ARUBA PEC S.p.A.
via Sergio Ramelli, 8 - 52100 - Arezzo
tel. 0575 0500 - fax 0575 862020

DocFly
Developer's Guide

vers. 1.9

Sommario

1. Riferimenti

[1]. DPCM 3/12/2013: Regole tecniche in materia di sistema di conservazione

[2]. CAD articolo 44: Requisiti per la conservazione dei documenti informatici

[3]. Manuale della Conservazione

[4]. Specifica Jsend

[5]. RFC 2046

[6]. RFC 3253

2. Introduzione

Questa guida vuole essere per l'utente un valido strumento ai fini di un corretto utilizzo del sistema di conservazione DocFly erogato in modalità *outsourcing*.

Aruba DocFly garantisce la conservazione a norma di qualsiasi tipo di documento informatico in linea con la normativa vigente, ovvero il DPCM 3 dicembre 2013[1]., assicurandone le caratteristiche di autenticità, integrità, affidabilità, leggibilità e reperibilità, come previsto dall'articolo 44 del CAD[2]..

Per una descrizione dettagliata delle caratteristiche del servizio si rimanda alla consultazione del Manuale di Conservazione pubblicato sul sito ufficiale dell'Agid[3].

3. Descrizione generale

Ogni tipologia di documento (per esempio fattura, contratto, libro giornale, LUL, etc.) viene denominata *classe documentale*. Essa è definita come un insieme di *meta-dati* che descrivono pienamente il documento.

Tra i meta-dati sono distinguibili principalmente dati obbligatori per norma e dati utili alla reperibilità dei documenti.

I versamenti vengono conservati in *Pacchetti di archiviazione* o PDA. Per pacchetto di Archiviazione si intende l'unità di archiviazione che viene conservata in un'unica soluzione. Esso è composto a partire da uno o più *Pacchetti di Versamento* o PdV.

Ogni Pacchetto di Versamento è costituito da un insieme di documenti omogenei tra loro per classe documentale, ovvero per insieme di meta-dati che li descrivono.

Per fornire i valori dei meta-dati dei documenti da conservare per ogni PdV, ossia per ogni classe documentale, il sistema versante deve fornire al sistema di conservazione un *Indice del Pacchetto di Versamento* o IPdV.

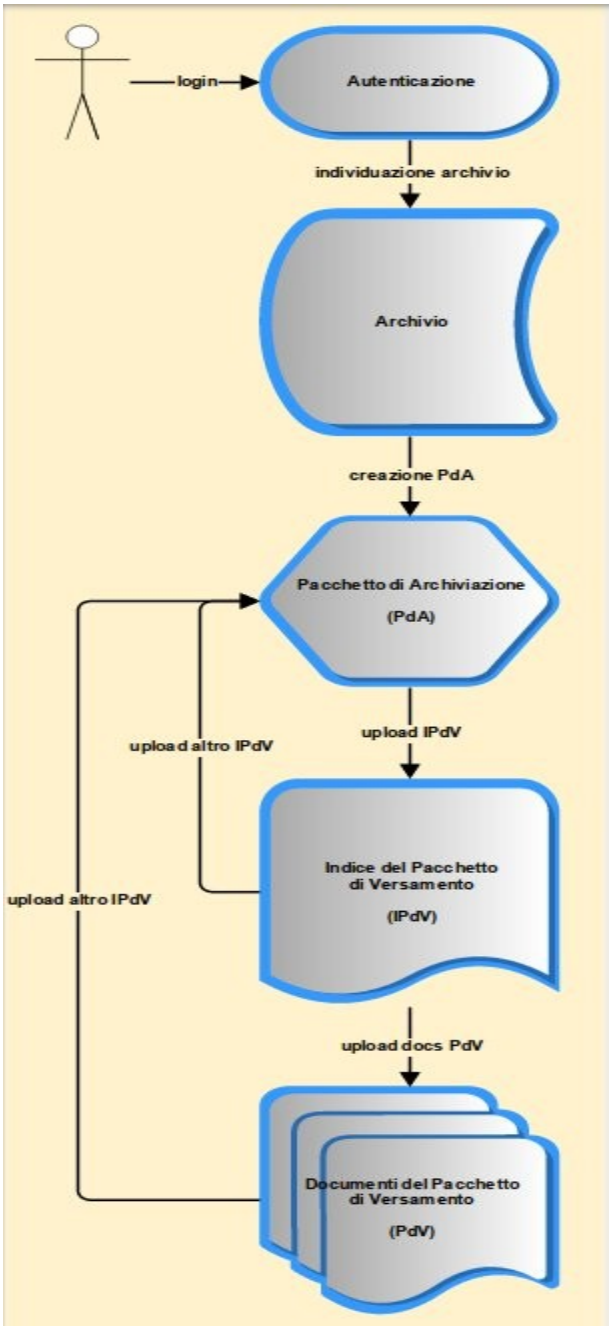
Eventuali documenti che **non** sono stati preventivamente indicati in un Indice vengono automaticamente scartati dal sistema di conservazione.

