellato.
Questo javascript ha in input `wfItemElement` che è l'elemento per il quale verificare l'eliminabilità.
Toricamente l'oggetto può essere eliminato se non sono stati inserite informazioni da un operatore.
Se la configurazione non è presente allora l'elemento viene considerato eliminabile, indipendentemente dalla presenza di dati.

**Configurazioni della ValidateLogic:**

- `ap.TIPOLOGIA.internalOrganizationUnit.inferredChidCsv`
- `ap.TIPOLOGIA.internalOrganizationUnit.inferredChid.<elemento>.jsValidator`

```java
if (!errors.hasErrors()) {
    WfUtil.deleteOrgUnitAssociatedToThisPersonElement(object, wfService, gaService, errors);
}
```
**addElementInferredFromOrgUnitElementValidator**
Questa logica, ogni volta che viene inserito o modificato una struttura aggiunge degli elementi figli collegati. Questi elementi vengono definiti in nella configurazione `ap.<tipologia>.internalOrganizationUnit.inferredChidCsv` che riporta l’elenco separato da virgole degli elementi figli.

Ad esempio nel caso di progetti esiste nel modello l’elemento `internalOrganizationUniCost`, Se la variabile ha questo valore, verrà aggiunto un elemento `internalOrganizationUniCost` (se non già presente) per la struttura che si sta aggiungendo /aggiornando.
Gli unici elementi di modello che possono essere inseriti in questa variabile sono quelli che prevedono nel modello dati l’attributo `oudl`.

**Configurazioni della ValidateLogic:**
- `ap.TIPOLOGIA.internalOrganizationUnit.inferredChidCsv`

```java
if(!errors.hasErrors()){  
    WfUtil.addElementInferredFromOrgUnitElement(object, wfService, gaService, errors);
}
```

**deleteElementAssociatedToOrgUnitElementValidator**
Questa logica, ogni volta che viene ELIMINATA una struttura elimina gli eventuali elementi figli collegati. Questi elementi vengono definiti in nella configurazione `ap.<tipologia>.internalOrganizationUnit.inferredChidCsv` che riporta l’elenco separato da virgole degli elementi figli.

Ad esempio nel caso di progetti esiste nel modello l’elemento `internalOrganizationUniCost`, Se la variabile ha questo valore, verrà eliminato anche l’elemento associato alla struttura che si sta eliminando.
Gli unici elementi di modello che possono essere inseriti in questa variabile sono quelli che prevedono nel modello dati l’attributo `oudl`.
Continuando con l’esempio `internalOrganizationUniCost` nei progetti, è possibile definire anche la variabile `ap.<tipologia>.internalOrganizationUnit.inferredChid.<elemento>.jsValidator` e cioè `ap.prj.internalOrganizationUnit.inferredChid.internalOrganizationUniCost.jsValidator`.
Questa configurazione deve contenere un javascript che deve ritornare true o false in base al fatto che l’elemento possa o meno essere cancellato.
Questo javascript ha in input `wfItemElement` che è l’elemento per il quale verificare l’eliminabilità.
Tipicamente l’oggetto può essere eliminato se non sono stati inserite informazioni da un operatore.
Se la configurazione non è presente allora l’elemento viene considerato eliminabile, indipendentemente dalla presenza di dati.

**Configurazioni della ValidateLogic:**
- `ap.TIPOLOGIA.internalOrganizationUnit.inferredChidCsv`
- `ap.TIPOLOGIA.internalOrganizationUnit.inferredChid.ELEMENTO.jsValidator`

```java
if(!errors.hasErrors()){  
    WfUtil.deleteElementAssociatedToThisOrgUnitElement(object, wfService, gaService, errors);
}
```

**notEmptyResearchPositionValidatorTutor**
//validator che controlla se il tutor inserito abbia una posizione di ricerca presso un dipartimento

```java
function showOrHiddenError(error, mandatory){  
    if(mandatory){  
        errors.reject(error);  
    }else {  
        log.error(error);  
    }
}

//prima controllo che non ci siano stati errori  
//se non ci sono stati allora vado avanti  
if(!errors.hasErrors()){  
    inutile, ma mi serve per ricordarmi il contesto  
    var wfItemElement=object;
    var fragmentInfo=Packages.it.cilea.core.fragment.util.FragmentUtil.
```
ownerWithSameDepartmentAsHeadValidator
Questa validazione viene triggerata in fase di creazione di un nuovo oggetto solo se l'utente è in visione dipartimentale. Verifica che l'utente che sta creando un nuovo oggetto a livello dipartimentale, specifici come responsabile scientifico una persona che afferisce al proprio dipartimento.
Se non viene specificato nessun responsabile scientifico, questa validazione verrà ignorata.
Se presente il campo booleanMap[customizedInternalOrganization] ed è stato valorizzato a true, questa validazione verrà ignorata.
Se presente il campo "tipologia"-owner.position.date.
Se questa configurazione
• NON è presente, viene usata la data inserita nel metadato startDate
• è presente ed ha valore uguale a CURRENT viene usata la data corrente
• è presente ed ha valore uguale a CURRENT_OR_LAST viene usata l'ultima afferenza dipartimentale disponibile (attiva o cessata)
• è presente ed NON ha un valore tra quelli elencati allora viene usata la data presente nel metadato con il nome specificato
E' inoltre possibile stabilire quali siano i contesti all'interno dei quali cercare l'afferenza dipartimentale per il responsabile scientifico. Questo viene pilotato dalla configurazione ap."tipologia"-owner.position.context.
Se questa configurazione
• NON è presente, viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
• è uguale a research allora viene usato il solo contesto di ricerca (docenti, ricercatori, dottorandi, ...)
• è uguale a support allora viene usato il solo contesto di supporto (TA)
• è uguale a ALL allora vengono usati entrambi i contesti
Con "tipologia" viene inteso il codice a tre lettere in minuscolo che indica la tipologia radice dell'oggetto in questione come ad esempio : lab, eqp, inm, prj, ...
Per un elenco esaustivo fare riferimento all’"Alberatura tipologie RM/AP".
```java
var owner = object.getPersonMap().get("owner");
if(owner != null){
    var checkDate=WfUtil.getCheckDate(object, "owner", wfService);
    var positionContext=WfUtil.getPositionContext(object, wfService);
    var ownerPositionSet=WfUtil.getPositionSet(owner, checkDate, positionContext, "department", gaService);
    var department=GaUtil.getPriorityOrganizationUnit(ownerPositionSet, positionContext, "department");
    var isCurrentUserAuthorizedOnDepartment=WfUtil.isCurrentUserAuthorizedOnDepartment(object, department, gaAuthorizationService, gaService, wfService);
    if(!isCurrentUserAuthorizedOnDepartment){
        var positionContextDescription="";
        var checkDateDescription="";
        if ("research"==positionContext){
            positionContextDescription="di ricerca";
        } else if ("support"==positionContext){
            positionContextDescription="di supporto";
        } else {
            checkDateDescription="corrente";
        }
    } else {
        if (checkDate==null){
            checkDateDescription="ultima disponibile";
        } else {
            checkDateDescription="alla data " + FastDateFormat.getInstance("dd/MM/yyyy").format(checkDate);
        }
    }
    errors.reject("error.owner.position.department.mustBeSameAsHead", [positionContextDescription, checkDateDescription], null);
}
```

**departmentChangeRoleValidator**

Questa validazione opera sull'elemento `internalOrganizationUnit` figlio dell'oggetto radice. Se l'utente è in visione dipartimentale per il dipartimento X e tenta di modificare il ruolo del dipartimento X nell'oggetto radice, viene sollevato un warning se il nuovo ruolo prevede diritti inferiori. Se l'utente è in visione dipartimentale per il dipartimento X e tenta di aggiungere un nuovo dipartimento, viene sollevato un warning se il nuovo ruolo del nuovo dipartimento prevede diritti superiori rispetto al suo dipartimento X.

Ad esempio se un utente accede in visione dipartimentale ad un progetto e cambia il ruolo del dipartimento da "Principale" ad "Aggregato", l'operazione viene segnalata perché il ruolo "Aggregato" prevede diritti inferiori.

