Il sistema di autenticazione ed autorizzazione

####Not jet implemented/released####

Il server 24 introduce un nuovo modello di autenticazione ed autorizzazione rispetto a quanto avveniva sino al server 23 compreso. Questo per rendere più potente ma, al contempo, più semplice, la gestione delle autorizzazioni ed autenticazioni.
Sino ad ora, infatti, abbiamo visto che per avere una gestione utenti completa e corretta, con riflessi significativi anche sulle applicazioni, è comunque necessario avvalersi di un sistema di *Access Control List* che sia corredato da un meccanismo di profili adeguatamente ricco perché le applicazioni sappiano regolarsi di conseguenza. Sino ad ora il server è sempre stato relegato ad un compito passivo, accettando quel che le applicazioni richiedevano senza un'effettiva cognizione di cosa fosse opportuno fare o meno.

La premessa

Per meglio comprendere il perché delle variazioni apportate ed i vantaggi che si celano dietro di esse, facciamo prima un po' di storia. Com'è noto eXtraWay prende le mosse dal suo predecessore, HighWay, che si basava su un archivio degli utenti nei quali erano presenti un risicato numero di informazioni disponibili. Tra esse, oltre alla password che ne consentiva l'accesso, un livello d'accesso che avrebbe dovuto autorizzare o negare alcune attività e la definizione di un *mazzo di chiavi* in grado di consentire o negare l'accesso ad uno o più documenti. Questa pratica, per quanto interessante, non ha mai trovato effettiva diffusione.
Quando le applicazioni Web hanno soppiantato le soluzioni *Client-Server*, il meccanismo di accesso autenticato ed il rapporto 1:1 tra un Client ed un Server hanno perso completamente di significato. Quelli che sino ad allora venivano considerati Client sono divenuti una batteria di *Application Server* il cui scopo è quello di fornire ad un numero imprecisato di utenti, aventi ciascuno diritti propri ed ovviamente una propria identità che, chiaramente, non viene più gestita dal server. Ciò nonostante, era comunque opportuno che il server mantenesse un qualche controllo sull'accesso di tali *Application Server* così da garantirsi che qualche applicazione potesse maliziosamente collegarsi al server ad accedere ai dati da esso gestiti. Per tale ragione, il concetto di utente corredato da password è rimasto. Come detto, però, non parliamo più di rapporto 1:1 con l'applicazione Client ma di un rapporto con una batteria di *Application Server* che servono un numero indefinito di utenti, ciascuno con il proprio ID e con il proprio Ip Address di provenienza. Tali dati vengono quindi notificati al server perché esso possa compiere le proprie registrazioni con effettiva cognizione di chi ha svolto le operazioni e possa contestualmente compiere verifiche sull'Ip Address di provenienza per negare l'accesso dello stesso utente da postazioni differenziate.
Abbiamo quindi due tipologie di utenti: gli utenti con caratteristiche d'accesso e gli utenti generici.
Il server eXtraWay ha ereditato tutto questo da Highway limitandosi, per così dire, ad abbandonare l'archivio degli utenti e favore di un file XML[1] nel quale compiere la registrazione di ambo le categorie di utenti. Seguendo logiche che adesso sarebbe complesso spiegare, agli utenti d'accesso viene normalmente associata una password mentre a tutti gli altri la password associata, per quanto riportata nel file `xusers.xml` è di fatto nulla o non utilizzata. La password riportata in tale file per ciascun utente si ottiene con un particolare algoritmo disponibile tramite il modulo `xwpasswd`.
Nel file `xusers.xml` era quindi possibile indicare per quali utenti fosse ammesso l'accesso da Ip Address molteplici senza che questo comporti errore, condizione necessaria a garantire un corretto comportamento anche in condizioni che potremmo definire “particolari” ma che non vengono approfondite in questa sede.

Il nuovo sistema

Innanzitutto il nuovo sistema è concepito per fruire di un *provider* di autenticazione ed autorizzazione.
Esso, nell'accezione più completa, è un server LDAP che possa dirci se un utente è riconosciuto come valido se accoppiato ad una password e quali siano i profili[2] cui esso appartiene.
Per non imporre che ogni installazione, anche la più semplice, debba necessariamente prevedere un sistema LDAP esistente, un alternativo *provider* può semplicemente essere rappresentato da un sistema file di cui l'installazione dev'essere corredata.

