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

```
                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                        paramMap.put("withdrawn", "0");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);

                        }
                }
                true;
```

wfStartLogicParentCall
Questa logica viene eseguita in fase di creazione di un nuovo bando.
Se attiva la creazione con associazione a un bando radice, tramite la variabile di configurazione
ap.prj.create-by-call.enabled valorizzata a **true**, la logica effettua il collegamento del Bando al Bando radice
SOLO nel caso in cui si scelga una gestione gerarchica del bando (metadato isHierarchical).
Se la gestione NON è gerarchica o la configurazione NON è true, verrà creato un figlio senza padre.
(cfr dettaglio delle altre configurazioni disponibili)
Configurazioni della StartLogic:

- ap.call.createWithParent.enabled

```
                        var isParentEnabled = ConfigurationUtil.getConfigValue("ap.call.createWithParent.
enabled");
                        var isHierarchical = wfItem.getBooleanMap().get("isHierarchical");
                        if (isHierarchical && StringUtils.equalsIgnoreCase(isParentEnabled, "true")){
                                var isParent = wfItem.getBooleanMap().get("isParent");
                                if(!isParent){
                                        var callId = wfItem.getIntegerMap().get("parentCallId");
                                        if (callId != null) {
                                                var call = wfService.getWfItem(callId);
                                                var callParentLink = new Packages.it.cilea.wf.model.
WfItemLink();
                                                callParentLink.setDiscriminator("callParentLink");
                                                callParentLink.setParentId(callId);
                                                callParentLink.setChildId(wfItem.getId());
                                                wfService.saveOrUpdate(callParentLink);
                                                wfItem.getParentLinkSet().add(callParentLink);
                                        }
                                }
                        } else {
                                wfItem.getBooleanMap().put("isParent", false);
                                wfItem.getBooleanMap().put("isHierarchical", false);
                        }
                        wfItem.getIntegerMap().put("parentCallId", null);
                        true;
```

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

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

wfActionLogicSaveHierarchicalCall
Questa logica di sistema gestisce il metadato **isParent** e **isHierarchical**.
Quando un bando figlio viene sganciato da un padre, viene aggiornato un flag (isHierarchical) che lo identificherà come bando senza gerarchia.
Quando un bando figlio viene agganciato ad un padre, viene aggiornato un flag (isHierarchical) che lo identificherà come bando con gerarchia.
Per maggiori dettagli fare riferimento al modello dati dell'entità Bando

isCreateProjectByCallEnabled
Questa è un'applicability rule che pilota le validazioni a cui è agganciata.
Se la variabile **ap.prj.create-by-call.enabled** è settata a true la validazione è abilitata, in tutti gli altri casi NON è abilitata.
Configurazioni della ValidateLogic:

- ap.prj.create-by-call.enabled

isCreateWithParentCallEnabled
Questa è un'applicability rule che pilota le validazioni a cui è agganciata.
Se la variabile **ap.call.createWithParent.enabled** è settata a true la validazione è abilitata, in tutti gli altri casi NON è abilitata.
Configurazioni della ValidateLogic:

- ap.call.createWithParent.enabled

isHierarchicalCall
Questa è un'applicability rule che viene valutata prima di eseguire alcune validazioni.
Se ritorna true allora vengono eseguite le validazioni altrimenti no.
Questa regola considera il campo "Creazione gerarchia di bandi?" (attributo **booleanMap[isHierarchical]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesta la compilazione del campo "Bando radice" (attributo **booleanMap[isParent]**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

linkedProjectTypeValidator
Questa validzione 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

isOpenEndedCall
Questa è un'applicability rule che viene valutata prima di eseguire alcune validazioni.
Se ritorna true allora vengono eseguite le validazioni altrimenti no.
Questa regola considera il campo "Bando senza scadenza" (attributo **booleanMap[openEnded]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a false viene richiesta la compilazione del campo "Data scadenza" (attributo **dateMap[endDate]**).

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

uniqueCodeValidator
Questa validazione controlla l'unicità del campo **Codice bando**.
Dal punto di vista del modello dati si tratta dell'attributo **stringMap[code]**.
Per maggiori dettagli cfr. excel modello dati dell'entità Bando di finanziamento.

checkHierarchyValidator
Questa logica controlla che la gerarchia dei Bandi sia consistente.
Deve esserci un solo bando radice. Un bando radice non può avere un padre.
Un bando foglia non può avere a sua volta figli. Un bando figlio può essere orfano se non gerarchico.

deleteChildCallFromParentValidator
Questa logica, ogni volta che viene ELIMINATA 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.inferredChid.csv** viene effettuata anche la cancellazione degli elementi elencati come associati all'unità organizzativa.

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 **ouId**
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.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.
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.inferredChid.csv
- ap.TIPOLOGIA.internalOrganizationUnit.inferredChid.ELEMENTO.jsValidator
- ap.TIPOLOGIA.[owner|contributor].role.COD_DIZIONARIO.inheritedInternalOrganizationUnit.role

```
                        if(!errors.hasErrors()){
                                WfUtil.deleteChildCallFromParentValidator(object, wfService, gaService, errors);
                        }
```

wfStartLogicContinuousTraining
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

```
                        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,
wfService);
                                                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(!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());

                                                                var myOrganizationId = Packages.java.lang.
Integer.parseInt(myOrganizationIdString);

                                                                //syso.println("myOrganizationId: "+
myOrganizationId);

                                                                var myOrganization = gaService.
getOrganizationUnit(myOrganizationId);

                                                                var externalOrganizationUnitElement = new
Packages.it.cilea.wf.model.WfItemElement();
                                                                externalOrganizationUnitElement.setDiscriminator
("externalOrganization");
                                                                externalOrganizationUnitElement.setWfItemId
(wfItem.getId());
```

```
                                                                externalOrganizationUnitElement.
getOrganizationUnitMap().put("ouId", myOrganization);
                                                                externalOrganizationUnitElement.
getWfDictionaryMap().put("roleId", coordinatorDictionary);

                                                                wfService.saveOrUpdate
(externalOrganizationUnitElement);
                                                                wfItem.getWfItemElementSet().add
(externalOrganizationUnitElement);
                                                                wfService.saveOrUpdate(wfItem);

                                                                if(department != null){

                                                                        var organizatorDictionaryId = WfUtil.
getDictionaryByCode(wfService, "ouRoleContinuousTraining.organizator");
                                                                        var organizatorDictionary = wfService.
getWfDictionary(organizatorDictionaryId);

                                                                        //syso.println
("organizatorDictionaryId: "+ organizatorDictionaryId);
                                                                        //syso.println("organizatorDictionary:
"+ organizatorDictionary);

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

                                                                        internalOrganizationUnitWfElement.
getWfDictionaryMap().put("roleId", organizatorDictionary);

                                                                        wfService.saveOrUpdate
(internalOrganizationUnitWfElement);
                                                                        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{
                                                        //syso.println("visione dipartimentale");

                                                        if(department != null){

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


                                                                internalOrganizationUnitWfElement.
getWfDictionaryMap().put("roleId", coordinatorDictionary);

                                                                wfService.saveOrUpdate
(internalOrganizationUnitWfElement);
                                                                wfService.saveOrUpdate(wfItem);
                                                        }
                                                }
                                        }
                                }

                        true;
```

isWfItemTypeGenericFCO
Questa è un'applicability rule che pilota le validazioni a cui è agganciata.
Abilita i controlli solo per la Formazione Continua generica, non quindi quella di Medicina, ad esempio.

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
Configurazioni della ValidateLogic:

- rm.orgunit.external.myOrganization

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.

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.

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 di altri partecipanti (numberOfOtherParticipants)
Numero totali di partecipanti al corso (totalNumberOfParticipants)
Per maggiori dettagli cfr. modello dati dell'entità in questione.

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.

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

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**.
- Data proroga uffciale:
  Viene usato l'attributo **endDate** se sono prensenti delle proroghe (elementi di tipo extension).
  Viene inviato null se **non** sono prensenti delle proroghe (elementi di tipo extension).
- 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.con.owner.starDate.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.con.contributor.starDate.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.enabled**.
  In caso di invio viene valuatata 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.con.administrativeOwner.starDate.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.con.customer.send2ugov.enabled**.
- 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.con.customer.send2ugov.enabled
- ap.con.cup.send2ugov.enabled
- ap.con.externalDmsIdentifier.autogenerated
- ap.con.administrativeOwner.starDate.required
- ap.con.contributor.starDate.required
- ap.TIPOLOGIA.contributorAndOwner.validator.date.end.discriminator
- ap.con.administrativeOwner.role.ugov.default
- ap.con.administrativeOwner.send2ugov.enabled

isAccountancyInfoEnabledTrue
Questa è un'applicability rule che pilota le validazioni a cui è agganciata.
Se la variabile **ap.con.accountancyInfo.enabled** è settata a true la validazione è abilitata, in tutti gli altri casi NON è abilitata.
Configurazioni della ValidateLogic:

- ap.con.accountancyInfo.enabled

isFiscalCheckEnabledTrue
Questa è un'applicability rule che pilota le validazioni a cui è agganciata.
Se la variabile **ap.con.fiscalCheck.enable** è settata a true la validazione è abilitata, in tutti gli altri casi NON è abilitata.
Configurazioni della ValidateLogic:

- ap.con.fiscalCheck.enabled

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 settata a true o non è presente la validazione è abilitata, altrimenti NON è abilitata.
Configurazioni della ValidateLogic:

- ap.con.externalDmsIdentifier.enabled

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.

```
it.cilea.wf.surplus.logic.action.enter.eqp.WfActionLogicCreateRmItem
```

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.

```
it.cilea.wf.surplus.logic.action.enter.eqp.WfActionLogicDeleteRmItem
```

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

```
it.cilea.wf.surplus.logic.action.enter.eqp.WfActionLogicManufacturerCsv
```

deleteRmItemEquipmentValidator
Questa validazione va a cancellare i dati relativi al'rmItem equipment quando viene cancellato l'oggetto padre wfItem.
Per maggiori dettagli cfr. il modello dati dell'entità in questione.

isUrlPublicCalendarRequiredEquipment
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 "Calendario pubblico" (attributo **booleanMap[publicCalendar]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesta la compilazione del campo "URL a Calendario prenotazioni" (attributo **stringMap[urlPublicCalendar]**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

isRepairabilityRequiredEquipment
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 "In uso/Funzionante" (attributo **booleanMap[inUse]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesta la compilazione del campo "Riparabilità" (attributo **wfDictionaryMap[repairability]**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

isTariffRequiredEquipment
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 sezione "Finalità dell'accesso/Applicazioni" (element **application**).
Per tutti gli oggetti che hanno questa sezione valorizzata con la voce di dizionario "Prestazioni a tariffario" viene richiesta la compilazione del campo "Specificare altro" (attributo **stringMap[tariff]**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

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.
Configurazioni della ValidateLogic:

- ap.eqp.ugovpj.inventory.enabled

inventoryGoodValidator
Questa validazione verifica che, sia presente almeno un bene di inventario recuperato da UGOV.
Se non ce n'è nemmeno uno viene chiesto all'utente conferma a procedere e se viene fornita conferma viene chiesta la motivazione per la quale non sia possibile recuperare beni di inventario da UGOV.
Configurazioni della ValidateLogic:

- ap.eqp.ugovpj.inventory.enabled

isExternalOrganizationOwnerRequiredEquipment
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 "Proprietà dell'Ateneo?" (attributo **booleanMap[internalOwnership]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesta la compilazione del campo "Ente esterno proprietario" (attributo **organizationUnitMap[externalOrganizationOwner]**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

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;
```

wfActionDeletePersonElementFromOldTutor
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 uscita dallo stato riaperto.