Questa operazione è consentita in visione completa per i team abilitati. Di default è possibile modificare il ruolo del proprio dipartimento con presentazione di un messaggio di warning e di conferma da parte dell'utente se la modifica dovessere causare la perdita di diritti di scrittura. E' possibile disabilitare la modifica in caso di riduzione di permessi settando a false la variabile di configurazione generale `ap.HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled` o la variabile di configurazione specifica `ap.HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.[enter|save|element:element_discriminator:save|delete].departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled`.

Se la variabile di configurazione non è presente (default) o è settata a true è possibile effettuare la modifica. Per maggiori dettagli cfr. modello dati dell'entità in questione.

**Configurazioni della ValidateLogic:**

- `ap.HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled`
- `ap.HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.[enter|save|element:element_discriminator:save|delete].departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled`
function check(wfItemElement, ouIdToCheck, ouRoleToCheck)
{
    if(!WfUtil.organizationUnitKeepsTheSamePermissions(applicationContext,
        wfService, object, request.getParameter("wfTaskId"))
    {
        var wfItem = object.getWfItem();
        var stateDescription = wfItem.getWfState().getDescription();
        if(WfUtil.isApplicableValidationFromConfiguration(wfItem, "element:
            internalOrganizationUnit:save", wfItem.getWfState().getDescription(), "department",
            "departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied", "enabled")
        {
            if(object.getBooleanMap().get("roleSwitchAcknowledged") !=
                null) {
                    if(object.getBooleanMap().put("roleSwitchAcknowledged", null));
                } else {
                    object.getBooleanMap().put("roleSwitchAcknowledged", false);
                }
            if(ouIdToCheck != null && ouRoleToCheck != null)
            {
                object.getOrganizationUnitMap().put("ouId",
                    gaService.getOrganizationUnit(ouIdToCheck));
                object.getWfDictionaryMap().put("roleId",
                    wfService.getWfDictionary(ouRoleToCheck));
                wfService.saveOrUpdate(object);
                errors.reject("error.operation.department.
                    changeRole");
            }
        } else {
            object.getBooleanMap().put("roleSwitchAcknowledged", false);
        }
    }
    else {
        if(ouIdToCheck != null && ouRoleToCheck != null)
        {
            object.getOrganizationUnitMap().put("ouId",
                gaService.getOrganizationUnit(ouIdToCheck));
            object.getWfDictionaryMap().put("roleId",
                wfService.getWfDictionary(ouRoleToCheck));
            wfService.saveOrUpdate(object);
            errors.reject("error.operation.department.
                changeRoleNotAllowed");
        }
    }
}

if (GaUtil.isCurrentUserInDepartmentView()){
    var ouIdToCheck = object.getIntegerMap().get("ouIdPreviousValue");
    var ouRoleToCheck = object.getIntegerMap().get("roleIdPreviousValue");
    if(ouIdToCheck != null && ouRoleToCheck != null){
        check(object, ouIdToCheck, ouRoleToCheck);
    } else {
        var internalOrganizationUnitSet = Packages.it.cilea.core.fragment.util.
            FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object.getWfItem(), "it.cilea.wf.model.
                WfItemElement", "internalOrganizationUnit", wfService);
        var internalOrganizationUnitSetIterator = internalOrganizationUnitSet.iterator();
        while(internalOrganizationUnitSetIterator.hasNext())
        {
**departmentDeleteValidator**
Questa validazione opera sull'elemento `internalOrganizationUnit` figlio dell'oggetto radice.
Se l'utente è in visione dipartimentale per il dipartimento X e tenta di eliminare il dipartimento X dall'oggetto radice, viene sollevato un errore in quanto cancellando questa associazione, l'utente non potrebbe più operare su quell'oggetto.
Questa operazione è consentita in visione completa per i team abilitati.
Per maggiori dettagli cfr. modello dati dell'entità in questione

```java
var wfItemElement = internalOrganizationUnitSetIterator.next();
check(object, null, null);
}
}
```

**notEditableFragmentValidator**
Questa validazione impedisce la modifica dei dati in un fragment. Può agire sia in caso di salvataggio che in caso di eliminazione.

```java
errors.reject("error.operation.notallowed");
```

**contributorAndOwnerValidatorWithStartDateEndDate**
Questa validazione opera sugli elementi `owner` (responsabile scientifico) e `contributor` (partecipante) figli dell'oggetto radice.
Viene verificato che l'intervallo di validità degli oggetti figli siano compresi all'interno della validità dell'oggetto padre.
Gli attributi per le date di inizio e fine sono `startDate` e `endDate` per gli elementi `owner` e `contributor`.
Sono di default gli stessi anche per l'oggetto padre, a meno che non venga inserita la configurazione gerarchica opportuna, ad es. "ap.prj.contributorAndOwner.validator.date.end.discriminator", che definisce il discriminator del metadato di tipo data dell'oggetto padre.
Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configurazioni della ValidateLogic:
- `ap.TIPOLOGIA.contributorAndOwner.validator.date.start.discriminator`
- `ap.TIPOLOGIA.contributorAndOwner.validator.date.end.discriminator`

```java
var elementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "contributor", wfService);
if (elementSet.size()>0){
    var elementSetIterator = elementSet.iterator();
    var sourceItemType = wfService.getWfItemType(object.getWfItemTypeId());
    var startDateDiscriminator = WfUtil.getParametricConfiguration("ap.<wfItemType>.contributorAndOwner.validator.date.start.discriminator", sourceItemType, null);
    var endDateDiscriminator = WfUtil.getParametricConfiguration("ap.<wfItemType>.contributorAndOwner.validator.date.end.discriminator", sourceItemType, null);
    var wfItemStartDate = object.getDateMap().get(startDateDiscriminator);
    var wfItemEndDate = object.getDateMap().get(endDateDiscriminator);
    var formatter = Packages.org.apache.commons.lang.time.FastDateFormat.getInstance("dd/MM/yyyy");
    var elementWithWrongInterval=[];
```
while (elementSetIterator.hasNext())
{
    var element = elementSetIterator.next();
    var wfItemElementStartDate = element.getDateMap().get("startDate");
    if( wfItemElementStartDate != null && wfItemStartDate &&
        (wfItemElementStartDate.before(wfItemStartDate) || (wfItemEndDate!=null && wfItemElementStartDate.after(wfItemEndDate)) ) ) {
        var discriminator = element.getDiscriminator();
        if(element.getDiscriminator().contains(".")){
            discriminator = element.getDiscriminator().substring(0, element.getDiscriminator().indexOf("."));
        }
        elementWithWrongInterval.push(element.getPersonMap().get(discriminator + "Id").getDisplayValue());
    }
}
if (elementWithWrongInterval.length>0){
    errors.rejectAndLocate("error.contributorAndOwner.wrongInterval", "owner", [elementWithWrongInterval.toString()], "error.contributorAndOwner.wrongInterval");
}

var elementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "owner", wfService);
if (elementSet.size()>0){
    var elementSetIterator = elementSet.iterator();
    var elementToEdit = []; 
    while (elementSetIterator.hasNext())
    {
        var element = elementSetIterator.next();
        var wfItemElementStartDate = element.getDateMap().get("startDate");
        if(wfItemElementStartDate == null && !wfItemElementStartDate){
            var discriminator = element.getDiscriminator();
            if(element.getDiscriminator().contains(".")){
                discriminator = element.getDiscriminator().substring(0, element.getDiscriminator().indexOf("."));
            }
            elementToEdit.push(element.getPersonMap().get(discriminator + "Id").getDisplayValue());
        }
    }
    if (elementToEdit.length > 0){
        errors.rejectAndLocate("error.contributorAndOwner.startDate.required", "owner", [elementToEdit.toString()], "error.contributorAndOwner.startDate.required");
    }
}
var elementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "contributor", wfService);
if (elementSet.size()>0){
    var elementSetIterator = elementSet.iterator();
    var elementToEdit = []; 
    while (elementSetIterator.hasNext())
    {
        var element = elementSetIterator.next();
        var wfItemElementStartDate = element.getDateMap().get("startDate");
        if(wfItemElementStartDate == null && !wfItemElementStartDate){
            var discriminator = element.getDiscriminator();
            if(element.getDiscriminator().contains(".")){
                discriminator = element.getDiscriminator().substring(0, element.getDiscriminator().indexOf("."));
            }
            elementToEdit.push(element.getPersonMap().get(discriminator + "Id").getDisplayValue());
        }
    }
    if (elementToEdit.length > 0){
        errors.rejectAndLocate("error.contributorAndOwner.startDate.required", "owner", [elementToEdit.toString()], "error.contributorAndOwner.startDate.required");
    }
}
while (elementSetIterator.hasNext())
{
    var element = elementSetIterator.next();

