

## **PGED-ACTA-SRS-ACARIS Modulo “Servizi Applicativi”**

### **ACTA Archive Interoperability Services**

#### **Specifica dei requisiti del sistema**

**Versione <6.0>**

## INDICE

<b>1</b>	<b>Scopo e riferimenti del documento.....</b>	<b>4</b>
1.1	Scopo del documento .....	4
<b>2</b>	<b>Contesto di riferimento .....</b>	<b>4</b>
2.1	Modello di interoperabilità .....	4
2.2	Funzionalità del repository .....	7
<b>3</b>	<b>Modello dei Servizi.....</b>	<b>8</b>
3.1	Attori Principali.....	8
3.2	Quadro generale dei servizi .....	9
3.2.1	Repository Services .....	9
3.2.2	Object Services .....	9
3.2.3	Navigation Services .....	10
3.2.4	Multi-filing Services .....	10
3.2.5	Management Services .....	10
3.3	Descrizione delle singole operazioni.....	11
3.3.1	Repository Services .....	11
3.3.1.1	getRepositories .....	11
3.3.1.2	getRepositoryInfo .....	11
3.3.2	Object Services .....	12
3.3.2.1	createDocument .....	12
3.3.2.2	createAssociativeDocument.....	12
3.3.2.3	createFolder .....	12
3.3.2.4	createRelationship.....	12
3.3.2.5	getProperties .....	12
3.3.2.6	updateProperties .....	12
3.3.2.7	moveObject.....	12
3.3.2.8	getContentStream .....	13
3.3.3	Multi-filing Services .....	14
3.3.3.1	addAssociativeObjectToFolder.....	14
3.3.4	Relationships Services .....	15
3.3.4.1	getRelationships.....	15
3.3.5	Navigation Services .....	16
3.3.5.1	getDescendants [DEPRECATO] .....	16
3.3.5.2	getChildren .....	16
3.3.5.3	getFolderParent.....	16
3.3.5.4	getObjectParents .....	16
3.3.6	Management Services .....	17
3.3.6.1	Gestione contenuti .....	17
3.3.6.1.1	addAnnotazioni .....	17
3.3.6.1.2	getVitalRecordCode .....	17
3.3.6.2	Back office.....	17
3.3.6.2.1	dettaglioAOO .....	17
3.3.6.2.2	dettaglioStruttura .....	17
<b>4</b>	<b>Algoritmi del modulo ACARIS.....</b>	<b>18</b>

<b>4.1</b>	<b>ACARIS_A01_Property_Filters.....</b>	<b>18</b>
4.1.1	Finalità dell'algoritmo .....	18
4.1.2	Descrizione .....	18
<b>4.2</b>	<b>ACARIS_A02_Paging .....</b>	<b>20</b>
4.2.1	Finalità dell'algoritmo .....	20
4.2.2	Parametri in input .....	20
4.2.3	Output .....	20
4.2.4	Descrizione .....	20
<b>4.3</b>	<b>ACARIS_A03_Change_Tokens.....</b>	<b>21</b>
4.3.1	Finalità dell'algoritmo .....	21
4.3.2	Parametri in input .....	22
4.3.3	Output .....	22
4.3.4	Descrizione .....	22
<b>4.4</b>	<b>ACARIS_A04_Crea_PrincipalID.....</b>	<b>22</b>
4.4.1	Finalità dell'algoritmo .....	22
4.4.2	Parametri in input .....	22
4.4.3	Output .....	22
4.4.4	Descrizione .....	22
<b>4.5</b>	<b>ACARIS_A05_Decodifica_PrincipalID .....</b>	<b>23</b>
4.5.1	Finalità dell'algoritmo .....	23
4.5.2	Parametri in input .....	23
4.5.3	Output .....	23
4.5.4	Descrizione .....	23
<b>4.6</b>	<b>ACARIS_A06_Crea_ObjectId .....</b>	<b>23</b>
4.6.1	Finalità dell'algoritmo .....	23
4.6.2	Parametri in input .....	24
4.6.3	Output .....	24
4.6.4	Descrizione .....	24
4.6.5	Applicazione generale delle regole di valorizzazione degli ObjectId .....	26
<b>4.7</b>	<b>ACARIS_A07_Decodifica_ObjectId.....</b>	<b>26</b>
4.7.1	Finalità dell'algoritmo .....	26
4.7.2	Parametri in input .....	26
4.7.3	Output .....	26
4.7.4	Descrizione .....	26
4.7.5	Regole per l'accesso agli oggetti in operazioni di navigazione e lettura del dettaglio degli oggetti.....	26
4.7.5.1	Oggetti riconducibili al modulo Acta GCO .....	27
4.7.5.2	Oggetti riconducibili al modulo Acta GSA.....	29
<b>5</b>	<b>Tecnologie.....</b>	<b>30</b>

## 1 Scopo e riferimenti del documento

### 1.1 Scopo del documento

L'applicativo ACTA ha il compito di archiviare la documentazione prodotta e ricevuta da un Ente, utilizzando un ECM in grado di supportare politiche di gestione dei contenuti.

I processi di produzione dei documenti sono definiti e governati da applicativi che gestiscono il business di specifici ambiti ma hanno in comune l'esigenza di archiviare la documentazione prodotta.

Il ruolo di ACTA rispetto a questi applicativi è di permettere l'archiviazione tramite l'esposizione di servizi applicativi.

Per definire i servizi applicativi di ACTA è stato utilizzato come modello di riferimento il data model dello standard CMIS (*Content management Interoperability Services, V.0.6, Copyright OASIS, 2008*). Il modello è stato esteso per adattarlo alle esigenze di ACTA che pertanto verrà ad esporre servizi nelle modalità attese per un content repository; accanto ai servizi di accesso e gestione degli oggetti dell'archivio, documenti e strutture aggregative, sono disponibili servizi per funzionalità accessorie come lettura dati di configurazione o operazioni complesse di creazione oggetti.

In qualsiasi caso i riferimenti a CMIS non sono da considerare funzionali a produrre dei servizi totalmente compatibili con lo standard e quindi poter dichiarare ACTA CMIS compliant, bensì al recepire indicazioni sul modello dati e relative operazioni, così come disponibili sul sito dell'ente di standardizzazione ([OASIS CMIS](#)), per un content repository. Nel momento in cui lo standard fosse promulgato dall'OASIS in via definitiva si potrebbe ipotizzare una attività di adeguamento.

Scopo del presente documento è fornire l'elenco dei servizi applicativi esposti da ACTA, il documento contiene:

- descrizione del modello dati,
- descrizione del meccanismo di autorizzazione,
- elenco dei servizi applicativi pubblicati.