```
it.cilea.wf.surplus.logic.action.exit.inm.WfActionDeletePersonElementFromOldTutor
```

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
                                        gaService.saveOrUpdate(newIncomingTypeDictionary);
                                        return newIncomingTypeDictionary;
                                }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 wfItem = wfService.getWfItem(object.getId());
                        var wfItemId = wfItem.getId();
                        var wfItemIdentifier = wfItem.getIdentifier();

                        //poi vado a ricavare tutti i tutor inseriti
                        var tutorWfItemElementSet = Packages.it.cilea.core.fragment.util.FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "getWfItemElementSet", "it.cilea.wf.model.WfItemElement","
tutor", wfService);

                        //solo se ne trovo, allora eseguo il resto della procedura
                        if(!tutorWfItemElementSet.isEmpty()){

                                //ora vado a ciclare ogni wfItemElement, ricavando poi la persona collegata
                                var tutorWfItemElementSetIterator = tutorWfItemElementSet.iterator();
                                while(tutorWfItemElementSetIterator.hasNext()){

                                        var tutorWfItemElement = tutorWfItemElementSetIterator.next();

                                        var tutorPerson = tutorWfItemElement.getPersonMap().get
("contributorId");        //restituisce la person completa?
                                        tutorPerson = gaService.getPerson(tutorPerson.getId());        //nel
dubbio la ricavo dal service

                                        //ora che ho la persona, ne vado a ricavare gli element di incoming che
mi servono
                                        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
                                        //e dopo ne aggiungo uno nuovo nuovo appena creato
                                        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) ){

                                                        //lo sgancio dalla persona, lo elimino e salvo la
persona
                                                        tutorPerson.getPersonElementSet().remove
(incomingPersonElement);
                                                        gaService.deletePersonElement(incomingPersonElement.
getId());
                                                        gaService.saveOrUpdate(tutorPerson);
                                                }
                                        }

                                        //ora che mi sono accertato che per certo non ho quel personElement, me
ne vado a creare uno completamente nuovo ed aggiornato
                                        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 element 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);

                                        //controllo ulteriore nel caso in cui cambino le validazioni e non sia
più obbligatorio
                                        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);
                                }

                        }
```

deleteAllPersonElementsLinkedToMobility
Questa logica si occupa di eliminare dai profili personali del tutor/persona in ingresso (CV Scientifico) il collegamento tra le due persone.
Questa logica viene triggerata in fase di eliminazione.

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

ipTopicInheritanceValidator
Questa validazione verifica che sia inserito almeno una Area Tematica (ipTopic).
Controllo prima se ho un brevetto padre collegato
Se non ne ho, sono un prioritario e controllo se ho degli elementi "miei"
Se ne ho, sono una estensione, controllo prima gli elementi "miei" e poi quelli di mio padre
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

```
                                var wfItem = object;
                                var parentWfItemLinkSet = wfItem.getParentWfItemLinkSet
("priorityExtensionLink");

                                if(parentWfItemLinkSet == null || parentWfItemLinkSet.isEmpty()){
                                        // sono in un prioritario
                                        var ipTopicSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","ipTopic", wfService);

                                        if(ipTopicSet == null || ipTopicSet.isEmpty())
                                                errors.rejectAndLocate("error.patent.ipTopic.required",
"ipTopic");
                                }else{
                                        //sono in una estensione

                                        // controllo prima se ne ho io
                                        var ipTopicSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","ipTopic", wfService);

                                        // se non ne ho, controllo mio padre
                                        if(ipTopicSet == null || ipTopicSet.isEmpty()){

                                                var ipTopicSetInherited = WfUtil.
getWfItemElementSetFromWfItemLinkParent(wfItem, "priorityExtensionLink", "ipTopic", wfService);

                                                if(ipTopicSetInherited == null || ipTopicSetInherited.isEmpty())
                                                        errors.rejectAndLocate("error.patent.ipTopic.inherited.
required", "ipTopic");
                                        }
                                }
```

trlInheritanceValidator
Questa validazione verifica che sia inserito almeno una voce Technology Readiness Level (trl).
Controllo prima se ho un brevetto padre collegato
Se non ne ho, sono un prioritario e controllo se ho degli elementi "miei"
Se ne ho, sono una estensione, controllo prima gli elementi "miei" e poi quelli di mio padre
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

```
                                var wfItem = object;
                                var parentWfItemLinkSet = wfItem.getParentWfItemLinkSet
("priorityExtensionLink");

                                if(parentWfItemLinkSet == null || parentWfItemLinkSet.isEmpty()){
                                        // sono in un prioritario
                                        var trlSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(wfItem, "it.cilea.wf.model.WfItemElement","trl", wfService);

                                        if(trlSet == null || trlSet.isEmpty())
                                                errors.rejectAndLocate("error.patent.trl.required",
"trl");
                                }else{
                                        //sono in una estensione

                                        // controllo prima se ne ho io
                                        var trlSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(wfItem, "it.cilea.wf.model.WfItemElement","trl", wfService);

                                        // se non ne ho, controllo mio padre
                                        if(trlSet == null || trlSet.isEmpty()){

                                                var trlSetInherited = WfUtil.
getWfItemElementSetFromWfItemLinkParent(wfItem, "priorityExtensionLink", "trl", wfService);

                                                if(trlSetInherited == null || trlSetInherited.isEmpty())
                                                        errors.rejectAndLocate("error.patent.trl.inherited.
required", "trl");
                                        }
                                }
```

applicationInheritanceValidator
Questa validazione verifica che sia inserito almeno una Domanda di Deposito (application).
Controllo prima se ho un brevetto padre collegato
Se non ne ho, sono un prioritario e controllo se ho degli elementi "miei"
Se ne ho, sono una estensione, controllo prima gli elementi "miei" e poi quelli di mio padre
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

```
                                var wfItem = object;
                                var parentWfItemLinkSet = wfItem.getParentWfItemLinkSet
("priorityExtensionLink");

                                if(parentWfItemLinkSet == null || parentWfItemLinkSet.isEmpty()){
                                        // sono in un prioritario
                                        var applicationSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","application",
wfService);

                                        if(applicationSet == null || applicationSet.isEmpty())
                                                errors.rejectAndLocate("error.patent.application.required",
"application");
                                }else{
                                        //sono in una estensione

                                        // controllo prima se ne ho io
                                        var applicationSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement","application",
wfService);

                                        // se non ne ho, controllo mio padre
                                        if(applicationSet == null || applicationSet.isEmpty()){

                                                var applicationSetInherited = WfUtil.
getWfItemElementSetFromWfItemLinkParent(wfItem, "priorityExtensionLink", "application", wfService);

                                                if(applicationSetInherited == null || applicationSetInherited.
isEmpty())
                                                        errors.rejectAndLocate("error.patent.application.
inherited.required", "application");
                                        }
                                }
```

cupInheritanceValidator
Questa validazione verifica che sia inserito il CUP (cup).
Controllo prima se ho un brevetto padre collegato
Se non ne ho, sono un prioritario e controllo se ho degli elementi "miei"
Se ne ho, sono una estensione, controllo prima gli elementi "miei" e poi quelli di mio padre
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

```
var wfItem = object;
var parentWfItemLinkSet = wfItem.getParentWfItemLinkSet
("priorityExtensionLink");

if(parentWfItemLinkSet == null || parentWfItemLinkSet.isEmpty()){
        // sono in un prioritario

        var cup = wfItem.getStringMap().get("cup");

        if(cup == null)
                errors.rejectValue("stringMap[cup]","error.patent.cup.
required");
}else{
        //sono in una estensione

        // controllo prima se ne ho io
        var cup = wfItem.getStringMap().get("cup");

        // se non ne ho, controllo mio padre
        if(cup == null){

                var iterator = parentWfItemLinkSet.iterator();

                var priorityExtensionLink = iterator.next();

                var priorityPatent = wfService.getWfItem(priorityExtensionLink.
getParentId());

                cup = priorityPatent.getStringMap().get("cup");

                if(cup == null)
                        errors.rejectValue("stringMap[cup]","error.patent.cup.
inherited.required");
        }
}
```

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**.

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

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

```
```

```
                        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);

                        // lo setto poi come da NON verificare
                        wfItem.getBooleanMap().put("verifiedAfterPartialImportFromEPO", null);

                        var priorityPatentId = wfItem.getIntegerMap().get("priorityPatentId");

                        if(priorityPatentId != null){

                                if(wfItem.getBooleanMap().get("recoveryMetadataFromEPO") == null || !wfItem.
getBooleanMap().get("recoveryMetadataFromEPO")){

                                        // ------ per andare a recuperare gli altri owner
                                        var linkedPriorityPatent = wfService.getWfItem(priorityPatentId);

                                        var priorityPatentOwnerSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(linkedPriorityPatent, "getWfItemElementSet", "it.cilea.wf.model.
WfItemElement", "owner", wfService);
                                        var priorityPatentOwnerSetIterator = priorityPatentOwnerSet.iterator();

                                        while (priorityPatentOwnerSetIterator.hasNext()){

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

                                        // --------- per creare il nuovo link

                                        var priorityExtensionLink = new Packages.it.cilea.wf.model.WfItemLink();
                                        priorityExtensionLink.setParentId(priorityPatentId);
                                        priorityExtensionLink.setChildId(wfItem.getId());
                                        priorityExtensionLink.setDiscriminator("priorityExtensionLink");


                                        var parentWfItemLinkSet = wfItem.getParentWfItemLinkSet();
                                        parentWfItemLinkSet.add(priorityExtensionLink);
                                        wfItem.setParentWfItemLinkSet(parentWfItemLinkSet);

                                        wfService.saveOrUpdate(priorityExtensionLink);

                                        // wfItem.getIntegerMap().put("linkedInventionId", null);
                                        // wfItem.getBooleanMap().put("isPriority", false);
```

```
                        }
                }

                wfService.saveOrUpdate(wfItem);
```

priorityExtensionLinkDeleteValidator
Questa è una validazione che viene chiamata nel flusso dei Brevetti quando si prova ad eliminare un Brevetto prioritario.
Essa impedisce che si possa eliminare l'oggetto se è collegato ad uno o più Brevetti figli.
In caso la logica trovasse questi collegamenti, verrà presentato un messaggio di errore che avviserà dei link presenti e inviterà ad eliminarli per procedere.
Il messaggio di errore viene ricavato dall'etichetta con chiave **error.patent.priorityExtensionLink.toDelete**
Per maggiori dettagli cfr. modello dati dell'entità specifica.

```
                var wfItemFull = wfService.getWfItem(wfItem.getWfItemId());

                var priorityExtensionLinkCounter = 0;

                var childWfItemLinkSet = wfItemFull.getChildWfItemLinkSet();
                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('priorityExtensionLink') &&
                                !childLink.getDiscriminator().equals('version') &&
                                !childLink.getDiscriminator().contains('edit-post') &&
                                !childLink.getDiscriminator().contains('new-post') &&
                                !childLink.getDiscriminator().contains('deleted') &&
                                childWfItem!=null && childWfItem.getWithdrawn() == false){

                                priorityExtensionLinkCounter =
priorityExtensionLinkCounter+1;
                        }
                }

                if(priorityExtensionLinkCounter > 0)
                        errors.reject("error.patent.priorityExtensionLink.toDelete");
```

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",
wfService);
                                        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");
                        //se è null, c'è già un validator che lo controlla
                        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() != object.getId() && wfItemObtained.
getWithdrawn() == false) {
                                                errors.rejectValue("stringMap[applicationNumber]", "error.
patent.nonUniqueApplicationNumber");
                                                break;
                                        }
                                }
                        }
```

bookEntryTotalPatentValidator
Questa validazione verifica la somma dei Costi inseriti.
In particolare verifica che il valore dell'attributo **numberMap[bookEntryTotal]** sia uguale alla somma degli attributi **numberMap[amount]** presenti nella sezione "Costi".
Dal punto di vista del modello dati si tratta degli attributi **numberMap[bookEntryTotal]**, **numberMap[amount]** e l'elemento **bookEntry**.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

lastLegalStatusCalculatorPatent
Questa validazione si occupa, ad ogni salvataggio, di a cercare l'ultimo Status Legale valido basandosi sulla Data di Registrazione e salvarne la voce in un metadato nascosto.
In particolare va a cercare tra gli elementi con discriminator **legalStatus** e ne va a valutare la data all'interno di **dateMap[registrationStatus]**.
La voce di dizionario presente in **wfDictionaryMap[status]** viene poi salvata all'interno di **wfItem.wfDictionaryMap[lastLegalStatus]**.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

startPatentValidator
Questa validazione viene eseguita in fase di creazione di un Brevetto e verifica:

- Verifica che sia compilato il campo "Titolo" SOLO se l'utente ha deciso di NON recuperare i metadati da EPO
- Verifica che sia compilato il campo "Data di deposito" SOLO se l'utente ha deciso di NON recuperare i metadati da EPO.
- Verifica che sia compilato il campo "PI Prioritaria" SOLO se l'utente ha deciso di NON recuperare i metadati da EPO e che il corrente Brevetto NON è prioritario

Dal punto di vista del modello dati si tratta degli attributi **booleanMap[recoveryMetadataFromEPO]**, **description**, **dateMap[applicationDate]**, **booleanMap[isPriority]** e **integerMap[priorityPatentId]**.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```
                        var recoveryMetadataFromEPO = object.getBooleanMap().get("recoveryMetadataFromEPO");
                        var description = object.description;
                        var applicationDate = object.getDateMap().get("applicationDate");
                        var isPriority = object.getBooleanMap().get("isPriority");
                        var priorityPatentId = object.getIntegerMap().get("priorityPatentId");

                        if((recoveryMetadataFromEPO == null || !recoveryMetadataFromEPO) && description == null)
{
                                errors.rejectValue("description", "error.notNull.fieldLabelKey", [messageUtil.
findMessage("label.patent.description")], null);
                        }

                        if((recoveryMetadataFromEPO == null || !recoveryMetadataFromEPO) && applicationDate ==
null){
                                errors.rejectValue("dateMap[applicationDate]", "error.notNull.fieldLabelKey",
[messageUtil.findMessage("label.patent.applicationDate")], null);
                        }

                        if ((recoveryMetadataFromEPO == null || !recoveryMetadataFromEPO) && !isPriority &&
priorityPatentId == null){
                            errors.rejectValue("integerMap[priorityPatentId]", "error.notNull.fieldLabelKey",
[messageUtil.findMessage("label.patent.priorityPatentId")], null);
                        }
```

warningAcknowlegementValidator
Questa validazione viene effettuata durante il salvataggio di un Brevetto mentre è nello stato Bozza.
Essa controlla se al Brevetto Estensione corrente si sta cancellando il link verso il Prioritario.
Nel caso l'utente lo stia eliminando, chiede di confermare la scelta.
Se l'utente non conferma, la validazione continua a bloccarlo fino a quando non inserisce un collegamento ad un Prioritario.
Se l'utente conferma, a quel punto il Brevetto Estensione corrente viene trasformato in Prioritario.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```
                        var current_priorityExtensionLink = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink",
"priorityExtensionLink", wfService);
                        var saved_priorityExtensionLink = object.getParentWfItemLinkSet
("priorityExtensionLink");
                        var warningAcknowlegement = object.getBooleanMap().get("warningAcknowlegement");