3.1. Flusso di conservazione

Il processo di versamento ottimale si compone dei seguenti passi:

1. autenticazione e identificazione dell'archivio in cui effettuare il versamento
2. creazione del Pacchetto di Archiviazione: ovvero del contenitore dove depositare tutti i documenti che si vogliono conservare in un'unica soluzione
3. upload di uno o più Indici del Pacchetto di Versamento
4. upload dei file indicati negli Indici del punto precedente

Il flusso di versamento quindi avviene come nella sequenza descritta in basso:



Questo flusso è stato pensato in maniera da dare la possibilità al Produttore di organizzare i propri documenti in una visione “a cartelle” dove ogni cartella è un singolo PdA. In questo modo è possibile organizzare documenti correlati tra loro secondo una qualche logica e conservarli dentro un unico recipiente, il PdA appunto.

Il processo di conservazione vera e propria (detto anche “chiusura del PdA”) avviene automaticamente nel momento in cui tutti i documenti dichiarati in tutti i IPdV caricati nel PdA sono stati correttamente ricevuti.

In questo senso il Produttore deve essere sicuro di avere caricato inizialmente tutti gli IPdV di interesse per evitare che il processo di conservazione del PdA parta in anticipo.

NOTA BENE: il sistema consente comunque la costruzione di fascicoli documentali a partire dalla ricerca di un valore all'interno di un meta-dato comune a più classi documentali all'interno di un archivio.

I momenti cruciali del flusso di conservazione sono i seguenti:

1. ricevimento dell'IPdV. L'indice viene elaborato e viene verificato che sia consistente relativamente alla classe documentale dichiarata. L'esito di tale verifica può avere 3 risultati possibili:
    1. ok: l'IPdV rispetta in toto le caratteristiche condivise tra sistema e Produttore
    2. ko: ci sono degli errori bloccanti che non consentono il prosieguo del flusso di conservazione
    3. warning: sono state rilevate delle anomalie ma che non risultano bloccanti per il processo di conservazione
2. ricevimento di tutti i documenti del PdV. Per ogni documento in ingresso viene verificato nome, impronta e mimetype. Se essi corrispondono con i dati indicati nell'IPdV il file viene accettato, altrimenti viene scartato
3. messa in conservazione. Automaticamente, dopo avere verificato che tutti i PdV sono completi, ovvero sono arrivati tutti i documenti attesi, il PdA viene posto in conservazione

Nelle fasi principali di tale processo è prevista la generazione di opportune notifiche. Queste vengono inviate via PEC. I destinatari di tale mail dipendono dal tipo di evento da notificare e sono individuati in fase di censimento del Produttore.

E' possibile ricevere le notifiche del processo di conservazione anche attraverso un'interfaccia applicativa. Questo è possibile fornendo al sistema una url di callback messa a disposizione dal sistema versante.

Nell'Appendice B: notifiche tramite callback url viene descritta tale funzionalità.

4. Interfaccia Web Service

Essa fornisce tutti i metodi utili per il processo di versamento e per le operazioni di interrogazione al sistema.

L'interfaccia è stata realizzata secondo il protocollo REST, dando così maggiore flessibilità al client nel costruire le proprie politiche di integrazione.