## 2 Contesto di riferimento

### 2.1 Modello di interoperabilità

La definizione dei servizi applicativi di ACTA (ACTA Services) richiede a tutti gli effetti la definizione di un layer di servizi al di sopra di un ERMS (Enterprise Record Management System). ACTA in qualità di ERMS e quindi di ECM system, non è perciò esente dalle considerazioni che si trovano nella letteratura dei sistemi di ECM sull'interoperabilità dei Content Management repositories.

Nella definizione di uno standard di interoperabilità ritroviamo la definizione di un modello dei dati di riferimento e di un insieme di operazioni, a loro volta raggruppate in servizi, fruibili in una o più tecnologie di networking. Il modello dei dati di riferimento (domain data model) a sua volta può essere specializzato per renderlo conforme alle specificità del Content Management Repository in oggetto, in questo modo sistemi di ECM diversi possono offrire ai propri client modalità di integrazione (dati ed

operazioni) omogenee.

Esempi di standard di interoperabilità di questo tipo sono stati nel tempo le JCR API, con le specifiche JSR-170 e la JSR-283, il frame work del comitato dell'AIIM denominato [iECM](#), ed infine il CMIS.

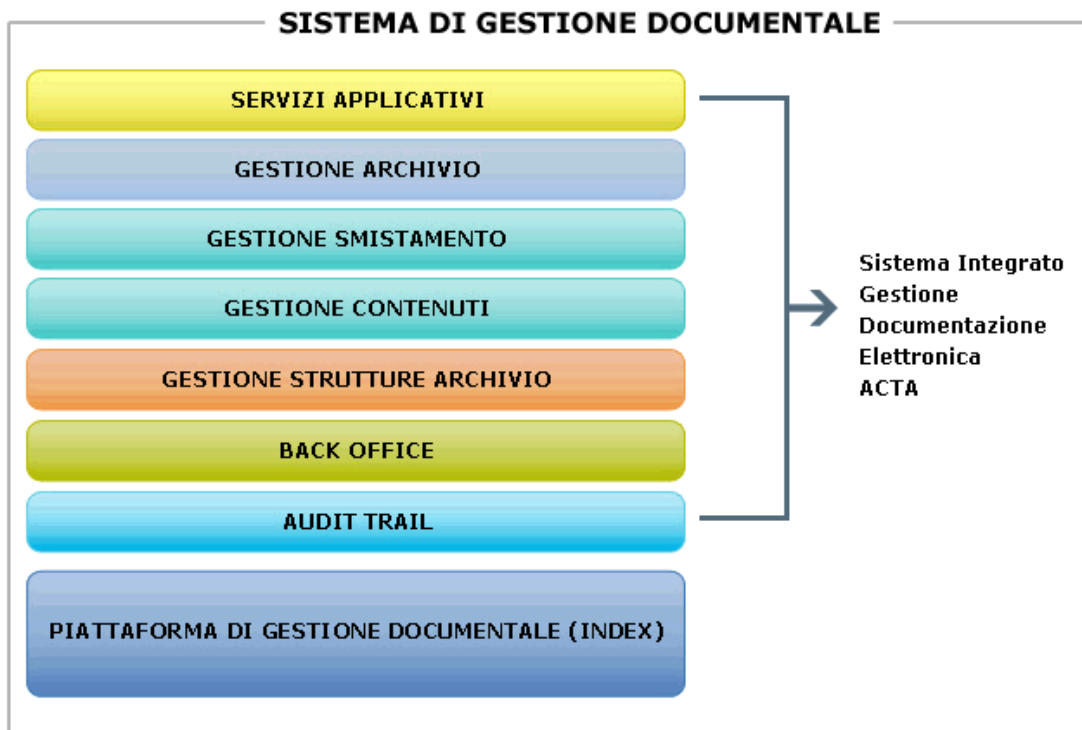
Ovviamente non tutte le specificità di un sistema di ECM possono essere ricondotte a questo modello, tipicamente servizi di configurazione ed amministrazione, autenticazione ed autorizzazione non possono essere convenientemente ricondotti a modelli comuni, anche se, in qualche caso, si potrebbe far uso di standard di riferimento ma difficilmente condivisibili.

Per gli ACTA Services si sono dovuti affrontare i problemi dapprima elencati, quindi i servizi di accesso e gestione degli oggetti dell'archivio si rifanno agli standard di interoperabilità per ECM systems mentre i servizi accessori, che trattano entità non riconducibili ad un Content Management Repository, sono assolutamente specifici di ACTA.

Infine occorre elencare i casi d'uso per cui gli ACTA Services sono stati definiti e per cui potranno essere costruite delle ECM application che ne faranno uso, fra questi casi d'uso possiamo elencare:

- Creazione collaborativa di contenuti, l'applicazione fornisce gli strumenti di redazione (authoring) mentre ACTA permette di archiviare e condividere i contenuti redatti,
- Workflow & BPM, nel corso dell'esecuzione delle attività del processo, umano o di sistema, occorre accedere ai contenuti di ACTA o archiviare contenuti in ACTA,
- Archiviazione (automatica) di contenuti provenienti da flussi informativi periodici (bollettini, mandati, etc.) o una tantum (migrazioni).

Il modello di riferimento, espresso nella figura seguente, posiziona i servizi applicativi all'interno dei confini del "sistema di gestione documentale".



Per approfondire il modello del sistema consultare le Specifiche dei Requisiti di Sistema di ACTA (PGED-ACTA-SRS-v01\_Requisiti funzionali, cfr. Allegato [A1]).

## 2.2 Funzionalità del repository

I servizi permettono ad una ECM application di lavorare sui contenuti di ACTA nel rispetto delle regole di business definite, ed utilizzando le classi fondamentali e le relazioni del modello UML di riferimento . Ovviamente non tutte le operazioni rese disponibili ad un utente opportunamente profilato per mezzo della user interface di ACTA saranno possibili con i servizi per i seguenti motivi:

- titolarità dei dati per cui un client può leggere informazioni ma non modificarle,
- funzionalità di ACTA che conducono ad entità definite nel modello concettuale strettamente interdipendenti per le quali non è ipotizzabile realizzare un'integrazione a servizi,
- funzionalità di ACTA che si concretizzano in elaborazioni batch.

Nella figura della pagina seguente sono indicate le relazioni fra il modello UML di ACTA Web ed il domain data model degli ACTA Services. Quest'ultimo non rappresenta altro che una vista sul concettuale che aggrega le classi in un modello più funzionale all'interazione a servizi e nasconde alla ECM application entità strettamente dipendenti dal business di ACTA Web.

Pertanto gli utenti di una ECM application che interagisca a servizi con il sistema ACTA invocano servizi su oggetti del domain data model degli ACTA Services, i servizi traducono a loro volta le operazioni sugli oggetti di questo domain model in operazioni sugli oggetti del modello UML di ACTA preservando quindi tutta la logica di business descritta negli algoritmi degli use case di ACTA Web (con le precisazioni di cui al par. **Errore. L'origine riferimento non è stata trovata.**).