                        // se non ho link al momento ma ne ho salvati, vuol dire che ne sto eliminando
                        if(current_priorityExtensionLink != null && saved_priorityExtensionLink != null &&
                                current_priorityExtensionLink.isEmpty() && !saved_priorityExtensionLink.
isEmpty()){

                                if(warningAcknowlegement == null)
                                        errors.rejectValue("booleanMap[warningAcknowlegement]","error.patent.
warningAcknowlegement.null.required");

                                if(warningAcknowlegement == false)
                                        errors.rejectValue("booleanMap[warningAcknowlegement]","error.patent.
warningAcknowlegement.false.required");

                                if(warningAcknowlegement == true){
                                        object.getBooleanMap().put("isPriority", true);
                                        object.getIntegerMap().put("priorityPatentId", object.getId());
                                        object.getBooleanMap().put("warningAcknowlegement", null);
                                }
                        }else {
                                object.getBooleanMap().put("warningAcknowlegement", null);
                        }
```

priorityExtensionLinkValidator
Questa validazione viene effettuata in entrata agli stati definitivi del flusso dei Brevetti.
Essa controlla se al Brevetto Estensione corrente sia presente un collegamento verso il Prioritario, in caso contrario segnalerà errore.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

```
                        var current_priorityExtensionLink = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(object, "getParentWfItemLinkSet", "it.cilea.wf.model.WfItemLink",
"priorityExtensionLink", wfService);

                        if(current_priorityExtensionLink != null &&        current_priorityExtensionLink.
isEmpty()){
                                errors.rejectAndLocate("priorityExtensionLink","error.patent.
priorityExtensionLink.required");
                        }
```

isNotPriority
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 se un Brevetto è o meno un brevetto prioritario e attiva la validazione solo per quelli che non lo sono, cioe le estensioni
Per tutti gli oggetti che hanno questo attributo valorizzato a true vengono disattivate le validazioni.

```
                        (
                                object.getBooleanMap().get("isPriority")!=null && object.getBooleanMap().get
("isPriority") == false
                        )
```

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 lastYear = positionlastDate.substring(6, 10);
                                        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();
                                                //Java17 test Elisa - OK funziona
                                                if (maxPriority==null || maxPriority == 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

```
                        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.
TeamUserLinkSearchCommand();
                                        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.

```


                        var gaUserDetail = Packages.it.cilea.ga.authorization.context.GaAuthorizationUserHolder.
getUser();
                        if (!"ROLE_ADMIN".equals(gaUserDetail.getCurrentAuthorityIdentifier())){
                                var state = wfItem.getWfState().getDescription();
                                var configurationKeyStart = "DEPARTMENT_FUND_STATE_START_DATE_" + state.
toUpperCase();
                                var configurationKeyEnd = "DEPARTMENT_FUND_STATE_END_DATE_" + state.
toUpperCase();
                                var stateStartDate = Packages.it.cilea.core.configuration.util.
ConfigurationUtil.getConfigValue(configurationKeyStart);
                                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");
                                }
                        }
```

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 settata a false la validazione NON è abilitata in tutti gli altri casi è abilitata.
Configurazioni della ValidateLogic:

- ap.prj.externalDmsIdentifier.enabled

isCupPopulated
Questa è un'applicability rule che ritorna true se l'attributo cup è valorizzato

isCigPopulated
Questa è un'applicability rule che ritorna true se l'attributo cig è valorizzato

isCigNotPopulated
Questa è un'applicability rule che ritorna true se l'attributo cig NON è valorizzato

isCupNotPopulated
Questa è un'applicability rule che ritorna true se l'attributo cup NON è valorizzato

grantorUniquenessValidator
Questa validazione controlla:

- in fase di aggiunta di un Ente finanziatore che NON sia già censito lo stesso Ente finanziatore con lo stesso Numero contratto PER TUTTI I PROGETTI
- in fase di salvataggio/spostamento del Progetto che NON siano presenti più di una volta gli stessi Enti finanziatori con gli stessi Numeri contratto PER TUTTI I PROGETTI

Dal punto di vista del modello dati si tratta degli elementi di tipo **grantor** e degli attributi **organizationUnitMap[grantorId]**, **stringMap[grantNumber]**.
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto.

wfUgovPjSenderValidatorProject
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.
Per maggiori dettagli su queste regole consultare il team di UGOV PJ.
Le controparti dei progetti 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 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**. Se **stringMap[grantNumber]** è nullo allora viene usato solo l'attributo **description** (senza nemmeno il trattino), se **description** è nullo allora viene usato solo l'attributo **stringMap [grantNumber]** (senza nemmeno il trattino).
  Se NON valorizzata a true allora viene usato l'attributo **description**.
  In ogni caso la stringa verrà troncata a 255 caratteri per limitazioni lato UGOV PJ (se il numero di caratteri sarà maggiore).
- Importo progetto
  Viene valutata la variabile di configurazione **ap.prj.financing.internalCost.send2ugov.forced**.
  Se valorizzata a true allora viene usato l'attributo **internalCost**
  Se NON valorizzata a true allora
  - Se l'ateneo è capofila allora viene inviato il risultato della seguente formula **globalContribution+internalCofinancing+manHoursCofin ancing**. E' possibile disabilitare questa verifica su una determinata tipologia di progetti (itemType), e di conseguenza non verrà inviato il risultato della seguente formula **globalContribution+internalCofinancing+manHoursCofinancing** (per la Ricerca non competitiva), settando a false la variabile di configurazione **ap.PARAMETRIC_TYPE_LOWER_CASE.partner.main.required**.
    Esempio variabile: **ap.prj.research.int.partner.main.required**. In tutti gli altri casi il controllo verrà fatto.
  - Se l'ateneo NON è capofila allora viene inviato **internalCost**

  L'invio viene effettuato se l'importo è non nullo e maggiore di zero.
- Data inizio validità:
  Viene usato l'attributo **startDate**.
- Data fine validità:
  Viene usato l'attributo **endDate**.
- Data inizio validità del progetto dal punto di vista contabile:
  Viene usato l'attributo **expenditureStartDate**.
- Data fine validità del progetto dal punto di vista contabile:
  Viene usato l'attributo **expenditureEndDate**.
- Data proroga uffciale:
  Viene usato l'attributo **endDate** se sono prensenti delle proroghe (elementi di tipo extension).
  Viene inviato null se **non** sono prensenti delle proroghe (elementi di tipo extension).
- CUP: **cup**
  Inviato se valorizzata a true la variabile di configurazione **ap.prj.cup.send2ugov.enabled**.
  Se l'invio del CUP è abilitato e UGOV PJ restituisce l'errore **Attenzione: CUP XXXXXXX non trovato** vuol dire che il CUP che IRIS AP sta tentando di inviare a UGOV PJ non è stato trovato su UGOV CO.
  E' necessario quindi inserire quel CUP in UGOV CO prima di potere effettuare invio, oppure è possibile disatttivare l'invio settando ap.prj.cup. send2ugov.enabled al valore false
- Riferimento Data Management System (Sistema Documentale): **externalDmsIdentifier**
  Inviato l'identifier del progetto se valorizzata a true la variabile di configurazione **ap.prj.externalDmsIdentifier.autogenerated**, 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.prj.owner.starDate.required**.
  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 configuration ap.prj.owner.role.enabled con default false)
  E' buona cosa inserire prima nel flusso, in caso di attivazione dei ruoli, la validazione ownerRoleSend2UgovValidatorProject, a cui è delegata la correttezza dei dati.
- Partecipanti: **contributor**
  Inviate anche date di inizio e fine se valorizzata a true la variabile di configurazione **ap.prj.contributor.starDate.required**.
  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.prj.administrativeOwner.send2ugov.enabled**.
  In caso di invio viene valuatata anche la configurazione **ap.prj.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.prj.administrativeOwner.starDate.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**
- Finanziatori: **grantor**
  Inviato se valorizzata a true la variabile di configurazione **ap.prj.grantor.send2ugov.enabled**.
  Se valorizzata a true la configurazione **ap.prj.grantor.financingQuota.send2ugov.enabled** le quote di finanziamento vengono inviate a PJ.
- Partner: **partner**
  Inviato se valorizzata a true la variabile di configurazione **ap.prj.partner.send2ugov.enabled**.
- Tipo progetto: come da mappatura tassonomia ap-item-type.xls
- 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.prj.grantor.send2ugov.enabled
- ap.prj.partner.send2ugov.enabled
- ap.TIPOLOGIA.FRAGMENT.role.main
- rm.orgunit.external.myOrganization
- ap.prj.financing.internalCost.send2ugov.forced
- ap.prj.descriptionAndGrantNumberConcatenationEnabled
- ap.prj.cup.send2ugov.enabled
- ap.prj.externalDmsIdentifier.autogenerated
- ap.prj.contributor.starDate.required
- ap.prj.administrativeOwner.send2ugov.enabled
- ap.prj.owner.starDate.required
- ap.prj.grantor.financingQuota.send2ugov.enabled

ethicCommiteeDeleteInconsistentDataProject
Questa validazione controlla che se non sono previste attività di ricerca che prevedono sperimentazioni passibili di giudizio da parte del Comitato Etico e 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.

ownerRoleSend2UgovValidatorProject
Questa validazione opera sugli elementi **owner** (responsabile scientifico) figli dell'oggetto radice.
Viene verificato che esista almeno un responsabile scientifico da inviare ad UGOV PJ, e che il loro numero non superi un valore massimo configurabile.
Il parametro per il numero massimo è **ap.prj.owner.send2ugov.number**, che se valorizzato deve essere un numero maggiore di 0: se non valorizzato non c'è limite.
Per i Responsabili scientifici, nella scheda, può essere visibile il Ruolo (visibilità pilotata da configuration **ap.prj.owner.role.enabled** con default false).
In tal caso è possibile limitare il conteggio ad una lista di ruoli da sincronizzare con PJ, mediante il parametro **ap.prj.owner.send2ugov.roles** (formato: comma separated dei codici di dizionario).
Configurazioni della ValidateLogic:

- ap.prj.owner.role.enabled
- ap.prj.owner.send2ugov.roles
- ap.prj.owner.send2ugov.number

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".
In base alla casistica è necessario customizzare l'etichetta **label.prj.internalOrganizationUnitCost.warning** con i valori indicati di seguito, il contenuto di questa etichetta viene visualizzato nel tooltip del punto interrogativo di finaco al nome della sezione "Strutture Interne" presente nel tab "Finanziamento"

requireOneForEveryInternalOrganizationUnit	notMandatoryIfOnlyOneInternalOrganizationUnit	Controllo	Testo etichetta
true	**true**	• Se viene inserito solo un elemento internalOrganization Unit allora non è necessario inserire l'elemento internalOrganization UnitCost associato perché si presume che tutto il finanziamento vada all'unica struttura • Se vengono inseriti più di un elemento internalOrganization Unit allora è necessario inserire l'elemento internalOrganization UnitCost associato per ogni internalOrganization Unit	Se viene inserita più di una struttura nella sezione "Strutture interne" del TAB "Soggetti interni" è necessario che ognuna di queste sia riportata in questa sezione, viceversa se viene inserita solo una struttura nella sezione "Strutture interne" del TAB "Soggetti interni" NON è necessario compilare questa sezione, in quanto si presume che tutto il finanziamento vada all'unica struttura.
false/non presente	**false/non presente**	• È necessario che sia presente almeno un elemento internalOrganization UnitCost	NON NECESSARIO (viene utilizzata l'etichetta di default)
true	**false/non presente**	• È necessario inserire l'elemento internalOrganization UnitCost associato per ogni internalOrganization Unit	E' necessario che ognuna delle strutture inserite nella sezione "Strutture interne" del TAB "Soggetti interni" sia riportata in questa sezione.
false/non presente	**true**	• Se viene inserito solo un elemento internalOrganization Unit allora non è necessario inserire l'elemento internalOrganization UnitCost associato perché si presume che tutto il finanziamento vada all'unica struttura • Se vengono inseriti più di un elemento internalOrganization Unit allora è necessario inserire almeno un elemento internalOrganization UnitCost	Se viene inserita più di una struttura nella sezione "Strutture interne" del TAB "Soggetti interni" è necessario che una di queste sia riportata in questa sezione, viceversa se viene inserita solo una struttura nella sezione "Strutture interne" del TAB "Soggetti interni" NON è necessario compilare questa sezione, in quanto si presume che tutto il finanziamento vada all'unica struttura.