Nota: Punto a sfavore di questa soluzione è se un'installazione gestisce diversi archivi e quindi diverse applicazioni, esse devono necessariamente rifarsi ad un unico *provider* di autenticazione ed autorizzazione, non è possibile diversificare il comportamento del server dipendentemente dall'archivio.

auth.properties (parte prima)

Iniziamo quindi dal file di *properties* che regola le modalità con le quali il server eXtraWay accede al *provider* prescelto. Esso si chiama auth.properties e va collocato nella directory /conf. In esso si trovano 2 raggruppamenti di informazioni che vedremo singolarmente.

Iniziamo a valutare il primo, quello che effettivamente stabilisce quale *provider* si utilizza. Come accennato attualmente sono previsti due diversi provider:

- LDAP (richiede la presenza di libauthldap.dll/.so nella directory dei binari)
- Password File (richiede la presenza di libauthpwdfile.dll/.so nella directory dei binari)

Di seguito un contenuto d'esempio:

```
#-------------------------------
# Property file for authLibrary
#-------------------------------
```

```
# LDAP properties
LDAP.Host = "localhost"
LDAP.Port = 389
LDAP.BindLogin = cn=Manager,dc=rtirabassi,dc=org
LDAP.BindPwd = secret
LDAP.Base = dc=rtirabassi,dc=org

LDAP.UserSearchBaseDN = ou=Users,{$base}
LDAP.UserSearchFilter = cn={$user}
LDAP.UserSearchAttributeName = cn
LDAP.GroupSearchBaseDN= ou=Groups,{$base}
LDAP.GroupSearchFilter = member=cn={$user},ou=Users,{$base}
LDAP.GroupSearchAttributeName = cn

# PWDFile properties
#PWDFile.FileName =
#PWDFile.BindLogin =
#PWDFile.BindPwd =

# Cache properties
# Timeout in seconds
Cache.timeOut = 60
```

La prima sezione, relativa alle “LDAP properties” consente di compiere l'accesso al privider LDAP richiesto. Per esso si indica porta ed indirizzo (LDAP.Port e LDAP.Host), le modalità di login (LDAP.BindLogin e LDAP.BindPwd) ed in fine la base sulla quale si intende operare (LDAP.Base) che verrà utilizzata in seguito.
Sempre nella prima sezione si indicano anche la Base, il filtro di ricerca e l'attributo di ricerca per identificare gli utenti ed i loro gruppi di appartenenza.
La variabile $base viene sostituita con quanto indicato alla voce “LDAP.Base” mentre la variabile $user viene sostituita man mano dall'utente per il quale si cerca di compiere il rilevamento delle credenziali d'accesso o le sue autorizzazioni.

Nota: Il server utilizzerà il *provider* LDAP se la voce di profilo “LDAP.Host” è presente e non vuota, se il suo valore è valido. In ogni altro caso si procederà con l'uso del *provider* su Password File.

La seconda sezione viene chiamata in causa nel momento in cui la prima non fosse presente o doverosamente configurata[3]. In essa appaiono solo 3 valori, ma di fatto solo uno di essi è realmente significativo.
La proprietà “PWDFile.FileName” indica il nome del file delle password. Anche se assente il suo valore di default è auth.passwd, che verrà rilevato nella stessa directory in cui si trova l'attuale file auth.properties.
Gli altri due valori, “PWDFile.BindLogin” e “PWDFile.BindPwd” consentono di specificare la coppia utente e password di default che si assume verrà utilizzata per compere l'accesso qualora non ne venga effettivamente esplicitata una. Nel nostro caso, cioè nel caso in cui il client dichiara la propria identità e password, tali valori non hanno significato ed anche se impostati vengono ignorati.

In ultimo, il tempo di *cache* espresso in secondi, consente al server di non reiterare una verifica inerente autenticazione o autorizzazione se non è passato un tempo minimo. Ciò ha evidenti impatti specialmente sulle performance ed ha particolare significato in presenza di *provider* LDAP. In caso di Password File l'impostazione ha egualmente valore ma risulta molto meno determinante.

Il secondo raggruppamento viene discusso in seguito.

auth.passwd

Solo nel caso in cui il *provider* prescelto sia quello semplificato che si avvale di un file di password, esso viene preso in esame. La sua collocazione e denominazione dipende da quanto impostato nel file auth.properties. Il nome di default è comunque auth.passwd.