### **3 Modello dei Servizi**

#### **3.1 Attori Principali**

Gli attori coinvolti nell'interazione sono la ECM client application, ACTA repository come service endpoint e l'utente fisico che opera sul client.

Le azioni dell'utente si ripercuotono sulla ECM application e sul repository, le policy di accesso che ACTA stabilisce per l'accesso ai singoli oggetti del repository (altrimenti detto 'controllo di accesso a grana fine') richiedono la disponibilità dell'identità dell'utente, inoltre tale identità è anche necessaria per poter tracciare le operazioni effettuate (audit trail).

Per poter realizzare questo scenario occorre che:

- la ECM application, in qualità di originatore dell'interazione, autentichi l'utente con meccanismi di autenticazione e verifica delle credenziali,
- l'identità dell'utente sia propagata sino ad ACTA repository,
- ACTA repository autorizzi l'operazione sugli oggetti comparando il profilo dell'utente con le security policy di accesso.

#### **3.2**



## 3.2 Quadro generale dei servizi

Gli ACTA Services permettono di operare sugli oggetti del domain data model, tale modello è definito in accordo alle entità definite in [A3] per gerarchia delle classi ed attributi. Per consentire la definizione di servizi che operino su oggetti omogenei le classi del modello sono derivate da 4 tipologie base Policy, Relationship, Folder e Document, significative per un Content Management Repository. Pertanto i servizi, catalogati nelle classi che seguono, vanno ad offrire operazioni CRUD (Create, Retrieve, Update and Delete) sugli oggetti delle classi base e su quelli da essi derivati.

### 3.2.1 Repository Services

I servizi di questa classe permettono ai client di ACTA di recuperare informazioni quali:

- i repository disponibili su di un ACTA service endpoint,
- i metadati (Repository Name e Repository Identity) necessari a individuare l'Ente attestato su quel repository ed a riferire univocamente al repository nelle operazioni CRUD degli ACTA Services,
- le configurazioni applicative del repository.

### 3.2.2 Object Services

Gli ACTA Services offrono operazioni CRUD per l'oggetto coinvolto identificato dal proprio ID (object ID), per ognuna delle operazioni elementari si distinguono:

- servizi di creazione di oggetti come createDocument, createFolder, createRelationships restituzione dell'identificativo univoco;
- servizi di lettura delle proprietà degli oggetti ed, opzionalmente, delle operazioni permesse su di essi;
- servizi di aggiornamento delle proprietà degli oggetti, per quelle a valore multiplo occorre sostituire l'intera lista di valori, le proprietà read-only non sono aggiornabili dai client;
- supporto all' "optimistic locking" per mezzo della proprietà ChangeToken, si tratta di un attributo generato dal repository, e da questo dipendente, che permette a quest'ultimo di verificare in fase di update se l'oggetto è stato aggiornato rispetto alla precedente lettura;
- servizi di creazione, lettura per mezzo dell'identificativo dell'oggetto Document di appartenenza, se un contenuto è già stato associato ad un Document al momento della creazione dello stream il contenuto è sovrascritto.

Non tutte le operazioni elencate sono sempre possibili per ogni istanza degli oggetti del modello, ciò non solo per le limitazioni imposte dai tipi di oggetti definiti nel modello (vedere [A4], paragrafo "Oggetti del modello") ma anche per le policy di gestione dell'archivio che modificano nel tempo le operazioni possibili sulle istanze (trasferimenti da corrente a deposito, attività collaborative sui documenti ed altri processi propri del ciclo di vita dei contenuti).

Sebbene un content stream possa apparire essere una proprietà di un oggetto Document in realtà i servizi che restituiscono le proprietà dell'oggetto, o permettono di ricercare per mezzo di queste, non restituiscono un content stream, invece la modifica di un content stream di un Document è considerata modifica al Document.

Per gli oggetti derivati dalla classe base Relationship le operazioni CRUD richiedono che il client mantenga l'integrità referenziale con gli oggetti source e target della relazione.

### 3.2.3 Navigation Services

Per gli oggetti della classe base Folder, oltre agli object services di cui sopra, sono definiti servizi di navigazione della gerarchia di oggetti del repository secondo la struttura descritta in [A4] nel paragrafo “Repository”.

Gli ACTA Services getChildren, getDescendants, getFolderParent e getObjectParents permettono di navigare attraverso i descendants, ancestors, children e parents di un oggetto. Sono disponibili funzionalità di paging, utili per scorrere folder di dimensioni considerevoli, e di attraversamento di un sotto albero sino al livello di profondità richiesto, lo schema di attraversamento non è definibile dal client.

Anche per gli oggetti della classe base Relationship sono definiti servizi di navigazione, questi permettono di individuare tutte le relazioni, o loro sottoinsieme, aventi un determinato oggetto come source o target.

### 3.2.4 Multi-filing Services

Per gli oggetti della classe base Folder, oltre agli object services di cui sopra, sono definiti servizi di mantenimento della gerarchia di oggetti del repository secondo la struttura descritta in [A4] nel paragrafo “Repository”. Gli ACTA Services permettono di creare e rimuovere associazioni di un oggetto derivato dalla classe base Document ad uno o più oggetti (si supporta il multi-filing) derivati dalla classe base Folder.

Invece gli oggetti derivati dalla classe base Folder non possono essere associati a più di un Folder, causa il vincolo che impone di avere un solo padre, e ad un oggetto della propria discendenza per il vincolo di ciclicità dell'albero, per questo è stato previsto un servizio specifico.

Inoltre è previsto un servizio di cancellazione di un albero di oggetti a partire da una radice rappresentata da un oggetto Folder intermedio.

### 3.2.5 Management Services

Gli ACTA Services permettono di operare sugli oggetti del modello di riferimento definito in [A4] paragrafo “Concetti generali” ed in aggiunta su entità di business non direttamente sotto il controllo di tale modello ma comunque necessarie per l'ERMS. Si tratta delle entità del modulo BKO di ACTA necessarie per individuare i Principal delle ACE, oppure entità associate ad oggetti derivati da Document e Folder funzionali al modulo SMS.

### **3.3 Descrizione delle singole operazioni**

#### **3.3.1 Repository Services**

##### **3.3.1.1 getRepositories**

Si tratta dell'unico servizio non specifico per un singolo repository, permette ai client di ACTA di individuare il repository su cui operare all'interno di una lista dei repository disponibili ad un service endpoint.

La localizzazione del repository avviene per mezzo dei dati caratteristici dell'ente: codice, denominazione, etc.