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

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.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeTopic.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeSubmitted.enabled

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

- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeAuthorizationNumber.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeAuthorizationDate.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeAuthorizationOrganization.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 **ethicCommiteeTopic**.
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto
Configurazioni della ValidateLogic:

- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeTopic.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeSubmitted.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeAuthorizationNumber.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeAuthorizationDate.enabled
- ap.prj.validation.ethicCommiteeValidatorProject.ethicCommiteeAuthorizationOrganization.enabled

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.

isInternalOrganizationUnitRequired
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 **Struttura capofila diversa da afferenza del responsabile?** (**booleanMap[customizedInternalOrganization]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesto l'inserimento della Struttura (**organizationUnitMap [internalOrganizationUnit]**).

isOwnerRoleEnabledProject
Questa è un'applicability rule che pilota le validazioni relative ai referenti interni del Progetto (owner).
La validazione viene applicata solo se la variabile **ap.prj.owner.role.enabled** è settata a true.
Configurazioni della ValidateLogic:

- ap.prj.owner.role.enabled

isOwnerDepartmentEnabledProject
Questa è un'applicability rule che pilota le validazioni relative ai referenti interni del Progetto (owner).
La validazione viene applicata solo se la variabile **ap.prj.owner.department.enabled** è settata a true.
Configurazioni della ValidateLogic:

- ap.prj.owner.department.enabled

isExpectedEvaluationProject
Questa è un'applicability rule che viene valutata prima di eseguire alcune validazioni.
Se ritorna true allora vengono eseguite le validazioni altrimenti no.
Questa regola considera il campo "Il Progetto prevede una valutazione?" (attributo **booleanMap[expectedEvaluation]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesta la compilazione del campo multiplo "Valutazioni" (elemento **evaluation**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

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

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 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 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
ap.prj.hidden
Se visibili nella specifica dell'ente finanziatore, 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

grantorApprovalDateRequiredValidatorProject
DEPRECATO - Questa validazione verifica che vengano inserite le date di approvazione e il numero di contratto associate a ogni Ente Finanziatore.
Dal punto di vista del modello dati si tratta degli elementi di tipo **grantor**.
Per maggiori dettagli cfr. excel modello dati dell'entità Progetto

```
                var grantorSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator(object,
"getWfItemElementSet", "it.cilea.wf.model.WfItemElement", "grantor", wfService);
                var grantorSetIterator = grantorSet.iterator();
                while (grantorSetIterator.hasNext()) {
                    var element = grantorSetIterator.next();
                    if(element.getDateMap().get("approvalDate") == null) {
                            errors.rejectAndLocate("error.project.grantor.approvalDate.required", "grantor");
                    }
                    if(element.getStringMap().get("grantNumber") == null) {
                            errors.rejectAndLocate("error.project.grantor.grantNumber.required", "grantor");
                    }
                }
```

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

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 rilassarne 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 wfUgov PjSenderValidatorProject

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.TIPOLOGIA.partner.uniquePartnerAndPartnerRoleCheck.enabled
- ap.prj.partner.role.uniquePartner.enabled
- ap.prj.partner.role.uniquePartner.code

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.prj.financing.requested.enabled è true (default true) e la configurazione **ap.prj.financing.internalCost.send2ugov.forced** è false (default false):

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

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

```
                        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
Questa validazione viene abilitata SOLO se la configurazione **ap.prj.financing.internalCost.send2ugov.forced** è settata a false o NON è presente.
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) - globalContributiontOriginalCurrency

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.prj.financing.internalCost.send2ugov.forced

checkInconsistentFundingValidatorProject
Questa validazione controlla la coerenza delle informazioni di finanziamento assegnato sia per le **internalOganizationUnit** (unità organizzative interne) che per i **partner** (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)

- ap.prj.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.prj.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.prj.financing.requested.enabled
  Se true significa che è stato abilitato anche la gestione del finanziamento richiesto globale
- ap.prj.financing.internalOrganizationUnitCost.requested.enabled
  Se true significa che è stato abilitato anche la gestione del finanziamento richiesto per singola struttura di Ateneo
- ap.prj.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.prj.financing.internalOrganizationUnitCost.requested.enabled e ap.prj.financing.requested.enabled sono a true
- ap.prj.financing.partnerCost.requested.enabled
  Se true significa che è stato abilitato anche la gestione del finanziamento richiesto per singolo partner
- ap.prj.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.prj.financing.partnerCost.requested.enabled e ap.prj.financing.requested.enabled sono a true

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

- ap.prj.internalOrganizationUnitCost.notMandatoryIfOnlyOneInternalOrganizationUnit.enabled
  Se
  1. la configurazione è valorizzata a true
  2. esiste **una sola struttura** censita (internalOrganizationUnit)
  3. **non ci sono elementi di suddivisione** del finanziamento (internalOrganizationUnitCost)

  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
  1. la configurazione è valorizzata a true
  2. esiste **un solo partner** censita (partner)
  3. **non ci sono elementi di suddivisione** del finanziamento (partnerCost)

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

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.hidden** e **ap.prj.grantor.financingQuota.validation.enabled**.
Nel caso in cui **non sia nascosta** la Quota per ogni Ente e **non sia presente o sia settata a true**, viene controllato che la somma delle quote associate agli Enti Finanziatori sia uguale al Contributo Totale Ateneo.
IN TUTTI GLI ALTRI CASI questo controllo NON viene effettuato.
Configurazioni della ValidateLogic:

- ap.prj.hidden
- ap.prj.grantor.financingQuota.validation.enabled

internalOrganizationUnitCostEditValidator
Questa validazione consente l'inserimento delle sole strutture censite nella sezione "Soggetti interni"

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

- ap.prj.financing.requested.enabled
- ap.prj.financing.internalOrganizationUnitCost.requested.enabled

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.

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.

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.

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 generateSequence = stringBuilder.toString();
                                var sequence = org.springframework.util.DigestUtils.md5DigestAsHex
(generateSequence.getBytes());
                        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 isPresnet = 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)
                            isPresnet = false;
                        return isPresnet;
                        }

                        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 isPresent = false;
                                       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 poteziale 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

wfStartLogicProjectTypeFromCall
Questa 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 (sempra 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

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

                if (creationByCallEnabled=="true" && isCreationByCall){
                        var callId=wfItem.getIntegerMap().get("callId");

                        if (callId!=null) {
```

```
                                var call=wfService.getWfItem(callId);
                                var projectTypeIdentifier=call.getStringMap().get("linkedProjectType");

                                var wfItemTypeList=wfService.getWfItemTypeByIdentifier(projectTypeIdentifier,
0);
                                if (CollectionUtils.isNotEmpty(wfItemTypeList)){
                                        wfItem.setWfItemTypeId(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;
```

wfStartLogicPnrrProject
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 (Sì/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

```
                        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);
                                                }
                                        }
```

```
                        }
                }
```

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

childCallValidatorProject
Questa validazione verifica che i bandi agganciati siano marcati come figli.
Dal punto di vista del modello dati si tratta degli elementi di tipo **callProjectLink**

isAlphaNumeric
Verifica che il campo isScoraNumeric sia a TRUE

```
                        if(object.getBooleanMap().get('isScoreNumeric')!=null)
                                !object.getBooleanMap().get('isScoreNumeric');
                        else
                                false;
```

wfStartLogicCopyFromPublicEngagement
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
Effettua la clonazione da una iniziativa esistente specificata nel form di start.
Per eseguire questa logica deve essere messa a true la configurazione ap.pen.create-by-cloning.enabled.
Nel caso in cui ci siano più flussi attivi per la stessa entità (qualora sia attiva la declinazione) tutti devono avere le stesse configurazioni di flavours e declinazioni, altrimenti non si garantisce il corretto funzionamento.
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement.
Configurazioni della StartLogic:

- ap.TIPOLOGIA.create-by-cloning.enabled

```
                var enabledConf = ConfigurationUtil.getConfigValue("ap.pen.create-by-cloning.enabled");
                var enabledCopy = wfItem.getBooleanMap().get("copyFrom");
                if (StringUtils.equalsIgnoreCase(enabledConf, "true") && enabledCopy){
                        var wfItemToClone = wfService.getWfItem(wfItem.getIntegerMap().get
("wfItemIdToClone"));
                        wfItem = WfUtil.clonePrimitiveData(wfItemToClone, wfItem);
                        wfItem = WfUtil.cloneElementData(wfItemToClone, wfItem, genericService);
                        wfItem.setIdentifier(null);
                        wfItem.getClobMap().put("transitionsLog", null);
                        wfItem.getIntegerMap().put("wfItemTypeIdPreviousValue", null);
                        wfItem.getStringMap().put("previousState", null);
                        wfItem.getIntegerMap().put("wfItemIdToClone", null);
                        var state = wfService.getWfState("draft", wfItemToClone.getWfState().
getWfDefinitionId());
                        wfItem.setWfStateId(state.getId());
                        wfItem.setWfState(state);
                        var wfItemId = wfService.saveOrUpdate(wfItem);

                        var attachmentSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(wfItem, "it.cilea.wf.model.WfItemElement", "attachment", wfService);
                        var attachmentSetIterator = attachmentSet.iterator();
                        while(attachmentSetIterator.hasNext()){
                                var element = attachmentSetIterator.next();
                                wfItem.getWfItemElementSet().remove(element);
                                wfService.deleteWfItemElement(element.getId());
                        }

                        var user = Packages.it.cilea.ga.authorization.context.GaAuthorizationUserHolder.
getUser();
```

```
                                var person = gaService.getPerson(user.getPersonId());
                                if (user.getCurrentAuthorityIdentifier().equals("ROLE_USER")) {
                                        var contributorSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "contributor",
wfService);
                                        var contributorSetIterator = contributorSet.iterator();
                                        var foundInContributor = false;
                                        while(contributorSetIterator.hasNext()){
                                                var element = contributorSetIterator.next();
                                                if(element.getPersonMap().get("contributorId").getId() == user.
getPersonId()) {
                                                        foundInContributor = true;
                                                        break;
                                                }
                                        }
                                        var ownerSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "owner",
wfService);
                                        var ownerSetIterator = ownerSet.iterator();
                                        var foundInOwner = false;
                                        while(ownerSetIterator.hasNext()){
                                                var element = ownerSetIterator.next();
                                                if(element.getPersonMap().get("ownerId").getId() == user.
getPersonId()) {
                                                        foundInOwner = true;
                                                        break;
                                                }
                                        }

                                        //se la persona che sta facendo il clone è SOLO nei contributor, allora
rimuovo tutti i contributor, owner e orgUnit
                                        //successivamente aggiungo la persona tra gli owner e recupero la sua
struttura di afferenza (questo compito rimane alla startLogic wfStartLogicMultipleOwners)
                                        if(foundInContributor && !foundInOwner) {
                                                var ownerSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "owner",
wfService);
                                                var ownerSetIterator = ownerSet.iterator();
                                                while(ownerSetIterator.hasNext()){
                                                        var element = ownerSetIterator.next();
                                                        wfItem.getWfItemElementSet().remove(element);
                                                        wfService.deleteWfItemElement(element.getId());
                                                }
                                                var contributorSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement", "contributor",
wfService);
                                                var contributorSetIterator = contributorSet.iterator();

                                                while(contributorSetIterator.hasNext()){
                                                        var element = contributorSetIterator.next();
                                                        wfItem.getWfItemElementSet().remove(element);
                                                        wfService.deleteWfItemElement(element.getId());
                                                }
                                                var internalOrganizationUnitSet = FragmentUtil.
getCurrentFragmentSetByParentAndDiscriminator(wfItem, "it.cilea.wf.model.WfItemElement",
"internalOrganizationUnit", wfService);
                                                var internalOrganizationUnitIterator =
internalOrganizationUnitSet.iterator();
                                                while(internalOrganizationUnitIterator.hasNext()){
                                                        var element = internalOrganizationUnitIterator.next();
                                                        wfItem.getWfItemElementSet().remove(element);
                                                        wfService.deleteWfItemElement(element.getId());
                                                }
                                                if (object.getPersonMap().get("owner") == null){
                                                        wfItem.getPersonMap().put("owner",
person);
                                                }
                                        }
                                }

                                var publicEngagementLink = new Packages.it.cilea.wf.model.WfItemLink();
```

```
                        publicEngagementLink.setDiscriminator("publicEngagementLink");
                        publicEngagementLink.setParentId(wfItemToClone.getId());
                        publicEngagementLink.setChildId(wfItemId);
                        wfService.saveOrUpdate(publicEngagementLink);
                        wfItem.getParentLinkSet().add(publicEngagementLink);
                }
                true;
```

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

- evaluationEnable (false)
- externalFinancing (0)
- totalBudget (0)

Per maggiori dettagli cfr. modello dati dell'entità publicEngagement

```
                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.getBooleanMap().get("personOrDepartment")!=null){
                        wfItem.getBooleanMap().put("personOrDepartment",null);
                }
                if(wfItem.getBooleanMap().get("copyFrom")!=null){
                        wfItem.getBooleanMap().put("copyFrom", null);
                }
                if(wfItem.getIntegerMap().get("customizedYear")==null){
                        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.getIntegerMap().put("customizedYear", new Packages.java.lang.Integer
(calendar.get(Calendar.YEAR)));
                }
                true;
```

wfStartLogicAcademicAreaFromOwner
Questa logica viene eseguita in fase di creazione di un nuovo oggetto (solo se non sono già presenti metadati relativi all'Area Accademica).
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

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 **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

E' 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(?)