Le chiamate utili alla conservazione restituiscono le loro risposte in formato JSON.

La semantica del risultato rispetta la specifica Jsend[4].

Di seguito sono riportati i dati utili per l'integrazione.

Ambiente	Descrizione	URL
Collaudo	DOCFLY_WS_URL	https://collaudo.docfly.it/api
Produzione	DOCFLY_WS_URL	https://api.docfly.it

NOTA BENE: è **obbligatorio** effettuare tutti i test esclusivamente nell'ambiente di collaudo in quanto le operazioni di conservazione sono da ritenersi irreversibili.

Di seguito vengono descritti i metodi REST messi a disposizione:

i. login

Effettua l'autenticazione e restituisce il ticket di sessione da usare per future interazioni.

I parametri per la chiamata sono:

- **endpoint**: /signin
- **method**: POST
- **params**:
  - username: username dell'utente
  - password: password dell'utente
  - options: non obbligatorio. Al momento l'unico valore valido è *archivesInfoON* che attiva la visualizzazione degli archivi visibili dall'utente[1].
- **returns**:
  - data.ticket: ticket di autenticazione necessario per le interrogazioni
  - data.homeNodeRef: riferimento unico alla home del Produttore, necessaria per la navigazione degli archivi

Di seguito un esempio in Python2 della chiamata di login

```
def login(username, password):


    """ Effettua login """

    endpoint = 'https://collaudo.docfly.it/api/signin'

    params = {

        'username': username,

        'password': password,
```

1 Nel caso del produttore visualizza tutti gli archivi. Se il produttore ha un numero molto alto di archivi e tale informazione non è interessante **è fortemente consigliato** non utilizzare tale opzione

```
    }

    response = requests.post(endpoint, data=params)

    json = response.json()


    ticket = json['data']['data']['ticket']

    home_node_ref = json['data']['homeNodeRef']

    return (ticket, home_node_ref)
```

ii. logout

Effettua la chiusura della sessione creata in fase di login.

I parametri per la chiamata sono:

- **endpoint**: /signout
- **method**: DELETE
- **params**:
  - ticket: ticket recuperato in fase di autenticazione

```
def logout(ticket):

    """ Esegue la signout. Ritorna True se OK, False altrimenti """

    endpoint = 'https://collaudo.docfly.it/api/signin'

    params = {

        'ticket': ticket,

    }

    return requests.delete(endpoint, params=params).json()
```

4.1 Informazioni di sistema

Permette il reperimento delle informazioni di sistema dell'utente che ha effettuato il login

I parametri per la chiamata sono:

- **endpoint**: /getSystemInfo
- **method**: GET
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - username: username dell'utente
- **results:**
  - data: contiene una struttura con le caratteristiche di sistema proprie dell'utente

4.2 Individuazione archivio

Una volta acceduto, l'utente deve individuare l'archivio su cui vuole effettuare il versamento. Normalmente esso coincide col codice fiscale legato all'archivio di riferimento. Tale dato deve essere indicato nel path in cui si vuole effettuare il versamento.

Se l'archivio indicato non esiste, il sistema restituisce immediatamente un errore.

4.3 Procedure di versamento: creazione PdA e upload IPdV

All'interno dell'archivio scelto l'utente deve scegliere il PdA contenitore dei materiali da mettere in conservazione. Per fare questo l'utente indica in un apposito parametro il nome del PdA. Se esso non esiste viene creato ex novo, altrimenti il versamento ha luogo al suo interno.

Come descritto nel capitolo Flusso di conservazione l'upload di un PdV si distingue in due passi: upload dell'IPdV e upload dei documenti da conservare per tale IPdV. I due passi devono essere eseguiti esattamente in questo ordine, altrimenti il sistema potrebbe scartare documenti perché non dicharati in alcun IPdV.

Inoltre nel caso di PdA composti da più di un PdV è bene effettuare prima l'upload di tutti gli IPdV e successivamente dei documenti da conservare. In questo modo non si rischia la prematura chiusura del PdA.