    var wfItemElementStartDate = element.getDateMap().get("startDate");
    if(wfItemElementStartDate == null && !wfItemElementStartDate)
    {
        var discriminator = element.getDiscriminator();
        if(element.getDiscriminator().contains(".")){
            discriminator = element.getDiscriminator().substring(0, element.getDiscriminator().indexOf("."))
        }
        elementToEdit.push(element.getPersonMap().get(discriminator + "Id").getDisplayValue());
    }
}

if (elementToEdit.length > 0){
    errors.rejectAndLocate("error.contributorAndOwner.startDate.required", "contributor", [elementToEdit.toString()], "error.contributorAndOwner.startDate.required");
}

ownerDuplicatedRoleValidator
Questa validazione opera sugli elementi owner e viene eseguita SOLO se la configurazione ap.TIPOLOGIA.owner.role.enabled è valorizzata a true.
In questo caso viene valutata anche la configurazione ap.TIPOLOGIA.owner.role.nonDuplicated.csv che contiene l’elenco dei ruoli che possono essere associati al massimo ad un owner.
Un valore accettablile per questa configurazione è ownerRole.scientific, ownerRole.administrative
Non vengono considerati gli eventuali intervalli di validità.

ownerMissingRequiredRoleValidator
Questa validazione verifica la presenza di un tipo di ruolo per i responsabili scientifici, la validazione opera sugli elementi owner e viene eseguita SOLO se la configurazione ap.TIPOLOGIA.owner.role.enabled è valorizzata a TRUE.
In questo caso viene valutata anche la configurazione ap.TIPOLOGIA.owner.role.required.csv che contiene l’elenco dei ruoli che devono essere necessariamente essere presenti.
Un valore accettablile per questa configurazione è ownerRole.scientific, ownerRole.administrative (che sono i codici del dizionario relativo).
Non vengono considerati gli eventuali intervalli di validità.
Configurazioni della ValidateLogic:
- ap.TIPOLOGIA.owner.role.required.csv
var elementRoleEnabled = WfUtil.getParametricConfiguration("ap.<itemType>.owner.role.enabled", object.getWfItemType(), null);
if ("true"==elementRoleEnabled) {
    var roleMap = WfUtil.getElementRoleMap(object, wfService, gaService, "owner", "ap.<itemType>.owner.role.required.csv");
    var iteratorRoleMap=roleMap.keySet().iterator();
    while (iteratorRoleMap.hasNext()){
        var role=iteratorRoleMap.next();
        var elementListWithThisRole=roleMap.get(role);
        if (elementListWithThisRole.size()==0) {
            errors.rejectAndLocate("error.owner.role.required", "owner", [WfUtil.getWfDictionaryByCode(role, wfService).getDisplayValue()], "error.owner.role.required");
        }
    }
}

notEditableByDepartmentValidator
Questa validazione controlla che l'utente corrente NON appartenga al team "Direttori di dipartimento cui negare accesso in modifica ai Progetti/Contratti". Questo è un team con agganciata la funzione "Risorsa per Direttori di dipartimento cui negare accesso in modifica ai Progetti/Contratti" (/ap/department/onlyRead.function)

La visione dipartimentale consente a chiunque appartenga ai team "Profilo di dipartimento per XXX" di accedere con i diritti dipartimentali al flusso. In questo team sono presenti sia i direttori di dipartimento che i referenti amministrativi.
Questa validazione consente l'accesso in scrittura ai soli referenti amministrativi.

getUser();
var gaUserDetail = Packages.it.cilea.ga.authorization.context.GaAuthorizationUserHolder.getUser();
var userId = gaUserDetail.getUserId();
var searchCommand = new Packages.it.cilea.ga.command.TeamUserLinkSearchCommand();
searchCommand.setUserId(userId);
var linkList = gaService.getTeamUserLinkSearchList(searchCommand, 0);
var listIterator = linkList.iterator();
while(listIterator.hasNext()){
    var teamUserLink = listIterator.next();
    if(teamUserLink.getTeamId() == Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap.department.onlyRead.team")){
        errors.reject("error.operation.notallowed");
        break;
    }
}

notCreatableByDepartmentValidator
Questa validazione controlla che l'utente corrente NON appartenga al team "Direttori di dipartimento cui negare accesso in creazione ai Progetti/Contratti". Questo è un team con agganciata la funzione "Risorsa per Direttori di dipartimento cui negare accesso in modifica ai Progetti/Contratti" (/ap/department/onlyRead.function)

La visione dipartimentale consente a chiunque appartenga ai team "Profilo di dipartimento per XXX" di accedere con i diritti dipartimentali al flusso. In questo team sono presenti sia i direttori di dipartimento che i referenti amministrativi.
Questa validazione consente l'accesso in scrittura ai soli referenti amministrativi.

getUser();
var gaUserDetail = Packages.it.cilea.ga.authorization.context.GaAuthorizationUserHolder.getUser();
var authority = gaUserDetail.getCurrentAuthorityIdentifier();
if(authority == "ROLE_DEPARTMENT"){
    var validateLogic = Packages.it.cilea.wf.WfConstant.VALIDATE_LOGIC_MAP.get("notEditableByDepartmentValidator");
    validateLogic.validate(request, object, errors);
}
**uniquePeopleInOwnerAndContributorSetValidator**

Questa validazione controlla che all'interno delle tabelle dei Soggetti Interni NON ci siano nominativi con intervalli di date che si sovrappongono. Sono ammessi nominativi duplicati SOLO nel caso in cui siano specificati su intervalli temporali disgiunti.

Dal punto di vista del modello dati si tratta degli elementi di tipo **owner** e **contributor**.

Per maggiori dettagli cfr. excel modello dati dell'entità in questione.

```javascript
var duplicatedPeopleList = WfUtil.getDuplicatedElementList(object, wfService, gaService, [
  "owner", "contributor"], true, false);
if (!duplicatedPeopleList.isEmpty()) {
  errors.reject("error.contributorAndOwner.duplicate", [WfUtil.toCsv(duplicatedPeopleList)], "error.contributorAndOwner.duplicate");
}
```

**transitionCommentValidator**

Questa validazione, restringe solo ad alcuni attori l'obbligo di inserire la motivazione, per riportare allo stato di bozza (draft).

Viene valutata la configurazione `ap.<itemType>.transitionComment.actor.discriminatorCsv`.

Se non presente o vuota il controllo viene effettuato per TUTTI gli attori, quindi per qualsiasi persona che tenta di riporare l'oggetto in bozza.

Se presente allora il controllo viene fatto SOLO per gli attori indicati nella configurazione.

La configurazione deve essere popolata con i discriminator degli attori interessati tutti separati da virgola, i valori degli attori disponibili sono visibili su Wiki nel modello dati per ogni singolo flusso alla colonna **Attori** (è il valore contenuto tra le parentesi tonde).

Esempio: `ap.prj.transitionComment.actor.discriminatorCsv = helpdesk,helpdesk/research,administrativeOwner,department` si sta indicando che la motivazione per la transizione di stato deve essere motivata per le seguenti persone: Helpdesk, Divisione Ricerca, Referente amministrativo, Organi dipartimentali.