                        var academicElementSet = wfItem.getWfItemElementSet("scientificAreasInvolved");

                        if(academicElementSet.size() == 0){
                                //devo usare per forza il metodo lungo perchè 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 personPositionSet = 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.setDate(startDate.getTime());
                                                        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");
                                                }
                                        }
                                }
                        }
```

isPersonRequiredPublicEngagement
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 "Responsabile o dipartimento?" (attributo **booleanMap[personOrDepartment]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a true viene richiesto l'inserimento di un Responsabile scientifico in fase di creazione.
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

isDepartmentRequiredPublicEngagement
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 "Responsabile o dipartimento?" (attributo **booleanMap[personOrDepartment]**).
Per tutti gli oggetti che hanno questo attributo valorizzato a false viene richiesto l'inserimento di un Dipartimento in fase di creazione.
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

isSpecificOtherValueRequiredPublicEngagement
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 "Pubblico Coinvolto" della sezione Pubblici Coinvolti (attributo **wfDictionaryMap[dictionary]**).
Per tutti gli oggetti che hanno questo attributo valorizzato con la voce di dizionario "Altro" viene richiesta la compilazione del campo "Specificare altro" (attributo **stringMap[specificOtherValue]**).
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

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");
```

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 modello dati dell'entità in questione.

ownerTimeReportingValidator
Questa validazione verifica che ogni responsabile (owner) abbia valorizzato il campo Giornate/uomo o Ore/uomo.
Vengono valutati gli attributi **element.integerMap[manDay]**, **element.integerMap[manHour]**.
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 messggio 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

publicInvolvedValidatorPublicEngagement
Questa validazione valuta la configurazione **ap.pen.publicInvolved.required.onlyTraining-communication.enabled**.
Se la configurazione **è presente e valorizzata a true** allora attiva il controllo SOLO se la tipologia dell'iniziativa è "PEN.FORMAZIONE-COMUNICAZIONE" ("Giornate organizzate di formazione alla comunicazione - rivolta a PTA o docenti"): viene controllato che esista almeno un elemento di tipo **publicInvolved** e viene controllato che vengano aggiunti solo i seguenti Pubblici coinvolti (code dei dizionari):

- publicInvolvedPublicEngagement.other-professors
- publicInvolvedPublicEngagement.other-administratives
- publicInvolvedPublicEngagement.other-unstructored-research-staff
- publicInvolvedPublicEngagement.other-students-phd

Se invece la configurazione è assente o valorizzata a false, allora per qualsiasi tipologia di iniziativa verrà controllato SOLO che sia presente almeno un elemento di tipo **publicInvolved**.
Dal punto di vista del modello dati si tratta degli elementi **publicInvolved**.
Per maggiori dettagli cfr. modello dati dell'entità publicEngagement.
Configurazioni della ValidateLogic:

- ap.pen.publicInvolved.required.onlyTraining-communication.enabled

checkDatePublicEngagement
Questa validazione verifica la coerenza delle informazioni temporali dell'iniziativa di PE.
Vengono applicate le regole esplicitate dalla seguente tabella.
Nella colonna note viene riportato "NON NECESSARIO" nel caso in cui la validazione NON viene implementata perché già prevista una validazione di obbligatorietà per un campo.
Ad esempio la validazione "C" non è necessaria perché la combinazione valida il caso in cui knownEndDate sia vuoto.
Questa condizione però non si potrà mai verificare perché l'attributo è sempre obbligatorio.
Ragionamento analogo per le altre validazioni non necessarie.

	periodicEvent	oneDayOnlyEvent	knownEndDate	CONDIZIONE DI ERRORE	NOTE

A	SI	SI	SI	endDate is null or numTotalDay<>1	
B	SI	SI	NO	endDate is not null or numTotalDay<>1	
C	SI	SI	EMPTY	endDate is not null or numTotalDay<>1	NON NECESSARIA
D	SI	NO	SI	endDate is null or numTotalDay=1	
E	SI	NO	NO	endDate is not null or numTotalDay=1	
F	SI	NO	EMPTY	endDate is not null or numTotalDay=1	NON NECESSARIA
G	SI	EMPTY	*	knownEndDate is not null or endDate is not null	NON NECESSARIA
H	NO	SI	SI	endDate is null or numTotalDay<>1 or endDate-startDate+1<>1	
I	NO	SI	NO	la configurazione è di suo un errore	
L	NO	SI	EMPTY	la configurazione è di suo un errore	NON NECESSARIA
M	NO	NO	SI	endDate is null or numTotalDay=1 or numTotalDay is null or endDate-startDate<>numTotalDay+1	
N	NO	NO	NO	endDate is not null or numTotalDay is not null	
O	NO	NO	EMPTY	endDate is not null or numTotalDay is not null	NON NECESSARIA
P	NO	EMPTY	*	knownEndDate is not null or endDate is not null	NON NECESSARIA

Oltre ai campi riportati in tabella viene verificata anche:

- la coerenza degli attributi numHoursIfLessThanOneDay (valore ammesso tra 1 e 7 solo con iniziative di un solo giorno)
- periodicity (necessario se evento periodico)
- periodicity (se presente numTotalDay) rispetti i seguenti vincoli

periodicity	**check**	**messaggio di errore**
publicEngagementPeriodicity.continued	numTotalDay==null	La periodicità è continuativa e la durata in giorni quindi deve essere vuota
publicEngagementPeriodicity.continued	periodicEvent==true and oneDayOnlyEvent==false	La periodicità è continuativa e la durata in giorni quindi deve essere vuota
publicEngagementPeriodicity.weekly	numTotalDay<7	La periodicità è settimanale e la durata in giorni quindi deve essere inferiore a 7
publicEngagementPeriodicity.monthly	numTotalDay<31	La periodicità è mensile e la durata in giorni quindi deve essere inferiore a 31
publicEngagementPeriodicity.annual	numTotalDay<366	La periodicità è annuale e la durata in giorni quindi deve essere inferiore a 366

- numTotalDay (se presente sia <=36600)

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

isPeriodicEvent
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Torna true se l'attributo booleanMap[periodicEvent] è valorizzato con SI (true)

isNotPeriodicEvent
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Torna true se l'attributo booleanMap[periodicEvent] è valorizzato con NO (false).

isKnownEndDate
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Torna true se l'attributo booleanMap[knownEndDate] è valorizzato con SI (true)

evaluationToolEnabled
Questa è un'applicability rule che viene valutata prima di eseguire altre validazioni.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Torna true se l'attributo booleanMap[evaluationEnable] è valorizzato con SI (true)

wfStartLogicResearchCentre
Questa logica, in caso di creazione di un centro di ricerca figlio della tipologia estratta dalla variabile **ap.rsc.interuniversity.rootType** crea come partner di default l'Ateneo.
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)
Dal punto di vista del modello dati si tratta di **partner**
Per maggiori dettagli cfr. modello dati dell'entità in questione
Configurazioni della StartLogic:

- ap.rsc.interuniversity.rootType
- rm.orgunit.external.myOrganization

```
                        //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 parentWfItemTypeSetIterator = parentWfItemTypeSet.iterator();

                                while(parentWfItemTypeSetIterator.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 = gaService.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();

                                                        var researchCentrePartnerRoleMainSeat = null;

                                                        while(researchCentrePartnerRoleWfDictionaryListIterator.
hasNext()){

                                                                var researchCentrePartnerRoleWfDictionary =
researchCentrePartnerRoleWfDictionaryListIterator.next();
                                                                var  dictionaryCode =
researchCentrePartnerRoleWfDictionary.getStringMap().get("code");

                                                                if(Packages.org.apache.commons.lang.StringUtils.
equals("researchCentrePartnerRole.main-seat", dictionaryCode)){
                                                                        researchCentrePartnerRoleMainSeat =
researchCentrePartnerRoleWfDictionary;
                                                                }
                                                        }

                                                        if(researchCentrePartnerRoleMainSeat != null){
```

```
                                                                partnerElement.getWfDictionaryMap().put
("roleId", researchCentrePartnerRoleMainSeat);

                                                                wfService.saveOrUpdate(partnerElement);
                                                                wfItem.getWfItemElementSet().add
(partnerElement);
                                                                wfService.saveOrUpdate(wfItem);

                                                                break;
                                                        }
                                                    }else{
                                                        //        errors.reject("error.researchCentre.rm.
orgunit.external.myOrganization.configuration.required");
                                                    }
                                                }
                                            }
                                        }

                                        true;
```

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

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

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 contegggio 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

```
                        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.getDateMap().get("startDate") != null && owner.getDateMap().
get("startDate").compareTo(versionDate) <= 0 && (owner.getDateMap().get("endDate") == null || owner.
getDateMap().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();
                                            ownerSummaryElement.setDiscriminator("ownerSummary-snapshot");
                                            ownerSummaryElement.setWfItemId(wfItemCurrentMostValidated.getId());
                                            ownerSummaryElement.getWfDictionaryMap().put("roleId", WfUtil.
getWfDictionaryByCode(key, wfService));
                                            ownerSummaryElement.getIntegerMap().put("count", new Packages.java.lang.
Integer(ownerSummaryMap[key]));
                                            wfService.saveOrUpdate(ownerSummaryElement);
                                            wfItemCurrentMostValidated.getWfItemElementSet().add
(ownerSummaryElement);
                                        }
                                    }
                                }
                            }
```