Per le caratteristiche peculiari dell'IPdV fare riferimento al capitolo Appendice A: caratteristiche dell'IPdV.

Tutte le operazioni di upload, sia degli IPdV sia dei documenti vengono fatte tramite lo stesso endpoint, *uploadQueue*.

I parametri per la chiamata sono:

- **endpoint**: /uploadQueue
- **method**: POST
- **params**:
  - username: username dell'utente
  - password: password dell'utente
  - owner: archivio di riferimento identificato dal codice fiscale
  - destination: path del PdA a partire dal parametro *owner*

- filename: nome del file da versare (sia esso un IPdV o un documento da conservare)
- filedata – binario del file da versare

NOTA BENE: nel caso si rendesse necessario modificare le caratteristiche del versamento, ad esempio per correggere alcuni metadati o aggiungere/rimuovere documenti dal versamento, è sufficiente fare l'upload di un nuovo IPdV corretto che riporti stesso nome file e stesso *pdvid* riportato nel l'IPdV considerato obsoleto.

NOTA BENE: se l'upload va a buon fine il metodo restituisce un messaggio di successo. Questo indica esclusivamente il buon esito del trasferimento del file. Esso viene quindi messo in una coda di elaborazione per le successive fasi di validazione ed eventuale chiusura dei pacchetti. Tali elaborazioni verranno effettuate solo per quei PDA che riportano almeno un IPdV e almeno un documento

Al capitolo Appendice C: esempi di codice con chiamate servizio REST è presente un esempio in Python2 di versamento con l'invio di un IPdV e di un solo documento

4.4 Upload docs PdV

Il sistema ha tutte le informazioni necessarie per l'elaborazione del PdV. L'utente ha quindi la possibilità di operare l'upload dei documenti da mettere in conservazione.

Il procedimento è identico a quanto già fatto in fase di upload dell'IPdV.

Al capitolo Appendice C: esempi di codice con chiamate servizio REST è presente un esempio in Python2 di versamento con l'invio di un IPdV e di un solo documento

4.5 Check dei file rifiutati

Per controllare se i file inviati via web service sono stati rifiutati è possibile richiamare il metodo *getDispose*. Esso, relativamente a un singolo PDA, restituisce un file contenente indicazioni riguardo a file processati ma rifiutati, se ce ne sono.

I parametri per la chiamata sono:

- **endpoint**: /getDispose
- **method**: GET
- **params**:
  - username: username dell'utente
  - password: password dell'utente
  - owner: archivio di riferimento identificato dal codice fiscale
  - destination: path del PdA a partire dal parametro owner

- **returns**:
  - file testuale contenente le seguenti informazioni, separate dal carattere ":":
    - data del processamento
    - nome del file
    - descrizione dell'errore

Nel caso in cui vengano indicati *owner* e/o *destination* errati o qualsiasi altro tipo di errore, la richiesta restituisce il codice *400*, ovvero una *Bad Request*.
Nel caso il file non esista invece viene restituito comunque un codice *200* e un messaggio in formato JSON che indica questa evenienza.

NOTA BENE: è probabilmente utile approfondire l'argomento del controllo dei versamenti. La procedura migliore per fare verifiche sui documenti versati via web service è la seguente:
Effettuare una ricerca del documento mediante il metodo *search* descritto al capitolo Appendice C: esempi di codice con chiamate servizio REST:

1 se la ricerca non restituisce nulla significa che il documento non è stato ancora elaborato. In questo caso chiamare il metodo *getDispose*
  1.1 se non viene restituito alcun file degli errori o esso non contiene il documento cercato, significa che il documento non è stato ancora elaborato
  1.2 altrimenti è possibile verificare quale problema ci sia stato in fase di elaborazione

2 se la ricerca restituisce il documento con le sue proprietà, verificare l'attributo *archived*
  2.1 se esso vale *true* il documento è stato correttamente elaborato e conservato
  2.2 se esso vale *false* il documento è stato correttamente elaborato ma non è stato ancora conservato

Una volta che il PDA viene conservato, a intervalli regolari l'intero spazio dedicato al PDA dal sistema web service viene cancellato. Per cui, poiché l'utilità del metodo *getDispose* è strettamente legata al controllo durante il processo di messa in conservazione, la sua invocazione dopo l'operazione di cancellazione restituisce un errore.
In ultimo il buon esito della conservazione di tutto il PDA può essere controllato tramite il meccanismo delle callback descritto nell'Appendice B: notifiche tramite callback url.