Per maggiori dettagli cfr. modello dati dell'entità in questione

**Configurazioni della ValidateLogic:**

- `ap.TIPOLOGIA.transitionComment.actor.discriminatorCsv`
attachmentValidator
Questa validazione valuta la configurazione `ap.<wfItemType>.attachment.<state>` dove `wfItemType` rappresenta la tipologia gerarchica e `state` rappresenta lo stato di arrivo (in fase di spostamento) o di salvataggio (in fase di salvataggio) in base a dove è stata messa la validazione.
Se la configurazione è presente viene controllato che siano presenti gli allegati con le tipologie presenti nella configurazione. Se la configurazione NON è presente viene controllato che sia presente ALMENO un allegato (indipendentemente dalla tipologia).
La configurazione deve essere popolata con l'elenco dei "code" delle voci di dizionario (ossia le tipologie) separati dal carattere virgola (per recuperare i code è possibile andare alla pagina di gestione dizionari e valutare i risultati nella colonna Codice). Esempio: `ap.spi.attachment.inquiry=spinoffAttachmentType.business-model,spinoffAttachmentType.commission-minutes`: si sta indicando che per andare nello stato inquiry deve essere presente ALMENO un allegato di tipo business-model e ALMENO un allegato di tipo commission-minutes.
E' possibile anche valutare il fatto che deve essere presente ALMENO una delle tipologie, ossia si può indicare che tra le tipologie ABC.. deve essere presente almeno UNA delle tre. Per farlo basta separare le tipologie interessate dal carattere pipe. Esempio: `ap.spi.attachment.active=spinoffAttachmentType.business-model,spinoffAttachmentType.service-agreement,spinoffAttachmentType.commission-minutes|spinoffAttachmentType.senate-deliberation`: si sta indicando che deve essere presente ALMENO un allegato di tipo business-model, deve essere presente ALMENO un allegato di tipo service-agreement, deve essere presente ALMENO un allegato di tipo commission-minutes o senate-deliberation.
Per rendere più leggibile il valore della configurazione è possibile anche sostituire i caratteri virgola con il carattere a capo.
Dal punto di vista del modello dati si tratta degli elementi di tipo attachment.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

Configurazioni della ValidateLogic:

- `ap.TIPOLOGIA.attachment.STATO`  

  ```java
  WFUtil.elementDictionaryCodeChecker(object, wfService, "attachment", "attachmentTypeId", true, request, errors);
  ```

approvalValidator
Questa validazione valuta la configurazione `ap.<wfItemType>.approval.<state>` dove `wfItemType` rappresenta la tipologia gerarchica e `state` rappresenta lo stato di arrivo (in fase di spostamento) o di salvataggio (in fase di salvataggio) in base a dove è stata messa la validazione.
Se la configurazione è presente viene controllato che siano presenti le approvazioni con le tipologie presenti nella configurazione. Se la configurazione NON è presente viene controllato che siano presenti ALMENO le approvazioni con le tipologie presenti nella configurazione.
La configurazione deve essere popolata con l'elenco dei "code" delle voci di dizionario (ossia le tipologie) separati dal carattere virgola (per recuperare i code è possibile andare alla pagina di gestione dizionari e valutare i risultati nella colonna Codice). Esempio: `ap.prj.approval.inquiry=approvalType.department,approvalType.board`: si sta indicando che per andare nello stato inquiry deve essere presente ALMENO una approvazione di tipo approvalType.department e ALMENO una approvazione di tipo approvalType.board.
E' possibile anche valutare il fatto che deve essere presente almeno UNA delle tipologie, ossia si può indicare che tra le tipologie ABC.. deve essere presente almeno UNA delle tre. Per farlo basta separare le tipologie interessate dal carattere pipe. Esempio: `ap.prj.approval.active=approvalType.universitySignature,approvalType.grantSignature,approvalType.contractCommitee|approvalType.spinoffCommitee`: si sta indicando che deve essere presente ALMENO una approvazione di tipo approvalType.universitySignature, deve essere presente ALMENO una approvazione di tipo approvalType.grantSignature, deve essere presente ALMENO una approvazione di tipo approvalType.contractCommitee, deve essere presente ALMENO una approvazione di tipo approvalType.spinoffCommitee.
Per rendere più leggibile il valore della configurazione è possibile anche sostituire i caratteri virgola con il carattere a capo.
Dal punto di vista del modello dati si tratta degli elementi di tipo approval.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

Configurazioni della ValidateLogic:

- `ap.TIPOLOGIA.approval.STATO`  

  ```java
  WFUtil.elementDictionaryCodeChecker(object, wfService, "approval", "approvalType", "attachmentTypeId", true, request, errors);
  ```

wfActionLogicSaveYearFromStartDate
Questa logica calcola il campo anno della scheda ad ogni salvataggio.
Viene estrapolato l'anno dal campo startDate.

```java
it.cilea.wf.logic.action.save.WfActionLogicSaveYearFromStartDate
```

notAllowedValidator
Questa logica scatena un errore e blocca l'azione sulla quale è collegata.

```java
errors.reject("error.operation.notallowed");
```
ownerCannotEditHimself
Questa validazione viene scatenata in visione personale.
Impedisce che l'utente sostituisca come responsabile se stesso con qualcun altro.
Se l'operazione fosse consentita, dopo il salvataggio, l'utente non avrebbe più i diritti per accedere all'oggetto
Per poter utilizzare questa validazione è necessario che nel modello dati sia presente l'attributo ownerIdPreviousValue, che contiene l'id della persona precedente.

```javascript
//log.error("--ownerCannotEditHimself--");
//inutile, ma mi serve per ricordarmi il contesto
var wfItemElement = object;

//controllo se è un nuovo element o meno
//se è un nuovo element allora non avrà la persona settata
if(wfItemElement.getId() != null){

    //prima ricavo la persona precedente a quella che sto aggiungendo
    var previousOwnerId = wfItemElement.getIntegerMap().get("ownerIdPreviousValue");
    //ulteriore controllo per controllare che la persona precedente ci sia non è utile (?)

    //successivamente vado a ricavare la persona che sta editando in questo momento
    var personalUserName = request.getUserPrincipal().getName();
    var personId = gaService.getUser(personalUserName).getPersonId();

    //se le due persone coincidono, vuol dire che sto provando ad editarmi
    if(previousOwnerId == personId){
        //adesso ricavo la persona che sto mettendo in questo momento
        //se non sono io vuol dire che sto cercando di sostituirmi
        var ownerId = wfItemElement.getPersonMap().get("ownerId").getPersonId();

        if(ownerId != previousOwnerId){
            //allora vado a re-sostituire con me stesso e salvo
            var person = gaService.getPerson(previousOwnerId);
            wfItemElement.getPersonMap().put("ownerId", person);
            wsService.saveOrUpdate(wfItemElement);
            errors.reject("error.operation.ownerCannotEditHimself");
        }
    }
}
```

ownerCannotDeleteHimself
Questa validazione viene scatenata in visione personale.
Impedisce che l'utente elimini se stesso dai responsabili scientifici.
Se l'operazione fosse consentita, dopo il salvataggio, l'utente non avrebbe più i diritti per accedere all'oggetto

```javascript
//log.error("--ownerCannotDeleteHimself--");
//inutile, ma mi serve per ricordarmi il contesto
var wfItemElement = object;

//prima ricavo la persona che sto eliminando
var ownerId = wfItemElement.getPersonMap().get("ownerId").getPersonId();

//successivamente vado a ricavare la persona che sta editando in questo momento
var personalUserName = request.getUserPrincipal().getName();
var personId = gaService.getUser(personalUserName).getPersonId();

//se le due persone coincidono, vuol dire che sto provando ad eliminarli
if(ownerId == personId)
    errors.reject("error.operation.ownerCannotDeleteHimself");
```
**administrativeOwnerCannotDeleteHimself**

Questa validazione viene scatenata in visione personale.

Impedisce che l'utente elimini se stesso dai responsabili amministrativi.

Se l'operazione fosse consentita, dopo il salvataggio, l'utente non avrebbe più i diritti per accedere all'oggetto.

```javascript
// inutile, ma mi serve per ricordarmi il contesto
var wfItemElement = object;

// prima ricavo la persona che sto eliminando
var ownerId = wfItemElement.getPersonMap().get("administrativeOwner").getPersonId();

// successivamente vado a ricavare la persona che sta editando in questo momento
var personalUserName = request.getUserPrincipal().getName();
var personId = gaService.getUser(personalUserName).getPersonId();

// se le due persone coincidono, vuol dire che sto provando ad eliminarmi
if (ownerId == personId) {
    errors.reject("error.operation.administrativeOwnerCannotDeleteHimself");
}
```

**internalRepresentativeCannotDeleteHimselfElementValidator**

Questa validazione viene scatenata in visione personale.

Impedisce che l'utente elimini se stesso dai delegati.