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 contentente nella data di fine (attributo **dateMap[endDate]**) l'anno in questione.
Per maggiori dettagli cfr. modello dati dell'entità in oggetto.

```
                        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);
                                                        loopYear: for (currentYear; currentYear >
minYear; currentYear--) {
                                                                loopElement: for (var i = 0; i <
balanceElementList.size(); i++) {
                                                                        var currentElement =
balanceElementList.get(i);
                                                                        if(currentElement.getDateMap().
get("endDate").get(Calendar.YEAR) == 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));
                                                newElement.getNumberMap().put("value", 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 tipoliga 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 && 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 && object.
getBooleanMap().get("copyConstitutionDateInsideOwner") != null && object.getBooleanMap().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);
                                                }
                                        }

                                        var foundMyOrganization = false;
                                        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 {
                                                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() == 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 && object.getBooleanMap().get
("copyConstitutionDateInsideOwner")){
                                                newElement.getDateMap().put("startDate", object.getDateMap().get
("constitutionDate"));
                                        }
                                           newElement.getWfDictionaryMap().put("spinoffOwnerType", WfUtil.
getWfDictionaryByCode("spinoffOwnerType.structured-external-organization", wfService));
                                             newElement.getOrganizationUnitMap().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 && object.getBooleanMap().get
("copyConstitutionDateInsideOwner")){
                                        newElement.getDateMap().put("startDate", object.getDateMap().get
("constitutionDate"));
```

```
                                }
                                                newElement.getWfDictionaryMap().put("spinoffOwnerType", WfUtil.
getWfDictionaryByCode("spinoffOwnerType.structured-external-organization", wfService));
                                                newElement.getOrganizationUnitMap().put("ouId", myOrganization);
                                                wfService.saveOrUpdate(newElement);
                                        }
                                }
                        }
                        object.getBooleanMap().put("copyContributorAndExternalOrganizationUnitInsideOwner",
null);
                        object.getBooleanMap().put("copyConstitutionDateInsideOwner", null);
```

spinoffWfActionLogicRestoreConstitution
Questa logica serve per ripristinare la Spin-off con i dati presenti quando è stata costituita e per rimuovere tutti i monitoraggi e la costituzione.
Questa logica viene attivata SOLO se l'utente conferma di voler eseguire tale operazione.
Tutti monitoraggi e la costituzione verranno settati con **withdrawn = true**.
Dal punto di vista del modello dati si tratta degli attributi di tipo **booleanMap[restoreConstitution]**.
Per maggiori dettagli cfr. modello dati dell'entità in oggetto.

```
                        if(object.getBooleanMap().get("restoreConstitution") != null && 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
("Costituzione")){
                                        if(!versionLink.getChild().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.

spinoffCheckCopyContributorAndExternalOrganizationUnitInsideOwnerValidator
Questa validazione chiede all'operatore se vuole portare i Proponenti, Membri interni/esterni e l'ateneo (SE la tipoliga 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.

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.

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.

spinoffBoardUniversityMemberRemunerationElementValidator
Questa validazione controlla che in fase di salvataggio del fragment **boardUniversityMember** sia presente almeno un elemento dal fragment **remuneratio**
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.

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.

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 Inizio dell'elemento sia minore o uguale della data del corrento Monitoraggio.
Verifica che la data di Fine dell'elemento sia minore o uguale della data del corrento 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.

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.

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.

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.

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.

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.

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.

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.
- Sceelta 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.

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 **wfDictionaryMap[versionSelector]**, **dateMap[versionDate]**.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

spinoffFullTransferQuoteValidator
Questa validazione controlla che l'ateneo non abbia più quote di finanziamento associate SE la tipologia è **spinOff non partecipata dall'Ateneo** o se si sta tentando di portare la Spin-off nello stato Ceduto (**sold**).
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

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 inzio 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]**.
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.
Configurazioni della ValidateLogic:

- ap.spi.DISCRIMINATOR_ELEMENT.endDateMonitoring.required

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.

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

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.

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.

isNotCancelMonitoring
Questa è un'applicability rule che pilota le validazioni a cui è agganciata.
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.

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:

- 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 recuperata dalla configurazione: **ap.spi.ELEMENTO.validator.date.start.discriminator**
- 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.prepopulate.monitoring.**DISCRIMINATOR**.endDate.enabled è valorizzata con true.
Il segnaposto DISCRIMINATOR deve essere sostituito con il nome dell'elemento.
Ad esempio ap.spi.prepopulate.monitoring.**genderEquality**.endDate.enabled 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.prepopulate.monitoring.ELEMENTO.endDate.enabled

wfActionLogicEnterSetForwardInConfirmedDate

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

wfActionLogicEnterSetForwardInCompliantDate

```
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.

```
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.

```
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

```
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

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 configurazion **ap.(PRJ-CON).approval.approvalDateAfterStartDate.enabled** (una per i progetti e una per i contratti), queste due per le relative tipologie 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 due 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
Configurazioni della ValidateLogic:

- ap.checkDateExtensionValidator.enabled

- ap.prj.approval.approvalDateAfterStartDate.enabled
- ap.con.approval.approvalDateAfterStartDate.enabled

wfStartLogicIdentifier
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
Viene effetuata 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 effetuata la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:
<CODE_TIPOLOGIA>-<PROGRESSIVO>

- 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 (isPresent) {
                                var maxValSequence = wfService.getMaxSequencesValueFromIdentifier
(identifierCode + "-");
                                maxValSequence++;
                                counter = wfService.updateSequenceFromNumber(sequenceName, maxValSequence);
                                identifier = buildIdentifier(maxValSequence, identifierCode);
                            }
                            wfItem.setIdentifier(WfUtil.escapeForUgovPJ(identifier));
                        }
```

wfStartLogicYearFromProposalStartDateIdentifier
Questa logica viene eseguita in fase di creazione di un nuovo progetto.
Viene effetuata la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:
<ANNO>-<CODICE>-<PROGRESSIVO>

- ANNO: estratto dalla Data di avvio pratica (proposalStartDate)
- CODICE: come da tassonomia ap-item-type.xls; se si applica questa logica di costruzione identificativo evitare che il CODE abbia il carattere trattino -, non può essere creata su DB la sequenza corrispondente per il progressivo
- PROGRESSIVO: numero progressivo contestuale all'anno e codice

Esempio di costruzione dell'identificativo **2021-AL-0002**.
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 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 (isPresent) {
                                var maxNumber = wfService.getMaxSequencesValueFromIdentifier(identifierCode + "-
");
                                maxNumber++;
                                counter = wfService.updateSequenceFromNumber(sequenceName, maxNumber);
                                identifier = buildIdentifier(counter, identifierCode);
                            }
                            wfItem.setIdentifier(WfUtil.escapeForUgovPJ(identifier));
                        }
```

wfStartLogicYearFromStartDate
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
L'anno viene calcolato solo se il metadato è ancora nullo.
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

```
                        if(wfItem.getYear() == null || 2000 == wfItem.getYear()){
                                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)));
                        }
```

wfStartLogicOwnerDependentItemTypeFirstIdentifier
Questa logica effettua la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:
<CODE_TIPOLOGIA>-<ANNO>-<PRIMA_LETTERA_NOME>-<PRIME_QUATTRO_LETTERE_COGNOME>-<PROGRESSIVO>

- 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.

```
                    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(0,4) : 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 = "";
                                if (count < 9)
                                        countString += "0";
```

```
                countString += (count+1);
                identifier+= "_" + countString;

                wfItem.setIdentifier(WfUtil.escapeForUgovPJ(identifier));
        }
```

wfStartLogicOwnerDependentLastNameFirstIdentifierAndAcronym
Questa logica effettua la costruzione dell'identificativo dell'oggetto seguendo le seguenti regole:
<PRIME_QUATTRO_LETTERE_COGNOME>_<PRIMA_LETTERA_NOME>_<ANNO>_<CODE_TIPOLOGIA>_<ACRONIMO>_<PROGRESSIVO>

- PRIME_QUATTRO_LETTERE_COGNOME: prime quattro lettere del cognome del responsabile
- PRIMA_LETTERA_NOME: prima lettera del nome del responsabile
- ANNO: anno in due cifre estratto dalla data di inizio
- CODE_TIPOLOGIA: come da campo CODE da tassonomia scaricata dalla sezione "Alberatura tipologie RM/AP"
- ACRONIMO: acronimo dell'oggetto (può non essere presente)
- PROGRESSIVO: numero progressivo

Esempio di costruzione dell'identificativo **ROSS_P_21_PRJ_01**.
Per maggiori dettagli cfr. modello dati dell'entità Progetto e l'alberatura delle tipologie.
Configurazioni della StartLogic:

- ap.identifier.logic.date.discriminators

```
                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+1);
                                identifier += "_" + countString;

                                wfItem.setIdentifier(WfUtil.escapeForUgovPJ
(identifier));
                        }
```

wfStartLogicDepartmentAndOwnerDependentIdentifier
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

```
                        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";
                        }
                        switch(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;
                                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+1);
                                identifier += "_" + countString;
                            wfItem.setIdentifier(WfUtil.escapeForUgovPJ(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
- 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
- PROGRESSIVO: numero progressivo

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.

```
                        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+1);
                                identifier += "_" + countString;

                                wfItem.setIdentifier(WfUtil.escapeForUgovPJ
(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
- ANNO: anno in quattro cifre estratto dalla data di inizio del Bando di finanziamento collegato (se non presente verrà inserito "XX")
- ACRONIMO: acronimo dell'oggetto (può non essere presente)
- PRIME_SEI_LETTERE_COGNOME: prime sei lettere del cognome del responsabile
- PRIMA_LETTERA_NOME: prima lettera del nome del responsabile
- PROGRESSIVO: numero progressivo

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.

```
                        //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();

                                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);
                        stringBuilder.append('_');
                        stringBuilder.append(yearString);
                        stringBuilder.append('_');
                        if(acronym != null){
                                acronym = acronym.replaceAll("\s", "");
                                acronym = acronym.toUpperCase();
                                stringBuilder.append(acronym);
                                stringBuilder.append("_");
                        }
                        stringBuilder.append(lastName);
                        stringBuilder.append("_");
                        stringBuilder.append(firstName);

                        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+1);
                                identifier+= "_" + countString;

                                wfItem.setIdentifier(WfUtil.escapeForUgovPJ(identifier));
                        }
```

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.

```
                                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 == position.getPriority()) {
                                                if ("research".equals(position.getDiscriminator()) &&
"academicField2000"
                                                                                        .equals(position.getOrganizationUnit().
getOrganizationUnitType().getDescription())) {
                                                                                var ssdElement=new Packages.it.cilea.wf.model.
WfItemElement();
                                                                                ssdElement.setDiscriminator
("academicField2000");
                                                                                ssdElement.setWfItemId(wfItem.getId());
                                                                                ssdElement.getOrganizationUnitMap().put
("organizationUnit", position.getOrganizationUnit());
                                                                                wfService.saveOrUpdate(ssdElement);
                                                                                wfItem.getWfItemElementSet().add(ssdElement);
                                                                                break;
                                                }
                                        }
                                }

                                true;
```

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 **in qualsiasi altro caso** 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**.
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' 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

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.inferredChid.csv**.
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.inferredChid.csv** è valorizzata con **internalOrganizationUnitCost** questo vorrà dire che verrà creato un elemento di questo tipo associato alla struttura del 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.inferredChid.csv
- ap.prj.owner.customizableAutomaticInternalOrganizationExtractionInCreationPhase.enabled
- ap.TIPOLOGIA.FRAGMENT.role.main
- ap.TIPOLOGIA.owner.department.enabled
- ap.TIPOLOGIA.owner.position.context

```
                        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
("internalOrganizationUnit");

                                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);
                                }
                                WfUtil.addRoleInElementIfDefined(wfItem, ownerElement, wfService);

                                if (departmentSpecifiedInOwnerElement=="true" && department!=null &&
customizedInternalOrganization!=true){
                                        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);
                                                WfUtil.addRoleInElementIfDefined(wfItem,
internalOrganizationUnitElement, wfService);
                                        } else if(departmentCustom!=null &&
customizedInternalOrganization==true) {
                                                internalOrganizationUnitElement.getOrganizationUnitMap().put
("ouId", departmentCustom);
                                                WfUtil.addRoleInElementIfDefined(wfItem,
```

```
internalOrganizationUnitElement, wfService);
                                        }

                                        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);
                                }
                                if(wfItem.getOrganizationUnitMap().get("internalOrganizationUnit") != null){
                                        wfItem.getOrganizationUnitMap().put("internalOrganizationUnit", null);
                                }
                                wfItem.getBooleanMap().put("customizedInternalOrganization",
null);
                        }
                        true;
```

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 variaible **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

```
                                        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));
                                                                        partnerElement.getWfDictionaryMap().put
("partnerRole", 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[ouId]).
Aggiunge anche, se previsto il ruolo di default dell'unità organizzativa e 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 come 'ouRolePublicEngagement.manager', vuol dire che il ruolo con codice 'ouRolePublicEngagement.manager' 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 'ouRolePublicEngagement.manager', vuol dire che il ruolo con codice 'ouRolePublicEngagement.manager' è quello che verrà associato all'unità organizzativa selezionata nel caso di tutte le iniziative di public engagement
Configurazioni della StartLogic:

- ap.TIPOLOGIA.FRAGMENT.role.main

```
                        //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);
                        }
                        true;
```

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

```
```

```
                        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()){
                                                        var parentParent = parentParentSetIterator.next();
                                                        typeForConfig = parentParent;
                                                        break;
                                                }
                                        }
                                        break;
                                }
                        }
                        var profiles = '';
                        if(typeForConfig.getIdentifier() == 'CON' || 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;
```

wfStartLogicInheritInternalOrganizationUnitFromTeamOfLoggedUser
Questa logica viene eseguita in fase di creazione di un nuovo oggetto in visione dipartimentale.
Precisamente:

- viene recuperata la variabile di configurazione **ap.<macrotype>.internalOrganizationUnit.inheritFromTeamOfLoggedUser.enabled**
- SOLO se la configurazione è valorizzata con true allora
    - nel form di creazione viene presentata una tendina per consentire la scelta della struttura alla quale agganciare l'oggetto che si sta creando
    - alla nuova struttura viene associato il ruolo definito nella variabile **ap.<macrotype>.internalOrganizationUnit. inheritFromTeamOfLoggedUser.role**, se definita

Consideriamo questo esempio.
Supponiamo un operatore in visione dipartimentale profilato nei team

- Profilo di dipartimento per Dipartimento di Matematica
- Profilo di dipartimento per Dipartimento di Fisica

Supponiamo anche la variabile di configurazione

- ap.prj.internalOrganizationUnit.inheritFromTeamOfLoggedUser.enabled valorizzata a true
- ap.prj.internalOrganizationUnit.inheritFromTeamOfLoggedUser.role valorizzata a ouRole.main (Principale)

Questo vuol dire che l'operatore, una volta cliccato sul bottone "Nuovo" vedrà una tendina con la lista delle strutture in cui è profilato (Dipartimento di Matematica e Dipartimento di Fisica)
Una volta scelta la struttura, verrà creato l'oggetto agganciato a quella specifica struttura marcata con il ruolo ouRole.main (Principale)
Attenzione che tipicamente questa logica viene eseguita dopo quella standard che estrae la struttura dal responsabile.
Qualora la struttura estratta dal responsabile sia la stessa dell'operatore che sta facendo la creazione, NON verrà aggiunta un'altra struttura ma verrà mantenuto l'elemento già presente con il relativo ruolo.