In condizioni standard, quindi, anche questo file sarà collocato nella directory `/conf`.

Il suo formato è molto semplice:

UserID;MD5 della password dell'utente;Elenco, separato da virgole, dei “Gruppi” cui l'utente appartiene

Vediamo di seguito un semplice esempio che rappresenta il contenuto tipo di questo file:

```
gestore;4337F08642CD7995C44C817DDBF3005E;xwSuperUser
lettore;1DE9B0A30075AE8C303EB420C103C320;xwGlobalUser,xwFreeIp
xw.4879.fca;;xwGlobalUser,xwFreeIp
rtirabassi;;xwAdmin,xwGlobalUser
hwadmin;;xwAdmin,xwGlobalUser,xwFreeIp
```

In esso, quindi, vengono elencati gli utenti 'gestore' e 'lettore' per i quali viene indicata una password codificata in MD5. Ci sono poi anche gli utenti 'xw.4879.fca' e 'hwadmin'[4]. Come si può vedere essi sono privi di password in quanto non si assume che siano utenti che verranno utilizzati per fare la connessione ad eXtraWay ma che siano utenti dei quali il server riceverà semplicemente

notifica.
Ad ogni utente viene poi assegnato uno o più gruppi di appartenenza. Nel nostro esempio l'utene 'gestore' sarà un 'xwSuperUser', l'utente lettore sarà un generico 'xwGlobalUser' così come l'utente 'xw.4879.fca' mentre all'utente hwadmin viene garantito qualche diritto supplementare essendo appartenente anche al gruppo 'xwAdmin'.

auth.properties (parte seconda)

Adesso che abbiamo chiuso la parentesi di auth.passwd vediamo il secondo raggruppamento di informazioni presenti nel file auth.properties.
Esso rappresenta delle tabelle di equivalenza che servono fondamentalmente ad adeguarsi a *provider* di autorizzazione esistenti è già profilati. Di seguito, procedendo negli esempi, il tutto apparirà più chiaro.

Supponiamo comunque di aver realizzato e configurato un archivio perché i suoi utenti appartengano a determinati gruppi (ovvero siano titolari di determinati profili) e di non trovare corrispondenza nel LDAP dell'azienda dove si va a compiere l'installazione di simili profili. Naturalmente non è credibile che l'azienda stravolga il proprio sistema LDAP per adeguarsi a quello richiesto da 3D, specialmente se gli utenti hanno più o meno profili adeguati allo scopo.
Per non dover intervenire sulla configurazione dell'archivio, di cui si parlerà in seguito, procediamo a creare dei rapporti di equivalenza come ne seguente esempio.

```
# eXtraWay Connection properties
# Se non indicate il diritto deve corrispondere alla label.
xw.superuser = xwSuperUser
xw.freeip = xwFreeIp
xw.admin = xwAdmin
xw.fullcontrol = xwFullControl
xw.connect= xwConnect
xw.writer= xwWriter
xw.reader= xwReader
```

In questo modo, possiamo dire che gli utenti che sono dichiarati come appartenenti al gruppo 'xwSuperUser' sono analogamente appartenenti al gruppo 'xw.superuser' che ha lo stesso valore. Negli esempi seguenti le ragioni di questa tabella di equivalenza saranno più evidenti.

auth.profile.xml e <nomearchivio>.profile.xml

Ora che abbiamo detto come si identifica il *provider* di autenticazione ed autorizzazione e come vengono espressi da esso i profili cui un utente appartiene, vediamo come tali profili vengono utilizzati e tramutati, per così dire, in diritti.
Iniziamo parlando dei diritti generali, ovvero dei diritti che ogni utente ha nei confronti del sistema eXtraWay come tale, per poi scendere nel dettaglio dei diritti applicabili al singolo archivio.
I diritti generali sono rappresentati dal contenuto del file auth.profile.xml, file la cui collocazione ricalca quella del file auth.properties. Esso quindi si trova nella directory `/conf`.
I diritti specifici d'archivio si trovano in un file avente il nome dello stesso e collocato congiuntamente al file *nomearchivio*.conf.xml.

Il file dei profili, quello per mezzo del quale si stabilisce quali diritti spettino a chi, ha una conformazione relativamente semplice.
Innanzitutto esso prevede un elemento `arc_profile` corredato da un attributo `security`.
L'attributo `security` deve assumere uno dei seguenti valori.