##### **3.3.1.2 getRepositoryInfo**

Servizio utile per ottenere informazioni sulle funzionalità implementate nel repository localizzato per mezzo del servizio getRepositories, e l'informazione di RootFolder Identification indispensabile per iniziare la navigazione del repository a partire dalla radice.

Per un ACTA service endpoint le funzionalità implementate negli ACTA repositories attestati sono le stesse.

### 3.3.2 Object Services

#### 3.3.2.1 createDocument

Servizio necessario per creare un oggetto di un tipo derivato dalla classe base Document ed eventualmente creare una gerarchia padre-figlio con un oggetto della classe base Folder. Opzionalmente è possibile associare un contenuto all'oggetto se previsto.

Il tipo dell'oggetto da creare deve essere compatibile con il tipo dell'oggetto Folder padre.

#### 3.3.2.2 createAssociativeDocument

Servizio necessario per creare un oggetto di un tipo derivato dalla classe base Document e di creare una gerarchia padre-figlio con un oggetto della classe base Folder attraverso la creazione di un oggetto 'associativo' specificato come parametro nella chiamata al servizio. Rappresenta un'estensione del servizio createDocument.

#### 3.3.2.3 createFolder

Servizio necessario per creare un oggetto di un tipo derivato dalla classe base Folder ed eventualmente creare una gerarchia padre-figlio con un oggetto della classe base Folder.

Il tipo dell'oggetto da creare deve essere compatibile con il tipo dell'oggetto Folder padre.

#### 3.3.2.4 createRelationship

Servizio necessario per creare una relationship fra due oggetti, il source ed il target, del repository.

#### 3.3.2.5 getProperties

Il servizio restituisce una lista di proprietà dell'oggetto specificato come parametro, scelte fra quelle selezionate.

#### 3.3.2.6 updateProperties

Il servizio permette di modificare una lista di proprietà dell'oggetto specificato come parametro, scelte fra quelle selezionate. Se alcune delle proprietà definite per l'oggetto da modificare non sono specificate nella chiamata al servizio l'operazione di modifica è eseguita come se queste proprietà fossero state incluse ed impostate al loro valore attuale.

Per rendere possibili le verifiche collegate all' "optimistic locking" il client deve obbligatoriamente specificare il parametro change token.

#### 3.3.2.7 moveObject

Il servizio permette di modificare la relazione padre-figlio dell'oggetto specificato come parametro muovendo l'oggetto da un source Folder ad un target Folder.

Il tipo dell'oggetto da movimentare deve essere compatibile con il tipo dell'oggetto Folder target. Per rendere possibili le verifiche collegate all' "optimistic locking" il client deve obbligatoriamente specificare il parametro change token.

#### 3.3.2.8 getContentStream

Il servizio restituisce il 'content-stream' associato all'oggetto specificato come parametro, deve essere un oggetto di tipo derivato dalla classe base Document.

Il client deve poter richiedere un sottoinsieme del content-stream.

### 3.3.3 Multi-filing Services

#### 3.3.3.1 addAssociativeObjectToFolder

Il servizio permette di aggiungere una relazione padre-figlio all'oggetto specificato come parametro associandolo ad un oggetto della classe base Folder attraverso la creazione di un oggetto 'associativo' specificato come parametro. Rappresenta un'estensione del servizio addObjectToFolder.

### 3.3.4 Relationships Services

#### 3.3.4.1 getRelationships

Il servizio restituisce una lista di oggetti derivati dalla classe base Relationship dove l'oggetto specificato come parametro compare come source, target o entrambe nella relazione.

Il servizio permette di indicare il livello di profondità nella gerarchia tipo/sottotipo da cui selezionare i tipi derivati dalla classe base Relationship da includere nella lista.

### 3.3.5 Navigation Services

#### 3.3.5.1 getDescendants [DEPRECATO]

Il servizio restituisce una lista di oggetti in relazione padre-figlio (successore) con l'oggetto della classe base Folder specificato come parametro, per uno o più livelli di profondità nell'albero di oggetti del repository.

Per ogni oggetto vengono restituite solo le proprietà selezionate oppure gli oggetti il cui tipo base è una delle classi selezionate.

Gli oggetti della lista sono restituiti secondo un ordinamento dipendente dall'implementazione ma consistente (ad esempio, per ogni livello di profondità, prima tutti gli oggetti il cui tipo base è Folder poi gli oggetti figli) analogamente l'algoritmo di navigazione del repository è dipendente dall'implementazione.

#### 3.3.5.2 getChildren

Il servizio restituisce una lista di oggetti in relazione padre-figlio con l'oggetto della classe base Folder specificato come parametro, per un solo livello di profondità nell'albero di oggetti del repository.

Per ogni oggetto vengono restituite solo le proprietà selezionate oppure gli oggetti il cui tipo base è una delle classi selezionate.

#### 3.3.5.3 getFolderParent

Il servizio restituisce una lista di oggetti in relazione figlio-padre (antenato) con l'oggetto della classe base Folder specificato come parametro. La gerarchia nell'albero di oggetti del repository viene risalita sino alla radice se richiesto altrimenti si risale di un solo livello.

Per ogni oggetto devono essere restituite le proprietà di identità dell'oggetto e del proprio padre per permettere un eventuale riordinamento da parte del client.

#### 3.3.5.4 getObjectParents

Il servizio restituisce una lista di oggetti in relazione figlio-padre (antenato) con l'oggetto di una classe base diversa da Folder specificato come parametro.

Per ogni oggetto devono essere restituite le proprietà di identità dell'oggetto e del proprio padre per permettere un eventuale riordinamento da parte del client.



### 3.3.6 Management Services

#### 3.3.6.1 Gestione contenuti

I servizi elencati nel seguito sono riservati per sviluppi futuri, pertanto non sono ulteriormente dettagliati nei documenti di specifica, rappresentano quanto è possibile fare ad integrazione dei servizi attuali. E' possibile che al momento della loro definitiva definizione vengano modificati nel numero e nella semantica.

##### 3.3.6.1.1 addAnnotazioni

##### 3.3.6.1.2 getVitalRecordCode

Questa operazione permette di ottenere l'elenco dei VitalRecordCode validi per un repository.

#### 3.3.6.2 Back office

Alcuni dei servizi elencati nel seguito sono riservati per sviluppi futuri, pertanto non sono ulteriormente dettagliati nei documenti di specifica, rappresentano quanto è possibile fare ad integrazione dei servizi attuali. E' possibile che al momento della loro definitiva definizione vengano modificati nel numero e nella semantica.

##### 3.3.6.2.1 dettaglioAOO

Questa operazione permette di ottenere informazioni di dettaglio di una AOO dato l'identificativo dell'istanza della classe.