Per uniformità con il comportamento della start logic **wfStartLogicMultipleOwners** viene replicato il funzionamento pilotato da **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.
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 selezionata dall'operatore.
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.
Configurazioni della StartLogic:

- ap.TIPOLOGIA.internalOrganizationUnit.inheritFromTeamOfLoggedUser.enabled
- ap.TIPOLOGIA.internalOrganizationUnit.inheritFromTeamOfLoggedUser.role

```
                        if (GaUtil.isCurrentUserInDepartmentView()){
                                var wfItemType = wfService.getWfItemType(wfItem.getWfItemTypeId());
                                var inheritInternalOrganizationUnitFromTeamOfLoggedUser=ConfigurationUtil.
getLenientBoolean("ap."+ wfItemType.getRootWfItemType().getIdentifier().toLowerCase() +".
internalOrganizationUnit.inheritFromTeamOfLoggedUser.enabled", false);
                                if (inheritInternalOrganizationUnitFromTeamOfLoggedUser){
                                        var internalOrganizationUnitInheritedFromTeamOfLoggedUser = wfItem.
getOrganizationUnitMap().get("internalOrganizationUnitInheritedFromTeamOfLoggedUser");

                                        var iouAlreadyPresent=false;
                                        var iouSet = FragmentUtil.getCurrentFragmentSetByParentAndDiscriminator
(wfItem, "it.cilea.wf.model.WfItemElement","internalOrganizationUnit", wfService);
                                        var iouIterator=iouSet.iterator();
                                        while (iouIterator.hasNext()){
                                                var iou=iouIterator.next();
                                                if (internalOrganizationUnitInheritedFromTeamOfLoggedUser.
getId()!=null && internalOrganizationUnitInheritedFromTeamOfLoggedUser.getId()==iou.getOrganizationUnitMap().get
("ouId").getId())
                                                        iouAlreadyPresent=true;
                                        }

                                        if(!iouAlreadyPresent){
                                                var internalOrganizationUnitElement=new WfItemElement();
                                                internalOrganizationUnitElement.setDiscriminator
("internalOrganizationUnit");
                                                internalOrganizationUnitElement.setWfItemId(wfItem.getId());
                                                internalOrganizationUnitElement.getOrganizationUnitMap().put
("ouId", internalOrganizationUnitInheritedFromTeamOfLoggedUser);

                                                var role=ConfigurationUtil.getConfigValue("ap."+ wfItemType.
getRootWfItemType().getIdentifier().toLowerCase() +".internalOrganizationUnit.inheritFromTeamOfLoggedUser.
role");
                                                if (role){
                                                        internalOrganizationUnitElement.getWfDictionaryMap().put
("roleId", WfUtil.getWfDictionaryByCode(role, wfService));
                                                }


                                                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);
                                        }
                                }
                                wfItem.getOrganizationUnitMap().put
("internalOrganizationUnitInheritedFromTeamOfLoggedUser", null);
                        }
                        true;
```

approvalOrganizationUnitValidator
Questa validazione controlla le approvazioni inserite, cercando quelle con strutture agganciate.
In tal caso verifica che tutte le strutture siano censite anche nella sezione **Strutture**.
Dal punto di vista del modello dati si tratta degli elementi di tipo **approval** e **internalOrganizationUnit**.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

ownerAndOrganizationUnitMatchValidator
Questa validazione controlla che esista per ogni responsabile scientifico dell'oggetto, una unità organizzativa che sia interna nel momento della data di avvio dell'oggetto (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. modello dati dell'entità in questione.

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

inheritInternalOrganizationUnitFromTeamOfLoggedUserValidator
Questa validazione viene attivata solo in fase di creazione dell'oggetto.
Viene triggerata SOLO in visione dipartimentale e se la configurazione ap.<macrotype>.internalOrganizationUnit.inheritFromTeamOfLoggedUser.enabled è valorizzata con true
Questa validazione controlla che sia stata selezionata la struttura alla quale agganciare l'oggetto che si sta creando
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.internalOrganizationUnit.inheritFromTeamOfLoggedUser.enabled
- ap.TIPOLOGIA.internalOrganizationUnit.inheritFromTeamOfLoggedUser.role

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 **ap.<macrotype>.personal-view.create-for-other-people.enabled** è valorizzata a **true**, dove <macrotype> indica la macrotipologia dell'oggetto che si sta creando.
- l'attributo booleano **isCreatorOwner**, specificato in fase di creazione, è valorizzato a **false**

Per maggiori dettagli cfr. modello dati dell'entità in questione.
Configurazioni della StartLogic:

- ap.TIPOLOGIA.personal-view.create-for-other-people.enabled

```
                        var isCreatorOwner=wfItem.getBooleanMap().get("isCreatorOwner");
                        var wfItemType = wfService.getWfItemType(wfItem.getWfItemTypeId());
                        var rootWfItemType = wfItemType.getRootWfItemType();
                        var createForOtherPeopleEnabled = ConfigurationUtil.getConfigValue("ap." +
rootWfItemType.getIdentifier().toLowerCase() + ".personal-view.create-for-other-people.enabled");
                        if (createForOtherPeopleEnabled=="true" && !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 PJ).
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.

```
                        wfItem.getBooleanMap().put("legacy", false);
                        true;
```

wfStartLogicVisibileOnPortal
Questa logica viene eseguita in fase di creazione di un nuovo oggetto e solo se il metadato non è ancora popolato.
Viene effetuata 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.

```
                                        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

```
                        if(wfItem.getBooleanMap().get("visibleOnPortal") == null){
                                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);
                                                }

                                                if (Packages.org.apache.commons.lang.StringUtils.
equalsIgnoreCase(visibleOnPortalConfiguration, "false")){
                                                        wfItem.getBooleanMap().put("visibleOnPortal", false);
                                                }
```

```
                                    }
                                }
                            }
```

wfStartLogicIntegerMapYearFromStartDate
Questa logica viene eseguita in fase di creazione di un nuovo oggetto.
L'anno del metadato **integerMap[year]** viene recuperato (SOLO se ancora nullo) dalla Data di inizio (metadato **dateMap[startDate]**).

integerMapYearConsistencyValidator
Questa validazione verifica la coerenza del valore del metadato **integerMap[year]** rispetto alle date di inizio e fine dell'oggetto.
In particolare verifica che l'anno inserito NON sia minore dell'anno della data di inizio e che NON sia maggiore dell'anno della data di fine.
Per maggiori dettagli cfr. modello dati dell'entità in questione.

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.dateMap[endDate]**, e confrontati con le date di validità dell'item **dateMap[startDate]**, **dateMap[endDate]**, se non diversamente configurate.
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 ci sia continuità degli intervalli temporali.
  Se settata a **true** viene verificato che per gli elementi con quel discriminator esista un intervallo in cui sia previsto un elemento con la data di inizio dell'oggetto.
  Se settata a **true** viene verificato che per gli elementi con quel discriminator se nel caso in cui la data di fine sia popolata, esista un intervallo in cui sia previsto un elemento con la data di fine dell'oggetto.
  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.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.coverage = true** e configurazione **ap.prj.owner.overlap = true**, si sta indicando che ci deve essere una copertura completa per tutti il periodo di attività dell'oggetto, che NON ci siano sovrapposizioni di intervalli da parte dei Responsabili scientifici e che vengono valutati TUTTI

Le date dell'item che definiscono l'intervallo di validità sono configurabili dai parametri **ap.<macrotype>.<elementDiscriminator>.validator.date.<start|end>.discriminator**, in cui è possibile specificare un discriminator diverso da 'startDate' ed 'endDate' (che sono i rispettivi valori di default)
Se assenti, per owner e contributor vengono cercate le **ap.<macrotype>.contributorAndOwner.validator.date.<start|end>.discriminator**
(le medesime configurazioni pilotano anche le validazioni sulle date per owner e contributor: cfr checkElementDateBetweenItemDateValidator) 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
- ap.TIPOLOGIA.xxx.validator.date.start.discriminator
- ap.TIPOLOGIA.xxx.validator.date.end.discriminator
- ap.TIPOLOGIA.contributorAndOwner.validator.date.start.discriminator
- ap.TIPOLOGIA.contributorAndOwner.validator.date.end.discriminator

multipleContributorValidator
Questa validazione verifica che esista almeno un elemento di tipo **contributor**.
Se la variabile di configurazione **ap.<sourceItemType>.<contributor.withoutContributorConfirmation.enabled** è valorizzata con true viene consentito di procedere senza l'inserimento dopo avere confermato di volere procedere con dati NON completi.
Dal punto di vista del modello dati si tratta degli elementi di tipo **contributor**.
Per maggiori dettagli cfr. il modello modello dati dell'entità in questione.

startDateAndEndDateValidator
Questa validazione controlla che la date 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.

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:

- crwfd

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

checkElementDateBetweenItemDateValidator
Questa validazione controlla la coerenza delle date di inzio e fine in fase di salvataggio del fragment.
Verifica 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 inzio 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 della 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

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 pilotatao 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

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 pilotatao 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.
Qualora l'owner|contributor non preveda il ruolo, è possile lo stesso effettuare l'aggiunta della struttura con il ruolo. In questo caso viene attribuito alla struttura il ruolo presente nella configurazione **ap.<tipologia>.internalOrganizationUnit.role.default**, se non dovesse essere presente NON viene aggiunto il ruolo.

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.fullyInferredFromPeopleType.csv**
Questa variabile abilita il ricalcolo completo 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 estrarre le info delle strutture.
E' comunque sempre possibile operare in edit sulle strutture inferite.
questo significa che

1. se viene settata a true la variabile **ap.<tipologia>.internalOrganizationUnit.fullyInferredFromPeopleType.csv**
2. se viene modificata/eliminata/aggiunta direttametne una struttura nella sezione delle strutture
3. se viene modificata/eliminata/aggiunta una persona

verrà forzato il ricalcolo delle strututre 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.fullyInferredFromPeopleType.csv** 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.inferredChid.csv**.
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 come 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.fullyInferredFromPeopleType.csv
- ap.TIPOLOGIA;.internalOrganizationUnit.inferredChid.csv

deleteOrgUnitAssociatedToOldOwnerValidator
Questa logica, ogni volta che viene ELIMINATA 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.inferredChid.csv** viene effettuata anche la cancellazione degli elementi elencati come associati all'unità organizzativa.

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 **ouId**
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.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.
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.inferredChid.csv
- ap.TIPOLOGIA.internalOrganizationUnit.inferredChid.ELEMENTO.jsValidator
- ap.TIPOLOGIA.[owner|contributor].role.COD_DIZIONARIO.inheritedInternalOrganizationUnit.role

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.inferredChid.csv** 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 **ouId**
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.internalOrganizationUnit.inferredChid.csv

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

internalOrganizationUnitDeleteLinkedElementValidator
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.inferredChid.csv** 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 **ouId**
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.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.
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.inferredChid.csv
- ap.TIPOLOGIA.internalOrganizationUnit.inferredChid.ELEMENTO.jsValidator

notEmptyResearchPositionValidatorTutor
Questa validazione si applica alla selezione di un tutor per una mobilità in ingresso.
Viene controllato se il tutor inserito abbia una posizione di ricerca presso un dipartimento alla data di inizio.