- weak: L'eventuale sovrapposizione di profili è di tipo debole, quindi in situazione d'ambiguità su un diritto esso viene riconosciuto.
- strong: L'eventuale sovrapposizione di profili è di tipo forte, quindi in situazione d'ambiguità su un diritto esso viene negato.
- skip: La configurazione dei diritti non è necessaria e quindi può essere omessa. Questa modalità si riferisce normalmente alle sole configurazioni d'archivio e non alla configurazione dei diritti generali.

Esempio:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<arc_profile security="weak">
   ...
</arc_profile>
```

Poi, entro quest'elemento, sono presenti una serie di elementi `profile` corredati da due attributi: `name` e `baseAccess`. Il valore di `name` altro non è che uno dei gruppi o profili cui l'utente appartiene mentre l'attributo `baseAccess` assume uno dei seguenti valori:

- deny: indica che, per tutte le operazioni/diritti non esplicitate/i, il valore di default deve considerarsi negativo. Questo rappresenta il default quindi non vengono garantiti diritti se non diversamente esplicitato.
- allow: indica che, per tutte le operazioni/diritti non esplicitate/i, il valore di default deve considerarsi positivo.

Ogni elemento `profile` può a sua volta contenere delle `operation` che hanno, analogamente, gli stessi attributi `name` e `baseAccess`. Mentre `baseAccess` dichiara il diritto accordato, positivo o negativo, come nel caso precedente, il `name` definisce l'effettiva operazione che si intende autorizzare o meno.

A questo punto entriamo nel merito delle operazioni, ovvero dei diritti da considerarsi *generali* e quelli da considerarsi *d'archivio*.

Questo l'elenco di tali diritti:

Diritto/Operazione Generale	**Descrizione**
connect	Indica la facoltà di questo user di compiere la prima effettiva connessione al server. Deve trattarsi, necessariamente, di un utente corredato da password.
freeIp	Indica la facoltà di questo utente di accedere simultaneamente da Ip Address differenti. Il server, in questo caso, non compie le verifiche del caso ne registra alcunché di anomalo nei logs.
Diritto/Operazione d'Archivio	**Descrizione**
insertDoc	Facoltà dell'utente di inserire documenti.
modifyDoc	Facoltà dell'utente di modificare documenti.
eraseDoc	Facoltà dell'utente di cancellare documenti.
viewDoc	Facoltà dell'utente di visualizzare documenti.
exportDoc	Facoltà dell'utente di esportare documenti.

In riferimento alle operazioni d'archivio, esse possono essere sottoposte ad una o più regole. In tal caso l'elemento `operation` conterrà a sua volta uno o più elementi `rule`. Essi sono caratterizzati da 3 attributi: `type`, `value` e `access`. L'attributo `access` ha comportamento, significato e contenuto equivalente ai precedenti attributi `baseAccess`. Gli attributi `type` e `value` dichiarano invece una tipo di regola che dev'essere soddisfatta ed il contenuto della regola stessa. Solo se essa passa il vaglio, il diritto corrispondente verrà riconosciuto. In presenza di più regole è sufficiente che una di esser garantisca il diritto desiderato perché la cosa sia efficace.

Giunti a questo punto è però necessario portare alcuni esempi.
In principio vediamo un esempio di file auth.profile.xml che dichiara i diritti generali per eXtraWay.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<arc_profile security="weak">
   <profile name="xwSuperUser" baseAccess="allow"/>
      <!-- Allowing every operation as basic access mode I don't have to explain
           that this user has rights to do things that an administrator can not... -->
   <profile name="xwAdmin" baseAccess="deny">
      <!-- I don't define a 'allow' baseAccess in order to decide to allow only what really
desiderd -->
      <operation name="connect" baseAccess="allow"/>
   </profile>
   <profile name="xwGlobalUser" baseAccess="deny">
      <!-- I don't define a 'allow' baseAccess in order to decide to allow only what really
desiderd -->
      <operation name="connect" baseAccess="allow"/>
      <operation name="freeIp" baseAccess="allow"/>
   </profile>
   <!-- profile name="xwFullControl" baseAccess="deny"/ -->
   <!-- profile name="xwWriter" baseAccess="deny"/ -->
   <profile name="." baseAccess="deny"/>
</arc_profile>
```

L'esempio precedente ci dice che la sicurezza si deve considerare debole (securty="weak") e che diversi diritti vengono riconosciuti agli utenti appartenenti alle diverse categorie. Dato per assodato che attualmente i diritti generali riconosciuti sono solo `connect` e `freeIp` avremo.