##### 3.3.6.2.2 dettaglioStruttura

Questa operazione permette di ottenere informazioni di dettaglio di una Struttura dato l'identificativo dell'istanza della classe.

#### getPrincipalExt

Questa operazione permette di ottenere l'elenco dei 'principal' che possono operare sul repository dato il codice fiscale di un utente autenticato dal sistema fruitore. E' un codice assegnato ad un fruitore dei servizi per una durata definita (normalmente 1 h).

4

## 4 Algoritmi del modulo ACARIS

Nel seguito i riferimenti agli oggetti del domain data model degli ACTA Services, come descritto in [A4], sono effettuati per mezzo dei nomi definiti negli schema di riferimento ed in particolare:

- **complexType**  

```
<xs:complexType name="ObjectTypeId">
  <xs:complexContent>
    <xs:sequence>
      <xs:element name="PropertyId" type="PropertyType"/>
      ...
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>
```
- **simpleType**  

```
<xs:simpleType name="PropertyType">
  ...
</xs:simpleType>
```

Il nome utilizzato è quello specificato per l'attributo *name* dei tipi complessi o semplici definiti negli schema ACARIS-Core.xsd o ACARIS-Messaging.xsd. Può trattarsi a seconda dei casi del nome della classe dell'oggetto del modello (attributo *ObjectTypeId*) oppure il nome della property (attributo *PropertyId*) oppure il tipo della property (attributo *PropertyType*).

### 4.1 ACARIS\_A01\_Property\_Filters

#### 4.1.1 Finalità dell'algoritmo

Fornire ai servizi definiti per gli oggetti del domain data model un elenco di proprietà i cui valori debbono essere restituiti dall'invocazione dell'operazione.

#### 4.1.2 Descrizione

Alcune operazioni dei servizi definiti per gli oggetti del domain data model richiedono un elenco, opzionale, di proprietà per le quali il client vuole conoscere il valore associato. Lo scopo del filtro è quello di permettere al client di specificare un sottoinsieme delle proprietà, fra quelle definite per l'oggetto, per cui questi è interessato altrimenti in output verrebbero restituiti i valori di tutte.

I valori ammessi sono i seguenti:

- filtro non specificato (o filtro di default), l'insieme di proprietà per cui restituire un valore è determinato da ACTA repository operazione per operazione,
- filtro costituito da un elenco di nomi di proprietà validi ovvero da nomi elencati nei *PropertyName* dell'oggetto specificato nell'operazione,
- restituzione di tutti i valori ammessi (\*).

Gli allegati [A7] ed [A8] riportano per ogni operazione e per ogni oggetto del domain data model

l'insieme delle proprietà ammissibili nelle specifica di un filtro. In particolare:

- Operazione / Property list

La colonna indica con un contrassegno ('X') tutte le proprietà ammissibili nel filtro quando questo è specificato. PropertyName non contrassegnate ma specificate nel filtro non verranno restituite. La proprietà objectId deve sempre essere restituita anche se non specificata.

Per alcune operazioni ACTA repository può definire un valore massimo ammesso di proprietà da restituire che se superato impone la restituzione delle sole proprietà di default quindi come se il filtro non fosse stato specificato.

- Operazione / Null

La colonna indica con un contrassegno ('X') tutte le proprietà restituite quando il filtro non è specificato.

- Operazione / All

La colonna indica con un contrassegno ('X') tutte le proprietà restituite quando il filtro è specificato nella forma di '\*'.

La specifica del filtro avviene per mezzo del riferimento agli oggetti del domain data model degli ACTA Services, come descritto in [A4], utilizzando i nomi definiti negli schema di riferimento ed in particolare:

```
<xs:complexType name="PropertyFilterType">
  <xs:sequence>
    <xs:element name="filterType" type="tns:enumPropertyFilter"
      minOccurs="1" maxOccurs="1" />
    <xs:element name="propertyList" type="tns:QueryNameType"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

Il *filterType* indica la tipologia di filtro utilizzata fra property list, null e all. Se property list occorre valorizzare la *propertyList* nel seguente modo:

```
<xs:complexType name="QueryNameType">
  <xs:sequence>
    <xs:element name="className" type="tns:enumObjectType" />
    <xs:element name="propertyName" type="tns:string" />
  </xs:sequence>
</xs:complexType>
```

La coppia *className*, *propertyName* utilizzata è quella specificato per i tipi definiti negli schema ACARIS-Core.xsd. Nel caso in cui sia valorizzata la *propertyList* con *filterType* null o all questa non viene considerata, analogamente se *filterType* è property list e la lista è vuota si utilizza il filtro per *filterType* null.

## 4.2 ACARIS\_A02\_Paging

### 4.2.1 Finalità dell'algoritmo

Fornire ai servizi che restituiscono collezioni di oggetti il supporto al paging dei risultati.

### 4.2.2 Parametri in input

L'algoritmo riceve in input:

- `maxItems`: numero massimo di oggetti che l'operazione restituisce a seguito di una invocazione,
- `skipCount`: numero di risultati potenziali da paginare prima di restituire nuovi oggetti.

### 4.2.3 Output

L'algoritmo restituisce:

- `hasMoreItems`: indicatore della presenza nel repository di ulteriori risultati per l'invocazione di servizio effettuata e non ancora restituiti per le limitazioni imposte dal paging.

### 4.2.4 Descrizione

Le operazioni che restituiscono collezioni di oggetti a cardinalità, potenzialmente, elevata offrono il supporto al paging dei risultati secondo le regole seguenti.

Properties utilizzate per i controlli da parte del sistema e contenute nella vista *VariabiliEnteView*:

- *`limiteMassimoRisultatiQueryAcaris`*
- *`limiteMassimoRisultatiNavigazioneAcaris`*
- *`limiteMassimoRisultatiPaginaAcaris`*
- *`paginareAncheSeRichiestiSoloIdentificatori`*

#### Il fruitore non valorizza `maxItems`

il sistema determina il numero di oggetti da restituire

se superiore al limite massimo (*`limiteMassimoRisultatiQueryAcaris`* per il servizio *query* e *`limiteMassimoRisultatiNavigazioneAcaris`* per i servizi di navigazione) viene sollevata eccezione chiedendo che venga raffinato il criterio di ricerca (nel caso di *query*) oppure (in caso di navigazione) che si specifichi meglio il type (tramite LIST) o si effettui, in alternativa, una ricerca

se inferiore o uguale al limite massimo i risultati sono da restituire in forma paginata e la dimensione di ogni pagina sarà quella stabilita *`limiteMassimoRisultatiPaginaAcaris`*

se vengono richiesti solo gli identificatori il sistema li restituisce paginati o meno (con pagine della dimensione *`limiteMassimoRisultatiPaginaAcaris`*) a seconda della valorizzazione della property *`paginareAncheSeRichiestiSoloIdentificatori`*

#### Il fruitore valorizza maxItems

il sistema confronta il valore di maxItems con quello di *limiteMassimoRisultatiPaginaAcaris*

se superiore viene sollevata eccezione richiedendo che il valore sia inferiore o uguale a quello di *limiteMassimoRisultatiPaginaAcaris*

il sistema determina il numero di oggetti da restituire

se superiore al limite massimo ((*limiteMassimoRisultatiQueryAcaris* per il servizio *query* e *limiteMassimoRisultatiNavigazioneAcaris* per i servizi di navigazione) viene sollevata eccezione chiedendo che venga raffinato il criterio di ricerca (nel caso di *query*) oppure (in caso di navigazione) che isi specifichi meglio il type (tramite LIST) o si effettui, in alternativa, una ricerca

se inferiore o uguale al limite massimo i risultati sono da restituire in forma paginata suddividendoli per pagine della dimensione stabilita in maxItems

se vengono richiesti solo gli identificatori il sistema li restituisce paginati o meno (con pagine della dimensione di maxItems) a seconda della valorizzazione di *paginareAncheSeRichiestiSoloIdentificatori*

#### PropertyFilter e richiesta di tutti gli identificatori senza paginare

la richiesta di tutti gli identificatori può essere effettuata utilizzando il propertyFilter di tipo LIST

La vista *VariabiliEnteView* può essere utilizzata come target nell'invocazione del servizio *query* sul modulo *BackOfficeService*. Per un approfondimento sull'utilizzo del servizio *query* si veda l'apposita sezione dell'allegato tecnico.

Viene riportata la struttura (ricavabile anche con il servizio *getQueryableObjectMetadata*)

Entità	className	proprietà	selezionabile	filtrabile	pfALL	pfNONE	pfLIST
VariabiliEnteView	VariabiliEnteView	dbKey	S	S	S	S	S
VariabiliEnteView	VariabiliEnteView	idEnte	S	S	S	S	S
VariabiliEnteView	VariabiliEnteView	durataSessionePrincipal	S	N	S	S	S
VariabiliEnteView	VariabiliEnteView	limiteMassimoRisultatiQueryAcaris	S	N	S	S	S
VariabiliEnteView	VariabiliEnteView	limiteMassimoRisultatiNavigazioneAcaris	S	N	S	S	S
VariabiliEnteView	VariabiliEnteView	limiteMassimoRisultatiPaginaAcaris	S	N	S	S	S
VariabiliEnteView	VariabiliEnteView	paginareAncheSeRichiestiSoloIdentificatori	S	N	S	S	S
VariabiliEnteView	RepositoryType	idRepository	S	S	S	S	S

## 4.3 ACARIS\_A03\_Change\_Tokens

### 4.3.1 Finalità dell'algoritmo

Fornire al client supporto nell'implementare l' "optimistic locking" nelle operazioni di update per garantire assenza di conflitto fra utenti diversi che operano sul repository.

#### 4.3.2 Parametri in input

L'algoritmo riceve in input:

- ChangeToken: attributo generato dal repository, e da questo dipendente, da utilizzare per una operazione di aggiornamento di un oggetto.

#### 4.3.3 Output

L'algoritmo restituisce:

- ChangeToken: attributo generato dal repository ottenuto a seguito dell'invocazione di una operazione di lettura di un oggetto.

#### 4.3.4 Descrizione

Gli ACTA Services danno supporto all'implementazione dell' "optimistic locking" per mezzo della proprietà ChangeToken, si tratta di un attributo generato dal repository, e da questo dipendente, che permette a quest'ultimo di verificare in fase di update se l'oggetto è stato aggiornato rispetto alla precedente lettura.

### 4.4 ACARIS\_A04\_Crea\_PrincipalId

#### 4.4.1 Finalità dell'algoritmo

Fornire ai servizi definiti per gli oggetti del domain data model un identificatore univoco del principal (**chi** può fare **cosa** di una Access Control Entry), da utilizzare nelle operazioni di accesso agli oggetti del repository. La rappresentazione lessicale del PrincipalId è una stringa opaca per il client, generata dal repository, read-only e locale all'Ente di appartenenza del repository..

#### 4.4.2 Parametri in input

L'algoritmo riceve in input i dati specifici dell'identità dell'utente per la policy di accesso richiesta.

#### 4.4.3 Output

L'algoritmo restituisce in output il PrincipalId dell'utente.

#### 4.4.4 Descrizione

L'associazione fra PrincipalId restituito dal servizio e riferimenti agli oggetti definiti internamente al repository è così definita:

- Principal: utenti che utilizzano le policy di accesso ad ACTA individuati dall'identificativo dell'identità dell'utente, quindi PrincipalId composto da:
  - ActaACEPropertiesType come identificativo della classe
  - Identità.id come identificativo dell'utente

L'algoritmo di costruzione della stringa opaca è lasciato all'implementazione mentre gli elementi indicati permettono di distinguere ogni stringa dalla successiva rendendola perciò univoca.