Se l'operazione fosse consentita, dopo il salvataggio, l'utente non avrebbe più i diritti per accedere all'oggetto.

```javascript
var wfItemElement = object;
var wfItemType = wfService.getWfItemType(wfItemElement.getWfItem().getWfItemTypeId());
var rootWfItemType = wfItemType.getRootWfItemType();
var ownerId = wfItemElement.getPersonMap().get("internalRepresentative").getPersonId();
var personalUserName = request.getUserPrincipal().getName();
var personId = gaService.getUser(personalUserName).getPersonId();

if (ownerId == personId) {
    errors.reject("error." + rootWfItemType.getIdentifier().toLowerCase() + ".internalRepresentative.cannotLeave");
}
```

**checkCreationPermissionsValidator**

Questa validazione viene scatenata in fase di creazione dell'oggetto.

Controlla che l'utente corrente abbia i permessi per creare la tipologia di oggetto in questione.

Anche se agli utenti viene presentata sempre la lista dei tipi di oggetto che possono creare, questo ulteriore check serve per prevenire eventuali "hacking" di utenti che modificano volutamente il markup del form inserendo un tipo di oggetto per il quale non hanno le autorizzazioni.

```javascript
// da controllare solo in fase di prima creazione
if (wfItem.getId() == null) {
    var isCreateAllowed = Packages.it.cilea.wf.util.WfUtil.isWfItemTypeAllowedForCurrentUser(wfItem.getWfItemTypeId(), applicationContext);
    if (!isCreateAllowed) {
        errors.reject("error.ap.wfItemType.creationNotAllowed");
    }
}
```

**sameDefinitionForGenericItemValidator**

Questa validazione viene scatenata in fase di cambio della tipologia dell'oggetto.

Controlla che la nuova tipologia selezionata abbia agganciato lo stesso flusso della tipologia precedente.

```javascript
// estraggo il wfItemId
// log.error("entro nel validator sameDefinitionForGenericItemValidator");
var myNEWWfItemTypeId = object.getWfItemTypeId();
var myOLDWfItemTypeId = object.getIntegerMap().get("wfItemTypeIdPreviousValue");
```
if (myOLDWfItemType==null)  
    throw "To use changeItemTypeValidator you MUST specify
    wfItemTypeIdPreviousValue hidden value";

//estraggo il wfItemId  
var myNEWWfItemType = object.getWfItemId();  
var myOLDWfItemType = object.getIntegerMap().get("wfItemTypeIdPreviousValue");  
var sourceItemType = wfService.getWfItemType(myNEWWfItemType);  

if (myOLDWfItemType==null)  
    throw "To use changeItemTypeValidator you MUST specify
    wfItemTypeIdPreviousValue hidden value";

//confronto le due tipologie, se sono diverse vuol dire che il  
cambio di tipologia non è compatibile
if(myNEWWfItemType != myOLDWfItemType){  
    var gaUserDetail = GaAuthorizationUserHolder.getUser();  
    var authority = gaUserDetail.getCurrentAuthorityIdentifier();  
    if(!authority == "ROLE_ADMIN" || !gaUserDetail.hasGlobalAuthorities("/ADMIN.  
        profile", "ROLE_ADMIN") && Boolean.TRUE.equals(object.getBooleanMap().get("legacy"))){  
        if(WfUtil.getParametricConfiguration("ap.<wfItemType>.itemTypeChange.  
            enabled", sourceItemType, null).equalsIgnoreCase("true")){  
            var saveLogicIdentifier = Packages.it.cilea.wf.util.WfUtil.  
                getParametricConfiguration("ap.<wfItemType>.identifier.logic", sourceItemType, null);  
            if(saveLogicIdentifier == null){  
                object.setWfItemId(myOLDWfItemType);  
                //errors.reject("error.ap.itemTypeError.startLogic");  
            } else {  
                object.setWfItemId(myOLDWfItemType);  
                errors.reject("error.ap.itemTypeError");  
            }  
        } else {  
            var isCreateAllowed = Packages.it.cilea.wf.util.WfUtil.  
                isWfItemTypeAllowedForCurrentUser(myOLDWfItemType, applicationContext);  
            if(!Packages.java.lang.Boolean.TRUE.equals(isCreateAllowed)){  
                object.setWfItemId(myOLDWfItemType);  
                errors.reject("error.ap.itemTypeErrorError");  
            }  
        }  
    }  
}

changelItemTypeForNotSynchronizedItemValidator  
Questa validazione viene scatenata in fase di cambio di Tipo dell'oggetto.  
Non è possibile cambiare la tipologia, SE la scheda è già stata sincronizzata con U-GOV o l'utente non possiede i permessi sulla nuova tipologia selezionata.  
Il cambio di tipologia è concesso se la configuration ap.<wfItemType>.ItemTypeChange.enabled è uguale a true.  
Introdotta la possibilità di cambio tipologia per Assistenza Tecnica (anche con scheda già sincronizzata con UGOV PJ e senza valutare la configurazione di cui sopra) in caso di scheda con flag legacy attivo.  
Configurazioni della ValidateLogic:
  • ap.TIPOLOGIA.ItemTypeChange.enabled
changelItemTypeForSynchronizedItemValidator
Questa validazione viene scatenata in fase di cambio di Tipo dell'oggetto in un determinato stato del flusso.
Non è possibile cambiare la tipologia in questo stato del flusso in quanto non compatibile con la sincronizzazione già avvenuta.
Introdotta la possibilità di cambio tipologia per Assistenza Tecnica in caso di scheda con flag legacy attivo.

```javascript
//estraggo il wfItemId
var myNEWWfItemType = object.getWfItemId();
var myOLDWfItemType = object.getIntegerMap().get("wfItemTypeIdPreviousValue");
if (myOLDWfItemType==null)
    throw "To use changeItemTypeValidator you MUST specify wfItemTypeIdPreviousValue hidden value";
if(myNEWWfItemType != myOLDWfItemType){
    var gaUserDetail = GaAuthorizationUserHolder.getUser();
    var authority = gaUserDetail.getCurrentAuthorityIdentifier();
    if(!authority == "ROLE_ADMIN" || !(gaUserDetail.hasGlobalAuthorities("/ADMIN.profile", "ROLE_ADMIN") && Boolean.TRUE.equals(object.getBooleanMap().get("legacy")))){
        object.setWfItemTypeId(myOLDWfItemType);
        errors.reject("error.ap.itemTypeError.step2");
    }
}
```

roleElementValidator
Questa validazione controlla i ruoli per i possibili elementi: **owner** (Responsabili interni), **internalOrganizationUnit** (Unità organizzative interne), **partner** (Partner).
Viene valutato l'attributo `element.wfDictionaryMap[roleId]`.
Vengono valutate le seguenti configurazioni (che di default NON sono presenti), se non è presente ALCUNA configurazione la logica NON effettuerà alcun controllo.

- **Uniqueness**: `ap.<HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE>.<elementDiscriminator>.role.uniqueness.csv`
  Se presente, viene controllato che ogni ruolo sia presente al massimo una volta.
  Se non presente non viene fatto alcun controllo.
  E' necessario specificare i ruoli degli elementi a cui effettuare il controllo sopra indicato, per farlo bisogna valorizzare la configurazione con l'elenco dei code dei ruoli (quello che si vede da interfaccia dei dizionari alla colonna Codici) separati da virgola.

- **Required**: `ap.<HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE>.<elementDiscriminator>.role.required.csv`
  Se presente, viene controllato che ogni ruolo sia presente almeno una volta.
  Se non presente non viene fatto alcun controllo.
  E' necessario specificare i ruoli degli elementi a cui effettuare il controllo sopra indicato, per farlo bisogna valorizzare la configurazione con l'elenco dei code dei ruoli (quello che si vede da interfaccia dei dizionari alla colonna Codici) separati da virgola.