- Gli utenti appartenenti al profilo `xwSuperUser` possono fare accesso principale ed accedere da qualsiasi Ip Address. **Ad essi verrà automaticamente riconosciuto qualsiasi nuovo diritto generale dovesse essere introdotto in futuro**.
- Gli utenti appartenenti al profilo `xwAdmin`, invece, avendo la negazione del diritto come condizione di default, non potranno compiere che le operazioni espressamente indicate, vale a dire la connessione.
- Gli utenti appartenenti al profilo `xwGlobaUser` hanno una sorte simile agli utenti `xwAdmin` ma ad essi viene anche concesso l'accesso da Ip Address differenti. Analogamente agli utenti di tipo `xwAdmin`, non verrà loro riconosciuto alcun ulteriore diritto generale di futura introduzione.
- Si potrebbero definire altri raggruppamenti di utenti esplicitando ad esempio che a loro ogni diritto viene negato (si vedano i commenti inerenti `xwFullControl` e `xwWriter`) ma la cosa può essere semplificata con la dichiarazione dell'utente "." cui ogni diritto viene negato. **L'utente "." è l'utente generico, anonimo, ovvero qualsiasi utente che non sia stato riconosciuto dal *provider* di autenticazione e che sia stato semplicemente notificato al server. Ad essi, per natura ed indipendentemente da questa dichiarazione, non è concesso l'accesso primario.

Sulla base di quest'esempio, sarebbe sufficiente accodare all'utente generico il diritto `freeIp`...

```
   <profile name="." baseAccess="deny">
      <operation name="freeIp" baseAccess="allow"/>
```

```
   </profile>
```

... per abbattere globalmente il controllo sull Ip Address di provenienza.

Prima di procedere vediamo in cosa consista il concetto di sicurezza forte o debole.
Assumiamo di avere un utente che appartiene contemporaneamente a due gruppi/profili (ad eccezione del profilo "." che è espressamente anonimo). Prendiamo ad esempio il nostro utente `xwAdmin` che appartiene tanto a `xwGlobalUser` che a `xwAdmin`.
Adesso domandiamoci se questo utente ha il diritto di `connect`. Dal momento che in ambo i gruppi cui esso appartiene tale diritto gli viene riconosciuto[5], egli potrà fare `connect` senza riguardi alla sicurezza forte o debole.
Se invece ci chiediamo se abbia il diritto `freeIp` di fare accesso simultaneamente da indirizzi Ip differenti vediamo che questo diritto gli viene esplicitamente riconosciuto dal gruppo `xwGlobalUser` ed implicitamente negato dal gruppo `xwAdmin`. Ecco che entra in gioco la sicurezza. In questo caso, in presenza di sicurezza forte, il diritto gli viene negato in quanto non riconosciuto da tutti i gruppi cui esso appartiene. In presenza di sicurezza debole, invece, il diritto gli viene riconosciuto in quanto consentito da almeno uno dei gruppi d'appartenenza.

Alla luce di queste ultime considerazioni risulta evidente che si può costituire la combinazione di gruppi, diritti e quant'altro a propria discrezione riuscendo ad ottenere un sistema di diritti fortemente articolato.

Analogamente è possibile immaginare che ad ogni gruppo corrisponda esclusivamente un diritto ed in tal modo l'assegnazione dei gruppi ad un utente corrisponde, in rapporto 1:1 all'attribuzione ad esso del corrispondente diritto.