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, specifichi 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.
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 pilotatao 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.position.context

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 **suo** 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 dovesse casuare 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.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled** o la variabile di configurazione specifica **ap.HIERACHICAL_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.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled
- ap.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.validation.[enter|save|element:element_discriminator:save|delete].departmentChangeRoleValidator.parameters.allowRoleChangeWithConsequentWritePermissionsDenied.enabled

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

contributorAndOwnerValidatorWithStartEndDate
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

uniquePeopleValidator
Questa validazione controlla che NON ci siano nominativi con intervalli di date che si sovrappongono in base a come viene configurata.
Viene valuatata la configurazione **ap.<itemType>.uniquePeople.csv**.
Il valore della configurazione deve avere il seguente formato (ogni parametro deve essere separato dalla virgola):

- elenco di discriminator degli elementi da valutare separati dal carattere "|"
- true/false per indicare se sono consentiti elementi duplicate su intervalli temporali disgiunti
- true/false per indicare se sono consentiti elemente duplicate su ruoli distinti

Esempio: ap.prj.uniquePeople.csv=owner|contributor,true,false
Per maggiori dettagli cfr. excel modello dati dell'entità in questione.
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.uniquePeople.csv

transitionCommentValidator
Questa validazione, restringe solo ad alcuni attori l'obbligo di inserire la motivazione, per riportare allo stato di bozza (draft).
Viene valuatata la configurazione **ap.<itemType>.transitionComment.actor.discriminator.csv**.
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.discriminator.csv = 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.discriminator.csv

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 indacando 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 indacando 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

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 sia presente ALMENO una approvazione (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.prj.approval.inquiry=approvalType.department,approvalType.board: si sta indacando 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 indacando 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 o 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

wfActionLogicSaveNewYear
Questa logica serve per l'aggiornamento dell'anno della scheda in caso di cambio di data.
L'aggiornamento viene effettuato se viene recuperata una data e se l'anno della data è diverso dal valore presente nel campo anno (**year**).
La data viene recuperata dalla configurazione **ap.<wfItemType>.owner.position.date**,
dove può essere indicata direttamente (esempio: startDate), OPPURE:

se la configurazione viene valorizzarta con **CURRENT** o **CURRENT_OR_LAST**,
verrà effettuato un recupero della data presente nel metadato **startDate**,
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

```
it.cilea.wf.logic.action.save.WfActionLogicSaveNewYear
```

wfActionLogicSaveElementStartDate
Questa logica effettua l'inizializzazione della data di inizio dove iniettata e **SOLO se presente e valorizzata a true** la configurazione **ap.TYPE.prepopulate.[owner|contributor|administrativeOwner].startDate.enabled (ad esempio ap.con.prepopulate.owner.startDate.enabled)**.
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.prepopulate.ELEMENTO.startDate.enabled

```
it.cilea.wf.logic.action.save.WfActionLogicSaveElementStartDate
```

wfActionLogicSaveFieldReset
Questa logica resetta il valore specificato (anche element multitipo) se la condizione specificata è vera

```
it.cilea.wf.logic.action.save.WfActionLogicSaveFieldReset
```

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.

```
it.cilea.wf.logic.WfActionLogicTransitionLogger
```

notAllowedValidator
Questa logica scatena un errore e blocca l'azione sulla quale è collegata.

currentUserCannotEditHimself
Questa validazione viene scatenata in visione personale.
Impedisce che l'utente sostituisca se stesso dalla sezione in cui è implementata la validazione, le sezioni disponibili sono:

- owner

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 **DISCRIMINAOR_ELEMNT + IdPreviousValue**, che contiene l'id della persona precedente.

currentUserCannotDeleteHimself
Questa validazione viene scatenata in visione personale.
Impedisce che l'utente elimini se stesso dalla sezione in cui è implementata la validazione, le sezioni disponibili sono:

- internalRepresentative
- administrativeOwner
- owner

Se l'operazione fosse consentita, dopo il salvataggio, l'utente non avrebbe più i diritti per accedere all'oggetto.

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.

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.

changeItemTypeForNotSynchronizedItemValidator
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 Assitenza 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

changeItemTypeForSynchronizedItemValidator
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 Assitenza Tecnica in caso di scheda con flag legacy attivo.

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]**.
Viene valutata anche la configurazione **ap.<HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE>.<elementDiscriminator>.role.required**.
Se true o non definita viene supposta obbligatorietà del ruolo
Se false il ruolo non è obbligatorio

Vengono valutate anche 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>.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.<HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE>.<elementDiscriminator>.role.required.csv**
  Se presente, viene controllato che ogni ruolo elencato 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 **S ede 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.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.ELEMENTO.role.uniqueness.csv
- ap.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.ELEMENTO.role.required.csv

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 essepre 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

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.

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.

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 più creare nuovi item).
E' possibile mantenere un item che è già presente in quella tipologia.

wfActionLogicSaveNewIdentifier
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

```
                        var sourceItemType = wfService.getWfItemType(wfItem.getWfItemTypeId()).
getRootWfItemType();
                        var identifierRecalculationConfig = Packages.it.cilea.wf.util.WfUtil.
getParametricConfiguration("ap.<wfItemType>.identifier.recalculation.enabled", sourceItemType, true);
                        var identifierRecalculationEnabled = ("true" == identifierRecalculationConfig || !
identifierRecalculationConfig) ? true : false;
                        if(identifierRecalculationEnabled) {
                                var myNEWWfItemType = object.getWfItemTypeId();
                                var myOLDWfItemType = object.getIntegerMap().get("wfItemTypeIdPreviousValue");
                                if (myOLDWfItemType == null)
                                        throw "To use changeItemTypeValidator you MUST specify
wfItemTypeIdPreviousValue hidden value";

                                if(myNEWWfItemType != myOLDWfItemType){

                                        var recalculate = wfService.getWfItemDataValue(object.getId(),
"recalculate", "boolean");
                                        if (recalculate == null)
                                                recalculate = false;
                                        if (!recalculate) {
                                                var startLogic = Packages.it.cilea.wf.WfConstant.
START_LOGIC_MAP.get("wfStartLogicIdentifier");
                                                startLogic.start(object, request);
                                                //object.setWfStateId(wfStateId);
                                                object.getIntegerMap().put("wfItemTypeIdPreviousValue", object.
getWfItemTypeId());
                                        }
                                }
                        }
```

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

```
                        wfItem.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.

isYearGreaterOrEqualThanConfiguration
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 l'oggetto in questione sia relativo ad un anno maggiore o uguale a quello definito nella configurazione **ap.<macrotype>.year.validation.threashold (es: 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 oggetti che non rispettano nuove e più stringenti validazioni, ed evitare quindi di dovere fare una bonifica a tappeto.
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.year.validation.threashold

isCreateByCloning
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 **copyFrom**, tipicamente con etichetta "Copia da esistente" e la configurazione **ap.<macrotype>.create-by-cloning.enabled**.
Per tutti gli oggetti che hanno questo attributo valorizzato a true e la configurazione valorizzata a true vengono disattivate le validazioni.
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.create-by-cloning.enabled

areWarningEnabled
Questa è un'applicability rule che pilota i messaggi di warning.
Se ritorna true allora vengono eseguite le altre validazioni altrimenti no.
Se la variabile **ap.<macrotype>.warningAcknowlegement.validation.enabled** è settata a true o non è valorizzata i messaggi di warning sono abilitati.
Se la variabile **ap.<macrotype>.warningAcknowlegement.validation.enabled** è settata a false i messaggi di warning NON sono abilitati.
Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.warningAcknowlegement.validation.enabled

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

```
                        (
                                "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 dove xxxx è il nome del flusso** 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

```
                        (
                                (
                                        Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getLenientBoolean("ap." + object.getWfItemType().getWfDefaultDefinition().getDescription() + ".owner.create",
true)
                                )
                                &&
                                (
                                        !Packages.it.cilea.core.configuration.util.ConfigurationUtil.
getLenientBoolean("ap." + object.getWfItemType().getRootWfItemType().getIdentifier().toLowerCase() + ".personal-
view.create-for-other-people.enabled", false)
                                )
                        )
```

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

```
                        (
                                Packages.it.cilea.core.configuration.util.ConfigurationUtil.getLenientBoolean
("ap."+ object.getWfItemType().getWfDefaultDefinition().getDescription() +".owner.create", false)
                        )
```

childAndParentDetectorDeleteValidator
Questa è una validazione che viene chiamata in ogni flusso di Laboratori, Grupppi 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.

linkEntitiesAllowedStateValidator
Validazione che controlla il collegamento tra oggetti, in particolare verifica che gli oggetti collegati si trovino in determinati stati.
Viene valutata la configurazione parametrica: **ap.HIERACHICAL_ITEM_TYPE_IDENTIFIER_LOWERCASE.link.entities.csv**.
Qualora la configurazione NON sia presente o NON sia valorizzata la validazione NON effettuerà alcun controllo.
La configurazione deve seguire il seguente standard:

- Concatenazione dei seguenti parametri con il carattere "|"
  - discriminatorLink
  - child o parent
  - elenco degli stati accetabili separati dal carettere ";"
- concatenazione del carattere "," e degli attributi riportati sopra per il collegamento ad un altra entità
- per maggiore leggibilità è possibile utilizzare il carattere "a capo" dopo il carattere ","

Esempio: workgroupJointLink|child|approved;reopened,workgroupProjectLink|child|operative;concluded
Per maggiori dettagli cfr. modello dati dell'entità specifica.

isNewItem
Questa è un'applicability rule che ritorna true se l'item NON ha un id, quindi che non è ancora stato creato.
Tipiacemente questa applyLogic viene introdotta nelle validazioni in fase di creazione di un nuovo oggetto.

isNotNewItem
Questa è un'applicability rule che ritorna true se l'item ha un id, quindi che è già stato creato.
Tipiacemente questa applyLogic viene introdotta nelle validazioni successive allo stato iniziale.

elementStartDatePrepopulateLogic
Questa logica effettua l'inizializzazione della data di inizio dove iniettata e **SOLO se presente e valorizzata a true** la configurazione **ap.TYPE.prepopulate.ELEMENT_DISCRIMINATOR.startDate.enabled (ad esempio ap.prj.prepopulate.owner.startDate.enabled)**.
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.prepopulate.ELEMENTO.startDate.enabled

elementRolePrepopulateLogic
Questa logica effettua l'inizializzazione del ruolo del Dipartimento di afferenza **SOLO se presente e valorizzata a true** la configurazione **ap.TYPE.prepopulate.ELEMENT_DISCRIMINATOR.role.enabled (ad esempio ap.prj.prepopulate.internalOrganizationUnit.role.enabled)** e se esiste anche la configurazione **ap.TIPOLOGIA.ELEMENT_DISCRIMINATOR.role.default**.
Il ruolo viene valorizzato con il valore presente nella configurazione **ap.TIPOLOGIA.ELEMENT_DISCRIMINATOR.role.default**.
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).

Istruzioni per sanare il pregresso

Le seguenti operazioni vanno fatte nel caso in cui questa logica venga abilitata in un ambiente in cui siano già presenti degli item della entità in questione.

Lanciare la seguente query su AP

Questa query aggiunge il ruolo di default agli elementi già creati
Le seguenti parole chiave vanno sostituite con i valori corretti

- CODICE_RUOLO -> Valore della configurazione ap.TIPOLOGIA.FRAGMENT.role.default (es: ouRoleEquipment.other)
- CODICE_ENTITÀ_TO_UPPER_CASE -> Codice dell'entità (es: EQP)
- DISCRIMINATOR_ELEMENT -> Discriminator dell'elemento/sezione (es: internalOrganizationUnit)

```
                                    DECLARE
                                       vcount NUMBER (10);
                                    BEGIN
                                            SELECT count(*) INTO vcount FROM ap_dictionary WHERE
external_identifiers LIKE 'code=CODICE_RUOLO';
                                            IF (vcount = 1) THEN
                                                    SELECT id INTO roleId FROM ap_dictionary WHERE
external_identifiers LIKE 'code=CODICE_RUOLO';
                                                    FOR item_element in (
                                                            select ie.id
                                                            from ap_item i
                                                            LEFT JOIN ap_item_type it ON it.id = i.
fk_item_type AND it.identifier LIKE 'CODICE_ENTITÀ_TO_UPPER_CASE.%'
                                                            LEFT JOIN ap_item_element ie ON ie.fk_item = i.
id AND ie.discriminator = 'DISCRIMINATOR_ELEMENT'
                                                            LEFT JOIN ap_item_element_data ied ON ied.
fk_item_element = ie.id AND ie.discriminator = 'roleId'
                                                            WHERE ied.fk_dictionary_value is null
                                                            )
                                                    LOOP
                                                            INSERT INTO ap_item_element_data
(discriminator, fk_item_element, fk_dictionary_value)
                                                            VALUES('roleId', item_element.id, roleId);
                                                    END LOOP;
                                            END IF;
                                    END;
```

Fare una resync full solr

La resync permette di aggiorna i dati su solr, quindi di averli aggiornati anche sui servizi rest

```
                                    https://irisbuild01.private.cineca.it:9090/etl/job/ap-target-job-wfItem-
entities/build?delay=0sec
```

Configurazioni della ValidateLogic:

- ap.TIPOLOGIA.prepopulate.ELEMENTO.role.enabled
- ap.TIPOLOGIA.FRAGMENT.role.default