Alcuni esempi:
- **configurazione ap.prj.owner.role.uniqueness.csv = ownerRole.administrative**, si sta indicando che per i Responsabili scientifici dei Progetti deve essere presente al massimo un responsabile marcato con il ruolo **Responsabile Scientifico e Amministrativo**
- **configurazione ap.rsc.internalOrganizationUnit.role.required.csv = researchCentreRole.main-seat, researchCentreRole.other-seats-involved**, si sta indicando che per i Dipartimenti del Centro di ricerca devono essere presenti almeno due dipartimenti: uno marcato con il ruolo **Sede principale** e l'altro marcato con il ruolo **Altre strutture coinvolte**

Dal punto di vista del modello dati si tratta degli attributi **owner**, **internalOrganizationUnit** e **partner**.
Per maggiori dettagli cfr. modello dati dell'entità in questione

**Configurazioni della ValidateLogic:**
- `ap.HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.ELEMENTO.role.uniqueness.csv`
- `ap.HIERARCHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.ELEMENTO.role.required.csv`

```javascript
if(WfUtil.isWfItem(object)){
    var allDiscriminatorElement="owner","internalOrganizationUnit","partner";
    for(var i=0; i < allDiscriminatorElement.length; i++)   {
```
var discriminatorElement = allDiscriminatorElement[i];
var requiredConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".role.required", object.getWfItemType(), true);
var uniquenessCsvConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".role.uniqueness.csv", object.getWfItemType(), true);
var requiredCsvConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".role.required.csv", object.getWfItemType(), true);
if(StringUtils.equalsIgnoreCase(requiredConfig, "true")){
    WfUtil.roleRequiredChecker(object, discriminatorElement, wfService, errors);
}
if(uniquenessCsvConfig != null || requiredCsvConfig != null){
    WfUtil.roleCsvChecker(object, null, discriminatorElement, uniquenessCsvConfig, requiredCsvConfig, wfService, errors);
}
} else if(WfUtil.isWfItemElement(object)){
    var discriminatorElement = StringUtils.substringBefore(object.getDiscriminator(), ".");
    discriminatorElement = new String(discriminatorElement).toString();
    var requiredConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".role.required", object.getWfItem().getWfItemType(), true);
    var uniquenessCsvConfig = WfUtil.getParametricConfiguration("ap.<wfItemType>." + discriminatorElement + ".role.uniqueness.csv", object.getWfItem().getWfItemType(), true);
    if(StringUtils.equalsIgnoreCase(requiredConfig, "true")){
        WfUtil.roleRequiredChecker(object, discriminatorElement, wfService, errors);
    }
    if(uniquenessCsvConfig != null){
        WfUtil.roleCsvChecker(object.getWfItem(), object, discriminatorElement, uniquenessCsvConfig, null, wfService, errors);
    }
} else {
    throw "Cannot validate object: type not allowed";
}

requiredAndUniquenessElementValidator
Questa validazione controlla i dati inseriti per gli elementi di tipo: internalOrganizationUnit (Unità organizzative interne). Vengono valutate le seguenti configurazioni (che di default NON sono presenti), se non è presente ALCUNA configurazione la logica NON effettuerà alcun controllo.

- **Uniqueness:** ap.<HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE>.<elementDiscriminator>.uniqueness
  Se presente e valorizzata a true, viene controllato che ogni elemento NON sia presente più di una volta.
  In tutti gli altri casi non verrà effettuato alcun controllo.

- **Required:** ap.<HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE>.<elementDiscriminator>.required
  Se presente e valorizzata a true, viene controllato che per ogni tipo di elemento sia presente almeno un elemento.
  In tutti gli altri casi non verrà effettuato alcun controllo.

Alcuni esempi:

- configurazione ap.prj.internalOrganizationUnit.uniqueness = true, si sta indicando che per le Unità organizzative interne dei Progetti NON può essere inserita più volte la stessa Unità organizzativa
- configurazione ap.rsc.internalOrganizationUnit.required = true, si sta indicando che per le Unità organizzative interne dei Centri di Ricerca DEVE essere presente ALMENO una Unità organizzativa

Dal punto di vista del modello dati si tratta degli attributi internalOrganizationUnit.
Per maggiori dettagli cfr. modello dati dell'entità in questione

Configurazioni della ValidateLogic:

- ap.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.ELEMENTO.role.uniqueness.csv
- ap.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.ELEMENTO.role.required.csv

```javascript
var allDiscriminatorElement = ("internalOrganizationUnit":["ouId"]);
```
if(WfUtil.isWfItem(object)){
    for (var discriminatorElement in allDiscriminatorElement)
        var discriminatorElementData = allDiscriminatorElement[discriminatorElement][0];
        if(StringUtils.equalsIgnoreCase(WfUtil.getParametricConfiguration("ap.<wfItemType>" + discriminatorElement + ".required", object.getWfItemType(), true), "true"){
            var wfElementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", discriminatorElement, wfService);
            if(wfElementSet.isEmpty()){
                errors.rejectAndLocate("error." + discriminatorElement + ".required", discriminatorElement);
            }
        }
    if(StringUtils.equalsIgnoreCase(WfUtil.getParametricConfiguration("ap.<wfItemType>" + discriminatorElement + ".uniqueness", object.getWfItemType(), true), "true"){var isAlreadyPresent=WfUtil.contains(object, null, [discriminatorElementData], discriminatorElement, wfService);
        if (isAlreadyPresent){
            errors.rejectAndLocate("error." + discriminatorElement + ".uniqueness", discriminatorElement);
        }
    }
} else if(WfUtil.isWfItemElement(object)){
    var discriminatorElement = StringUtils.substringBefore(object.getDiscriminator(), ".");
    discriminatorElement = new String(discriminatorElement).toString();
    var discriminatorElementData = allDiscriminatorElement[discriminatorElement][0];
    if(StringUtils.equalsIgnoreCase(WfUtil.getParametricConfiguration("ap.<wfItemType>" + discriminatorElement + ".uniqueness", object.getWfItem().getWfItemType(), true), "true"){var isAlreadyPresent=WfUtil.contains(object.getWfItem(), object, [discriminatorElementData], discriminatorElement, wfService);
        if (isAlreadyPresent){
            errors.reject("error." + discriminatorElement + ".uniqueness");
        }
    }
} else {
    throw "Cannot validate object: type not allowed";
}

contactEmailElementValidator
Questa validazione controlla che in fase di salvataggio del fragment contact che il formato della mail sia corretto.
Dal punto di vista del modello dati si tratta degli attributi stringMap[description]
Dal punto di vista del modello dati si tratta degli elementi di tipo contact.
Per maggiori dettagli cfr. modello dati dell’entità in questione.

var wfItemElement=object;
if(wfItemElement.getGaDictionaryMap().get("contactType") != null && wfItemElement.getStringMap().get("description") != null){
    var contactTypeId=wfItemElement.getGaDictionaryMap().get("contactType").getId();
    var contactTypeDictionary=gaService.getGaDictionary(contactTypeId);
    if (contactTypeDictionary.getDescription().equals("mail")){
        var pattern = Packages.java.util.regex.Pattern.compile("^[\-A-Za-z0-9-]+\([\-A-Za-z0-9-]+\)*\([\-A-Za-z0-9-]+\)\[[\-A-Za-z0-9-]\]{1}[$]");
        var matcher = pattern.matcher(wfItemElement.getStringMap().get("description"));
        if (!matcher.matches()) {
            errors.rejectValue("stringMap[description]", "error.contact.mail.invalidValue");
        }
    }
}
**contactTypeMainElementValidator**
Questa validazione controlla che in fase di salvataggio del fragment contact che non venga marcato più di un contatto per tipologia.
Dal punto di vista del modello dati si tratta degli attributi gaDictionaryMap\[contactType\] Dal punto di vista del modello dati si tratta degli elementi di tipo contact.
Per maggiori dettagli cfr. modello dati dell’entità in questione.

```java
var wfItemElement=object;
var currentContactId=wfItemElement.getId();
var trueBoolean= Packages.java.lang.Boolean(true);

//prima controllo se è principale o meno, in modo da evitare un passaggio
if(wfItemElement.getGaDictionaryMap().get("contactType") != null && wfItemElement.
getStringMap().get("description") != null)
    if(trueBoolean.compareTo(wfItemElement.getBooleanMap().get("mainContact")) == 0){
        //ricavo wfItem completo, non mi serve il fragment dato che non posso
        var wfItem= wfService.getWfItem(wfItemElement.getWfItemId());
        var contactWfElementSet=Packages.it.cilea.core.fragment.util.
FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","contact",
wfService);
        var contactWfElementSetIterator=contactWfElementSet.iterator();
        var errorMessage= "error.contact.tooManyMainContacts.";
        while(contactWfElementSetIterator.hasNext()){ 
            var contactElement=contactWfElementSetIterator.next();
            if(trueBoolean.compareTo(contactElement.getBooleanMap().get
("mainContact")) == 0 && !contactElement.getId().equals(currentContactId)){
                var contactTypeId= wfItemElement.getGaDictionaryMap().
get("contactType").getId();
                if(contactTypeId==contactElement.getGaDictionaryMap().
get("contactType").getId()){
                    var contactTypeDesc=gaService.getGaDictionary
(contactTypeId).getDescription();
                    errorMessage+=contactTypeDesc;
                    errors.reject(errorMessage);
                }
            }
        }
    }
```