Una volta discussi i diritti generali, vediamo che possibilità di configurazione ci sono invece nel caso dei diritti d'archivio. Come abbiamo visto in uno schema precedente, i diritti sono diversi e riguardano sostanzialmente la capacità di fare accesso ai documenti.
Vediamo un esempio:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<arc_profile security="weak">
   <profile name="xw.superUser" baseAccess="allow"/>
      <!-- Allowing every operation as basic access mode I don't have to explain
            that this user has rights to do things that an administrator can not... -->
   <profile name="xw.admin" baseAccess="deny">
      <!-- I don't define a 'allow' baseAccess in order to decide to allow only what really
desiderd -->
      <operation name="insertDoc" baseAccess="allow"/>
      <operation name="modifyDoc" baseAccess="allow"/>
      <operation name="eraseDoc" baseAccess="allow"/>
      <operation name="viewDoc" baseAccess="allow"/>
      <operation name="exportDoc" baseAccess="allow"/>
   </profile>
   <profile name="xw.fullControl" baseAccess="deny">
      <!-- I don't define a 'allow' baseAccess in order to decide to allow only what really
desiderd -->
      <operation name="insertDoc" baseAccess="allow"/>
      <operation name="modifyDoc" baseAccess="allow"/>
      <operation name="eraseDoc" baseAccess="allow"/>
      <operation name="viewDoc" baseAccess="allow"/>
   </profile>
   <profile name="xw.writer" baseAccess="deny">
      <!-- This user can do very few things. Access to every document, insert his own documents
            and modify documents that he wrote -->
      <operation name="viewDoc" baseAccess="allow"/>
      <operation name="insertDoc" baseAccess="allow"/>
      <operation name="modifyDoc" baseAccess="deny">
         <rule type="xpath" value="/doc/author=$user" access="allow"/>
      </operation>
   </profile>
   <profile name="xw.reader" baseAccess="deny">
      <!-- This user can do very few things, he can just read documents. -->
      <operation name="viewDoc" baseAccess="allow"/>
   </profile>
</arc_profile>
```

In quest'esempio possiamo verificare diversi aspetti tra quelli precedentemente citati.
Innanzitutto è chiaro che le denominazioni dei profili non sono quelle stesse utilizzate nel precedente esempio. I profili assumono nomi quali `xw.superUser` o `xw.admin` ed è la tabella di equivalenza precedentemente mostrata nella descrizione del file auth.properties che completa l'accoppiamento degli utenti al loro gruppo di appartenenza.

Facciamo comunque un esempio chiarificatore. Dati per accettati gli attuali diritti d'archivio ed assumendo che non ne vengano introdotti nuovi nel corso del tempo, le seguenti dichiarazioni sono del tutto equivalenti.

La prima ha logica negativa

```
<profile name="xw.fullControl" baseAccess="deny">
   <operation name="insertDoc" baseAccess="allow"/>
   <operation name="modifyDoc" baseAccess="allow"/>
   <operation name="eraseDoc" baseAccess="allow"/>
   <operation name="viewDoc" baseAccess="allow"/>
</profile>
```

mentre la seconda esprime lo stesso concetto ma in logica positiva

```
<profile name="xw.fullControl" baseAccess="allow">
   <operation name="exportDoc" baseAccess="deny"/>
</profile>
```

|

Com'è stato detto sin da subito, esiste un'ulteriore modalità di sicurezza, pari a “skip”.
Essa si applica in tutti gli archivi in cui non è richiesta alcuna verifica ne l'applicazione di alcun diritto.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<arc_profile security="skip">
   <!--
   <profile name="xwAdmin" baseAccess="allow"/>
   <profile name="xwSuperUser" baseAccess="allow"/>
   <profile name="xwGlobalUser" baseAccess="allow"/>
   <profile name="." baseAccess="allow"/>
   -->
</arc_profile>
```

Quest'esempio mostra come lo stato della sicurezza skip consenta in pratica di agire in modo incontrollato da parte di qualsiasi utente, compiendo qualsivoglia operazione. All'atto pratico ciò equivale alla forma

```
<?xml version="1.0" encoding="iso-8859-1"?>
<arc_profile security="weak">
   <profile name="." baseAccess="allow"/>
</arc_profile>
```

in quanto si da comunque qualsiasi accesso a qualsiasi utente.

1)
xusers.xml collocato nella directory /conf
2)
Ovvero Gruppi
3)
Assenza o non correttezza della proprietà LDAP.Host
4)
Il primo è il nome solitamente utilizzato dal *File Conversion Service* per non confliggere con l'utente 'lettore', il secondo è un nome automaticamente assegnato dal server all'utente per operazioni non presidiate, ad esempio le importazioni effettuate via *WatchDoc*.
5)
Implicitamente o esplicitamente, non cambia
6)
La sintassi verrà descritta meglio in altra documentazione, visto che è attualmente solo abbozzata.