Ogni PrincipalId permette di avere accesso ad oggetti ed operazioni diverse governate dalle policy di accesso implementate in ACTA, il PrincipalId è creato dai management services di back office invocati dal client per mezzo dell'identità dell'utente autenticato dall'ECM application.

Per soddisfare i requisiti di security di ACTA occorre che il PrincipalId non sia 'clonabile' dall'ECM application pertanto oltre alle caratteristiche di opacità occorre cifrare la stringa ed implementare un meccanismo di limitazione della validità temporale della stringa (scope di sessione) a tutela del servizio.

## **4.5 ACARIS\_A05\_Decodifica\_PrincipalId**

### **4.5.1 Finalità dell'algoritmo**

Permettere ai servizi definiti per gli oggetti del domain data model di individuare nel PrincipalId i dati specifici dell'identità dell'utente e la policy di accesso richiesta.

### **4.5.2 Parametri in input**

L'algoritmo riceve in input il PrincipalId dell'utente ed il RepositoryId per cui è stato richiesto il servizio.

### **4.5.3 Output**

L'algoritmo restituisce in output i dati specifici dell'identità dell'utente e la policy di accesso richiesta.

### **4.5.4 Descrizione**

L'associazione fra PrincipalId ricevuto in input dal servizio e dati specifici è riportata nel paragrafo 4.4.4.

Il sistema verifica che il PrincipalId ricevuto in input dal servizio non sia scaduto ovvero siano stati superati i termini di validità temporale.

Se sono stati superati i termini il sistema genera una AcarisException con il messaggio SERGEN-E001.

Il sistema determina per l'Utente associato al PrincipalId la classe Ente di appartenenza, quindi verifica che l'Ente corrisponda all'Ente individuato dal RepositoryId.

Se gli enti non coincidono il sistema genera una AcarisException con il messaggio SERGEN-E002.

## **4.6 ACARIS\_A06\_Crea\_ObjectId**

### **4.6.1 Finalità dell'algoritmo**

Fornire ai servizi definiti per gli oggetti del domain data model un identificatore univoco dell'istanza dell'oggetto in ACTA repository, la proprietà, obbligatoria per tutte le classi del domain data model, viene valorizzata per mezzo dei riferimenti univoci agli oggetti definiti internamente al repository.

La rappresentazione lessicale dell'ObjectId è una stringa opaca per il client, generata dal repository, read-only ed univoca all'interno di un repository (archivio di proprietà dello stesso Ente).

#### 4.6.2 Parametri in input

L'algoritmo riceve in input l'ObjectTypeId della classe dell'oggetto.

#### 4.6.3 Output

L'algoritmo restituisce in output l'ObjectId dell'istanza dell'oggetto della classe.

#### 4.6.4 Descrizione

La relazione fra ObjectId restituito dal servizio e riferimenti agli oggetti definiti internamente al repository è riportata nella seguente tabella:

<b>ObjectTypeId</b> <i>identificatore della classe ...</i>	<b>ObjectId</b> <i>associabile a ...</i>
ContenutoFisicoPropertiesType	ContenutoFisicoPropertiesType + identificatore univoco nel repository dell'istanza di ContenutoFisico
DocumentoDBPropertiesType	DocumentoDBPropertiesType + Documento.id <sup>UUID</sup>
DocumentoRegistroPropertiesType	DocumentoRegistroPropertiesType + Documento.id <sup>UUID</sup>
DocumentoSemplicePropertiesType	DocumentoSemplicePropertiesType + Documento.id <sup>UUID</sup>
ClipsMetallicaPropertiesType	ClipsMetallicaPropertiesType + identificatore univoco nel repository dell'istanza di ClipsMetallica
GruppoAllegatiPropertiesType	GruppoAllegatiPropertiesType + identificatore univoco nel repository dell'istanza di GruppoAllegati
TitolarioPropertiesType	TitolarioPropertiesType + identificatore univoco nel repository dell'istanza di Titolario
ClassificazionePropertiesType	ClassificazionePropertiesType + Classificazione.id <sup>UUID</sup>
VocePropertiesType	VocePropertiesType + identificatore univoco nel repository dell'istanza di Voce
FascicoloTemporaneoPropertiesType	FascicoloTemporaneoPropertiesType + identificatore univoco nel repository dell'istanza di FascicoloTemporaneo
DocumentoFisicoPropertiesType	DocumentoFisicoPropertiesType + identificatore univoco nel repository dell'istanza di DocumentoFisico
SottofascicoloPropertiesType	SottofascicoloPropertiesType + Aggregazione.id <sup>UUID</sup>
DossierPropertiesType	DossierPropertiesType + Aggregazione.id <sup>UUID</sup>
VolumeSerieFascicoliPropertiesType	VolumeSerieFascicoliPropertiesType + Aggregazione.id <sup>UUID</sup>
VolumeSerieTipologicaDocumentiPropertiesType	VolumeSerieTipologicaDocumentiPropertiesType + Aggregazione.id <sup>UUID</sup>
VolumeFascicoliPropertiesType	VolumeFascicoliPropertiesType + Aggregazione.id <sup>UUID</sup>
VolumeSottofascicoliPropertiesType	VolumeSottofascicoliPropertiesType +



	Aggregazione.id <sup>UUID</sup>
SerieDossierPropertiesType	SerieDossierPropertiesType + Aggregazione.id <sup>UUID</sup>
SerieTipologicaDocumentiPropertiesType	SerieTipologicaDocumentiPropertiesType + Aggregazione.id <sup>UUID</sup>
SerieFascicoliPropertiesType	SerieFascicoliPropertiesType + Aggregazione.id <sup>UUID</sup>
FascicoloRealeEreditatoPropertiesType	FascicoloRealeEreditatoPropertiesType + Aggregazione.id <sup>UUID</sup>
FascicoloRealeLiberoPropertiesType	FascicoloRealeLiberoPropertiesType + Aggregazione.id <sup>UUID</sup>
FascicoloRealeContinuoPropertiesType	FascicoloRealeContinuoPropertiesType + Aggregazione.id <sup>UUID</sup>
FascicoloRealeLegislaturaPropertiesType	FascicoloRealeLegislaturaPropertiesType + Aggregazione.id <sup>UUID</sup>
FascicoloRealeAnnualePropertiesType	FascicoloRealeAnnualePropertiesType + Aggregazione.id <sup>UUID</sup>
DocumentAssociationPropertiesType	DocumentAssociationPropertiesType + identificatore univoco nel repository della relazione (es. associabile a GruppoAllegati e Classificazione.id, oppure ClipsMetallica e Classificazione.id)
HistoryModificheTecnichePropertiesType	HistoryModificheTecnichePropertiesType + identificatore univoco nel repository dell'istanza di HistoryModificheTecniche
DocumentCompositionPropertiesType	DocumentCompositionPropertiesType + identificatore univoco nel repository della relazione (es. associabile a Documento.id + DocumentoFisico.progressivoPerDocumento)
HistoryVecchieVersioniPropertiesType	HistoryVecchieVersioniPropertiesType + identificatore univoco nel repository dell'istanza di HistoryVecchieVersioni

La tabella indica come comporre un ObjectId a seguito della creazione di un oggetto della classe ObjectTypeId, sono ammissibili solo ObjectTypeId di oggetti creabili dal client oppure creati dal sistema e restituiti a seguito di operazioni di navigazione.

Pertanto nella colonna ObjectId il segno + è da intendersi come composizione di elementi che partecipano alla costruzione della stringa opaca, l'algoritmo di costruzione della stringa opaca è lasciato all'implementazione mentre gli elementi indicati permettono di distinguere ogni stringa dalla successiva rendendola perciò univoca.

Gli ObjectId composti da riferimenti univoci di tipo UUID possono essere resi persistenti nelle ECM client application mentre gli altri riferimenti sono creati per mezzo di identificativi informatici che non soddisfano il requisito di unicità ed indipendenza dal contesto del sistema informativo e dal dataset nel tempo.