**itemTypeNotWithdrawnValidator**
Questa validazione controlla che in fase di salvataggio non venga selezionata come tipologia di un item, una tipologia che è stata ritirata (ovvero per le quali non è possibile creare nuovi item)
È possibile mantenere un item che è già presente in quella tipologia

```java
var newWfItemTypeId = object.getWfItemTypeId();
var newWfItemType = wfService.getWfItemType(newWfItemTypeId);

//log.error("wfItemType: " + newWfItemType.getDisplayValue());
//log.error("getWithdrawn: " + newWfItemType.getWithdrawn());

if(newWfItemType.getWithdrawn() == true)
    //ho trovato una tipologia eliminata
    //adesso controllo se prima fosse la stessa, perché devo mantenere comunque il
pregresso

var oldWfItemTypeId = object.getIntegerMap().get("wfItemTypeIdPreviousValue");
var oldWfItemType = wfService.getWfItemType(oldWfItemTypeId);

//controllo inoltre che la previousValue sia presente
if (oldWfItemTypeId == null)
    throw "To use itemTypeNotWithdrawnValidator you MUST specify"
Questa logica serve per il ricalcolo dell'identificativo della scheda in caso di cambio di tipologia. Il ricalcolo viene effettuato SOLO se la variabile di configurazione ap.<wfItemType>.identifier.recalculation.enabled è settata a true o non è valorizzata.

E' necessario inserirla come save logic per tutti gli stati nei flussi progetti e contratti che precedono la sincronizzazione con Ugov PJ.

Configurazioni della ActionLogic:

- ap.TIPOLOGIA.identifier.recalculation.enabled

Questa logica serve per l'aggiornamento dell'anno della scheda in caso di cambio di data. L'aggiornamento viene effettuato quando si recupera una data e se l'anno della data è diverso dal valore presente nel campo anno (year).

E' necessario inserirla come save logic per tutti gli stati, in tutti i flussi tranne quelli di Public Engagement, perché in quei flussi l'anno viene gestito con una semantica differente.

La data viene recuperata dalla configurazione ap.<wfItemType>.owner.position.date, dove può essere indicata direttamente (esempio: startDate), oppure se la configurazione viene valorizzata con CURRENT o CURRENT_OR_LAST, verrà effettuato un recupero della data presente nel metadato startData, se non presente verrà recuperata la data presente nel metadato proposalStartDate, se non presente verrà recuperata la data presente nel metadato applicationDate.

Configurazioni della ActionLogic:

- ap.TIPOLOGIA.owner.position.date
if(wfItem.getYear() != new Packages.java.lang.Integer(calendar.get(Calendar.YEAR))){
    wfItem.setYear(new Packages.java.lang.Integer(calendar.get(Calendar.YEAR)));
}

**wfActionLogicSaveOwnerStartDate**

Questa logica effettua l'inizializzazione della data di inizio dove iniettata e SOLO se presente e valorizzata a true la configurazione **ap.TYPE.owner.startDate.prepopulate** (ad esempio **ap.con.owner.startDate.prepopulate**).

La data di inizio viene valorizzata con la data di inizio **dateMap[startDate]** recuperata dall'oggetto radice.

Inoltre la logica effettua l'inizializzazione SOLO se il campo NON viene nascosto dalla logica standard per nascondere i campi/sezioni.

La logica viene eseguita in fase di salvataggio dell'oggetto (wfItem).

**Configurazioni della ActionLogic:**

- **ap.TIPOLOGIA.ELEMENTO.startDate.prepopulate**

```javascript
function isEnabled(wfItem) {
    var enabled = false;
    var rootItemType = wfItem.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase();
    var startDatePrepopulateConf = ConfigurationUtil.getConfigValue("ap." + rootItemType + ".owner.startDate.prepopulate");
    if (startDatePrepopulateConf != null && "true" == startDatePrepopulateConf) {
        var hiddenManagerCsv = WfUtil.getHiddenConfigurationForm(wfItem.getId(), Packages.java.lang.Integer.valueOf(request.getParameter("wfTaskId")), applicationContext);
        if((hiddenManagerCsv != null && !hiddenManagerCsv.contains("owner.dateMap[startDate]"))) {
            enabled = true;
        }
    }
    return enabled;
}

if(isEnabled(wfItem)) {
    var startDateWfItem = wfItem.getDateMap().get("startDate");
    if(startDateWfItem != null) {
        var ownerElementSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "owner", wfService);
        if(!ownerElementSet.isEmpty()){var ownerElementSetIterator = ownerElementSet.iterator();
            while(ownerElementSetIterator.hasNext()){var ownerElement = ownerElementSetIterator.next();
                if(ownerElement.getDateMap().get("startDate") == null) {
                    ownerElement.getDateMap().put("startDate", startDateWfItem);
                }
            }
        }
    }
}
```

**wfActionLogicTransitionLogger**

Questa logica serve per la gestione del logging delle transizioni di stato nel metadato **transitionLog**. 
Se il metadato NON è presente nel modello dati, non sarà visibile nell'interfaccia web.

Questa logica viene inserita come enter action logic per tutti gli stati nei flussi per i quali si vuole abilitare il logging delle transizioni di stato.
Se necessario, è possibile richiedere all'utente che sta effettuando la transizione di stato di inserire un messaggio che giustifichi la transizione di stato: questo messaggio verrà conservato nel log.

La gestione del messaggio opzionale è gestito con il metadato **transitionComment** che deve essere configurato come metadato @Required. per la validazione di enter.

```javascript
var justExternallyCreated = null;
if(request != null)
```
justExternallyCreated=request.getParameter("justExternallyCreated");
if ((Packages.java.lang.Boolean.TRUE.equals(wfItem.isMaster())) && (!
justExternallyCreated || (justExternallyCreated=="true" && Packages.java.lang.Boolean.TRUE.equals(wfItem.
isMostValidated())))) {
    var formatter=new Packages.java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    var currentDate="["+formatter.format(new Packages.java.util.Date())+]";
    var user=Packages.it.cilea.ga.authorization.context.GaAuthorizationUserHolder.
    getUser();
    var message="";
    if(Packages.org.apache.commons.lang.StringUtils.isNotBlank(wfItem.getClobMap().
    get("transitionsLog"))){
        message+="_______________________________________________________________" + "
    } else {
        wfItem.getClobMap().put("transitionsLog", "");
        if (user!=null){
            message=currentDate+"<strong">+user.getLastName()+"+user.
            getFirstName()+</strong> ("+user.getUsername()+")+"+user.
            getUsername()+</strong> ("+user.getUsername()+")+"
        } else {
            message=currentDate+"<strong>system</strong>"
        };
        message+="Transizione verso "<strong">+wfItem.getWfState().getDisplayValue()+"<
        /strong>" ("+wfItem.getWfState().getDescription()+");
        if (Packages.org.apache.commons.lang.StringUtils.isNotBlank(wfItem.
        getStringMap().get("transitionComment")))
            message+="<strong>Motivazione transizione di stato</strong>
            <i>"+wfItem.getStringMap().get("transitionComment")+"</i>
        message+=""; //message+="i">+wfItem.getStringMap().get("transitionComment")+"<
        /i>");
    } else {
        message="";
    }
}
wfItem.getClobMap().put("transitionsLog"), wfItem.getClobMap().get
("transitionsLog")+message);
wfItem.getStringMap().put("transitionComment", null);
}

wfActionLogicWarningHandler
Questa logica serve per la gestione dei messaggi di warning.
E' necessario inserirla come enter action logic per tutti gli stati di tutti i flussi che prevedono messaggi di warning.
La gestione dei warning è pilotata dall'attributo booleano warningAcknowlegement che deve essere configurato come validazione di enter con il validator @Boolean.TRUE specificando di volta in volta il messaggio di warning da presentare. Per maggiori dettagli cfr. modello dati dell'entità in oggetto

wItem.getBooleanMap().put("warningAcknowlegement", null);

isNotLegacy
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera l'attributo legacy, tipicamente con etichetta "Disattivazione validazioni" (cfr il modello dati)
Per tutti gli oggetti che hanno questo attributo valorizzato a true vengono disattivate le validazioni e le eventuali sincronizzazioni.

object.getBooleanMap().get('legacy')==null
|| Packages.java.lang.Boolean.FALSE.equals(object.getBooleanMap().get('legacy'))
isOwnerCreationAllowed
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera la variabile di configurazione `ap.xxxx-flow.owner.create` dove `xxxx` è il nome del flusso.
Per tutti gli oggetti che hanno questa variabile di configurazione settata a true o non valorizzata vengono abilitate le validazione.
Per tutti gli oggetti che hanno questa variabile di configurazione settata a false vengono disabilitate le validazione.
Configurazioni della ValidateLogic:
- `ap.xxxx-flow.owner.create`

```java
{
    "true"==Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap."+ object.getWfItemType().getWfDefaultDefinition().getDescription() +".owner.create")
    ||
    Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap."+ object.getWfItemType().getWfDefaultDefinition().getDescription() +".owner.create")==null
}
```

isOwnerCreationForOtherPeopleAllowed
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera le variabili di configurazione `ap.xxxx-flow.owner.create` e `ap.xxxx.personal-view.create-for-other-people.enabled` dove `xxxx` è il code dell'entità (in minuscolo).
Per tutti gli oggetti che hanno `ap.xxxx-flow.owner.create` settata a true o non valorizzata e per tutti gli oggetti che hanno `ap.xxxx.personal-view.create-for-other-people.enabled` settata a false o non valorizzata vengono abilitate le validazione.
In tutti gli altri casi vengono disabilitate le validazione.
Configurazioni della ValidateLogic:
- `ap.xxxx-flow.owner.create`
- `ap.xxxx.personal-view.create-for-other-people.enabled`

```java
{
    Packages.java.lang.Boolean.TRUE.equals(Packages.java.lang.Boolean.valueOf(Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getWfDefaultDefinition().getDescription() + ".owner.create")))
    ||
    Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getWfDefaultDefinition().getDescription() + ".owner.create") == null
    &&
    Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase() + ".personal-view.create-for-other-people.enabled") == null
    ||
    !Packages.java.lang.Boolean.TRUE.equals(Packages.java.lang.Boolean.valueOf(Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase() + ".personal-view.create-for-other-people.enabled")))
}
```

isOwnerCreationForbidden
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Questa regola considera la variabile di configurazione `ap.xxxx-flow.owner.create` dove `xxxx` è il nome del flusso.
Per tutti gli oggetti che hanno questa variabile di configurazione settata a true o non valorizzata vengono disabilitate le validazione.
Per tutti gli oggetti che hanno questa variabile di configurazione settata a false vengono abilitate le validazione.
Configurazioni della ValidateLogic:
- `ap.xxxx-flow.owner.create`

```java
{
    Packages.java.lang.Boolean.TRUE.equals(Packages.java.lang.Boolean.valueOf(Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase() + ".owner.create")))
    ||
    Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase() + ".owner.create") == null
    &&
    Packages.it.cilea.core.configuration.util.ConfigurationUtil.getConfigValue("ap." + object.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase() + ".personal-view.create-for-other-people.enabled") == null
}
```
childAndParentDetectorDeleteValidator
Questa è una validazione che viene chiamata in ogni flusso di Laboratori, Gruppi di Ricerca, Grandi Attrezzature, Public-Engagement, Bandi di finanziamento e di Progetti quando si prova ad eliminare l'oggetto collegato. Essa impedisce che si possa eliminare l'oggetto se è collegato (sia da parte di padre, sia da parte di figlio) ad un altro item indicato qui sopra. In caso la logica trovasse questi collegamenti, verrà presentato un messaggio di errore che mostrerà un elenco con tutti gli oggetti collegati. Il messaggio di errore viene ricavato dall'etichetta con chiave error.delete.link.childAndParentDetected
Per maggiori dettagli cfr. modello dati dell'entità specifica.

```javascript
var wfItemFull = wfService.getWfItem(wfItem.getWfItemId());
var myList="";
var childWfItemLinkSet = wfItemFull.getChildWfItemLinkSet();
var childWfItemLinkSetIterator = childWfItemLinkSet.iterator();
while (childWfItemLinkSetIterator.hasNext()){ 
  var childLink = childWfItemLinkSetIterator.next();
  var childWfItem = childLink.getChild();
  //considero solo quelli che non sono version, edit, new, deleted e non eliminati
  if(!childLink.getDiscriminator().equals('version') && !childLink.getDiscriminator().contains('edit-post') && !childLink.getDiscriminator().contains('new-post') && !childLink.getDiscriminator().contains('deleted') && childWfItem!=null && childWfItem.getWithdrawn() == false){
    myList+=childWfItem.getIdentifier()+", ";
  }
}
var parentWfItemLinkSet = wfItemFull.getParentWfItemLinkSet();
var parentWfItemLinkSetIterator = parentWfItemLinkSet.iterator();
while (parentWfItemLinkSetIterator.hasNext()){ 
  var parentLink = parentWfItemLinkSetIterator.next();
  var parentWfItem = parentLink.getParent();
  //considero solo quelli che non sono version, edit, new, deleted e non eliminati
  if(!parentLink.getDiscriminator().equals('version') && !parentLink.getDiscriminator().contains('edit-post') && !parentLink.getDiscriminator().contains('new-post') && !parentLink.getDiscriminator().contains('deleted') && parentWfItem!=null && parentWfItem.getWithdrawn() == false){
    myList+=parentWfItem.getIdentifier()+", ";
  }
}
if(myList != ""){
  myList = new String(myList).toString();
  myList = myList.substring(0, myList.length() - 2);
  var messageContainer = [myList];
  errors.reject("error.delete.link.childAndParentDetected", messageContainer,
  "error.delete.link.childAndParentDetected.default");
}
```

**elementStartDatePrepopulateLogic**
Questa logica effettua l'inizializzazione della data di inizio dove iniettata e SOLO se presente e valorizzata a true la configurazione ap.TYPE.ELEMENT_DISCRIMINATOR.startDate.prepopulate (ad esempio ap.prj.owner.startDate.prepopulate).
La data di inizio viene valorizzata con la data di inizio dateMap[startDate] recuperata dall'oggetto radice.
Inoltre la logica effettua l'inizializzazione SOLO se il campo NON viene nascosto dalla logica standard per nascondere i campi/sezioni.
La logica viene eseguita in fase di creazione dell'oggetto (wfItemElement).
Configurazioni della ValidateLogic:
- ap.TIPOLOGIA.ELEMENTO.startDate.prepopulate

```javascript
function isEnabled(wfItem, element) {
  var enabled = false;
  var elementDiscriminator = FragmentUtil.getFragmentableInfo(element).
```
getDiscriminator();
var rootItemType = wfItem.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase();
var startDatePrepopulateConf = ConfigurationUtil.getConfigValue("ap." + rootItemType + ".startDate.prepopulate");
if (startDatePrepopulateConf != null && "true" == startDatePrepopulateConf) {
    var hiddenManagerCsv = WfUtil.getHiddenConfigurationForm(wfItem.getId(), Packages.java.lang.Integer.valueOf(request.getParameter("wfTaskId")), applicationContext);
    if((hiddenManagerCsv != null && !hiddenManagerCsv.contains(elementDiscriminator + ".dateMap[startDate]")) || hiddenManagerCsv == null) {
        enabled = true;
    }
}

var element = object;
var wfItem = wfService.getWfItem(element.getWfItemId());
var elementStatus = FragmentUtil.getFragmentableInfo(element).getStatus().name();
if (elementStatus == "NEW") {
    if(isEnabled(wfItem, element)) {
        var startDateWfItem = wfItem.getDateMap().get("startDate");
        if(startDateWfItem != null) {
            element.getDateMap().put("startDate", startDateWfItem);
        }
    }
}