#### Ne consegue che:

- la persistenza è garantita per quanto riguarda gli oggetti che fanno riferimento ad un UUID
- per gli altri identificativi che sono generati a runtime e che hanno, quindi, scope di sessione

(dell'applicazione client), nel caso di persistenza sul client non viene garantita l'univocità nei richiami successivi

#### **4.6.5 Applicazione generale delle regole di valorizzazione degli ObjectId**

Allo scopo di gestire la valorizzazione dell'ObjectId anche con riferimento alle classi non espressamente elencate nei paragrafi precedenti, viene espressa una regola generale che si articola nei seguenti punti:

1. ogni oggetto censito nel sistema come oggetto di tipo Properties (ereditato direttamente o indirettamente) espone una property di tipo ObjectIdType che rappresenta l'identificatore opaco generato dal sistema
2. la valorizzazione viene eseguita dal sistema concatenando (con modalità lasciate alla progettazione della soluzione) le seguenti informazioni:
  - className definito nell'object model
  - identificatore univoco

L'identificatore univoco viene così determinato:

1. UUID per le classi che hanno questo metadato (indipendentemente dal repository)
2. db key per le classi che non hanno un metadato UUID e risiedono su dbms
3. UID node di Index per le classi che non hanno un metadato UUID e risiedono su Index

### **4.7 ACARIS\_A07\_Decodifica\_ObjectId**

#### **4.7.1 Finalità dell'algoritmo**

Permettere ai servizi definiti per gli oggetti del domain data model di tradurre l'ObjectId in riferimenti univoci agli oggetti definiti internamente al repository.

#### **4.7.2 Parametri in input**

L'algoritmo riceve in input l'ObjectId dell'istanza dell'oggetto della classe.

#### **4.7.3 Output**

L'algoritmo restituisce in output l'ObjectId della classe dell'oggetto ed i riferimenti agli oggetti del repository.

#### **4.7.4 Descrizione**

L'associazione fra ObjectId ricevuto in input dal servizio e riferimenti agli oggetti definiti internamente al repository è riportata nella tabella di paragrafo 4.6.4.

La tabella indica come decomporre un ObjectId creato dal sistema ed utilizzato, ad esempio, dal client per operazioni di navigazione.

#### **4.7.5 Regole per l'accesso agli oggetti in operazioni di navigazione e lettura del dettaglio degli oggetti**

#### 4.7.5.1 Oggetti riconducibili al modulo Acta GCO

In GCO la regole per la visibilità delle informazioni da parte di un principal – in questo caso un'identità con un profilo collocata organizzativamente in modo specifico – possono essere così riepilogate:

- Voci. Sono visibili le sole voci in stato ATTIVA e DISATTIVA
- Fascicoli (tutte le tipologie). Sono visibili i fascicoli:
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
  - Per cui è vera la condizione fascicolo.archiviocorrente = VERO
  - Per cui l'identità possiede una ACL di lettura (R=Read)
  - Per cui l'identità è abilitata al caso d'uso GCOGEN014 se fascicolo.riservato = VERO
  - Per cui l'identità è abilitata al caso d'uso GCOGEN012 se fascicolo.sensibile = VERO
- Serie (tutte le tipologie). Sono visibili le serie:
  - Che NON sono in stato (DISATTIVA, CHIUSA IN DEPOSITO)
  - Per cui è vera la condizione serie.archiviocorrente = VERO
  - Per cui l'identità possiede una ACL di lettura (R=Read)
  - Per cui l'identità è abilitata al caso d'uso GCOGEN014 se serie.riservato = VERO
  - Per cui l'identità è abilitata al caso d'uso GCOGEN012 se serie.sensibile = VERO
- Dossier. Sono visibili i dossier:
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
  - Per cui è vera la condizione dossier.archiviocorrente = VERO
  - Per cui l'identità possiede una ACL di lettura (R=Read)
  - Per cui l'identità è abilitata al caso d'uso GCOGEN014 se dossier.riservato = VERO
  - Per cui l'identità è abilitata al caso d'uso GCOGEN012 se dossier.sensibile = VERO
- Sottofascicoli. Sono visibili i sottofascicoli:
  - Sono rispettate le condizioni di visibilità del fascicolo padre del sottofascicolo
  - Per cui l'identità è abilitata al caso d'uso GCOGEN014 se sottofascicolo.riservato = VERO

- Per cui l'identità è abilitata al caso d'uso GCOGEN012 se sottofascicolo.sensibile = VERO
- Volume di serie di fascicoli. Sono visibili i volumi:
  - Per cui sono rispettate le condizioni di visibilità della serie padre del volume
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
- Volume di serie di dossier Sono visibili i volumi:
  - Per cui sono rispettate le condizioni di visibilità della serie padre del volume
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
- Volume di serie di documenti. Sono visibili i volumi:
  - Per cui sono rispettate le condizioni di visibilità della serie padre del volume
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
- Volume di fascicolo. Sono visibili i volumi:
  - Per cui sono rispettate le condizioni di visibilità del fascicolo padre del volume
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
- Volume di sottofascicolo. Sono visibili i volumi:
  - Per cui sono rispettate le condizioni di visibilità del fascicolo padre del sottofascicolo in cui è posizionato il volume
  - Che NON sono in stato (CHIUSO IN DEPOSITO, SCARTATO)
- Documenti. Sono visibili i documenti:
  - Per cui sono rispettate le condizioni di visibilità della struttura aggregativa padre che li contiene
  - I documenti possono essere presenti all'interno di:
    - Fascicoli
    - Sottofascicoli
    - Serie di documenti
    - Dossier

- Volumi di serie di documenti
- Volumi di fascicolo
- Volumi di sottofascicolo

#### 4.7.5.2 Oggetti riconducibili al modulo Acta GSA

In GSA la regole per la visibilità delle informazioni da parte di un principal possono essere così riepilogate:

- In GSA non è significativa la collocazione dell'identità di accesso: chi accede a GSA non ha una visibilità in funzione della sua collocazione ma in funzione dell'Ente per cui l'operatore è censito
- Considerando la possibilità di avere più RaggruppamentiAOO per un solo Ente, in GSA è prevista la selezione del raggruppamento AOO – associati all'Ente cui appartiene l'operatore – e che determina i/il titolari/o da presentare all'operatore stesso
- A fronte della selezione del RaggruppamentoAOO non sono presenti altre limitazioni sulla visibilità degli oggetti gestibili da GSA.
  - Titolari. Tutti, in qualsiasi stato
  - Voci. Tutte, in qualsiasi stato
  - Serie. Tutte quelle che NON sono in stato <CHIUSA IN DEPOSITO>
  - Fascicolo Standard. Tutti, in qualsiasi stato.
  - Fascicolo Temporaneo. Tutti, in qualsiasi stato.
- Gli oggetti sopra elencati sono tutti e i soli gestibili/visibili nell'ambito di GSA che non prevede al suo interno la gestione di ulteriori strutture aggregative

**NB.** Nell'ambito dei servizi ACARIS, nell'attuale versione, non è presente alcuna discriminante in merito al RaggruppamentoAOO necessario per le gestione della visibilità del titolare (e di tutta la sua discendenza).

## 5 Tecnologie

L'esposizione dei servizi è in SOAP 1.1