4.6 Cancellazione PDA

E' possibile cancellare PDA esclusivamente se non sono ancora stati conservati.
I parametri per la chiamata sono:

- **endpoint**: /deletePDA
- **method**: POST
- **params**:

- ticket: ticket recuperato in fase di autenticazione
- nodeRef: id univoco del PDA che si vuole cancellare

NOTA BENE: la cancellazione di un PDA implica la cancellazione dei documenti al suo interno ed è una operazione **irreversibile**. In caso di cancellazione erronea è possibile soltanto ricreare il PDA e riversare nuovamente i documenti.

Al capitolo Appendice C: esempi di codice con chiamate servizio REST è presente un esempio in Python2 di cancellazione di un PDA

4.7 Browsing del sistema di conservazione

Tramite web service è possibile navigare all'interno dello spazio dedicato.

I parametri per la chiamata sono:

- **endpoint**: /viewRepository
- **method**: GET
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - nodeRef: id univoco della cartella in cui si sta navigando. Inizialmente usare il contenuto del parametro *homeNodeRef* restituito dal metodo signin
  - page: serve ad accedere alla singola pagina dei risultati. La prima pagina è la 1
- **returns:**
  - data.children: lista dei contenuti individuati all'interno del nodeRef indicato in input. Di particolare rilevanza troviamo:
    - data.children.nodeRef: id univoco dell'elemento
    - data.children.typeLocalName: tipologia dell'elemento (doc, sip, folder)
    - data.children.title: nome mnemonico dell'elemento
    - altri dati dipendenti dalla classe documentale di appartenenza, in caso di *data.children.typeLocalName = doc*

4.8 Interrogazioni al sistema di conservazione

Tramite web service è possibile ricercare i contenuti precedentemente caricati

I parametri per la chiamata sono:

- **endpoint**: /search
- **method**: POST
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - className: nome della classe documentale per cui si vuole fare la ricerca. Ovvero il valore del tag *docClass* dell'IPdV, come descritto nel capitolo Appendice A: caratteristiche dell'IPdV
  - elenco di coppie nome/valore, dove il *nome* è l'identificativo del singolo meta-dato definito nella classe documentale di appartenenza del documento**. Più in basso è descritto dettagliatamente il valore di tale parametro**
- **returns:**

- data.results: Ogni item all'interno di questa struttura rappresenta un singolo documento rilevato in base ai filtri di selezione

Poiché internamente al sistema i metadati vengono gestiti tramite il concetto dei namespace il nome del campo di ricerca deve rispondere a questa esigenza.

La gestione del nome del campo di ricerca è diversa a seconda se si voglia realizzare una ricerca filtrando su

- dati interni alla conservazione
- un dato di tipo *singlemetadata*
- un dato all'interno di un *complexmetadata*

Per dati interni alla conservazione si intendono i seguenti:

- *pdvid* ovvero l'id univoco del pacchetto di versamento così come indicato nell'IPdV. Il nome utile per la ricerca rimane **conservazione.doc:pdvid**
- *docid* ovvero l'id univoco del file così come indicato nell'IPdV. Il nome utile per la ricerca rimane **conservazione.doc:docid**
- *filename* ovvero il nome del file così come indicato nell'IPdV. Il nome utile per la ricerca rimane **conservazione.doc:filename**
- *closingDate* ovvero la data di ultima modifica del file così come indicato nell'IPdV. Il nome utile per la ricerca rimane **conservazione.doc:closingDate**

Nel caso di filtro su un *singlemetadata* è necessario comporre il nome del campo di ricerca secondo la seguente specifica:

*<nome namespace>***:***<nome metadato>*

Prendiamo in esame l'esempio seguente, ricavato dall'IPdV mostrato al capitolo Appendice A: caratteristiche dell'IPdV:

```
<singlemetadata>
        <namespace>conservazione.doc</namespace>
        <name>oggettodocumento</name>
        <value>Contratto Roma Salv</value>
</singlemetadata>
```

Se volessimo filtrare la ricerca sul campo *oggettodocumento* dovremmo indicare come nome del campo di ricerca:

conservazione.doc:oggettodocumento

Nel caso di filtro su un dato all'interno di un *complexmetadata* è sufficiente comporre il nome del campo di ricerca secondo la seguente specifica:

idxed:<*nome del metadato*>_<*nome del complexmetadata*>

Prendiamo in esame l'esempio seguente, ricavato dall'IPdV mostrato al capitolo Appendice A: caratteristiche dell'IPdV:

```
<complexmetadata namespace="conservazione.doc" name="soggettoproduttore"
namespaceNode="conservazione.soggetti" nodeName="soggetto">
        <singlemetadata>
                <namespace>conservazione.soggetti</namespace>
                <name>nome</name>
                <value>Pippo</value>
        </singlemetadata>
        <singlemetadata>
                <namespace>conservazione.soggetti</namespace>
                <name>cognome</name>
                <value>Claudio</value>
        </singlemetadata>
        <singlemetadata>
                <namespace>conservazione.soggetti</namespace>
                <name>codicefiscale</name>
                <value>RSSMRA85T10A562S</value>
        </singlemetadata>
</complexmetadata>
```

Se volessimo filtrare la ricerca sul campo *cognome* di questo *soggettoproduttore* dovremmo indicare come nome del campo di ricerca:

idxed:cognome_soggettoproduttore

NOTA BENE: il valore di ricerca può includere dei caratteri jolly per ricerche su dati parziali. In particolare può essere usato il carattere * per indicare una sequenza di caratteri lunga a piacere e il carattere *?* per indicare un singolo carattere.

NOTA BENE: il risultato della ricerca riporta dati interessanti relativi ai singoli documenti restituiti. Di particolare importanza è l'attributo ***archived*** che se vale t*rue* indica un documento correttamente conservato.

4.9 Richieste di esibizione

Tramite web service è possibile anche richiedere *Pacchetti di Distribuzione* ovvero il download a norma di legge di materiali in conservazione.

Sono forniti due metodi, a seconda se si voglia richiedere l'esibizione di un singolo PdA o di un elenco di documenti.

Nel caso di richiesta di esibizione di un singolo PdA i parametri per la chiamata sono:

- **endpoint**: /distribute
- **method**: GET
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - nodeRef: riferimento unico al PdA
- **returns:** un file .zip contenente il PdD

Nel caso di richiesta di esibizione di un insieme di contenuti i parametri per la chiamata sono:

- **endpoint**: /mixedDistribute
- **method**: POST
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - nodeRefs: lista di riferimenti unici ai materiali da esibire
- **returns:** un file .zip contenente il PdD coi contenuti richiesti

Al capitolo Appendice C: esempi di codice con chiamate servizio REST è presente un esempio in Python2 di versamento con l'invio di un IPdV e di un solo documento

4.10 Download di un documento

In alcune situazioni potrebbe non essere importante recuperare la conservazione a norma di un documento, ma solo il documento stesso. Questo è possibile con il metodo *downloadFile*

I parametri per la chiamata sono:

- **endpoint**: /downloadFile

- **method**: GET
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - nodeRef: riferimento unico del file cercato
- **returns:** il file richiesto, se è stato elaborato dal sistema

4.11 Recupero Rapporto di Versamento

Tramite web service è possibile verificare lo stato di un versamento richiedendone l'eventuale rapporto, se è stato creato. Il Rapporto è un file HTML contenente informazioni analoghe a quanto inviato via PEC immediatamente dopo il ricevimento di tutti i materiali costituenti il versamento (IPdV e tutti i documenti ivi indicati).

I parametri per la chiamata sono:

- **endpoint**: /getRdcv
- **method**: GET
- **params**:
  - ticket: ticket recuperato in fase di autenticazione
  - owner: archivio di riferimento identificato dal codice fiscale
  - pdvid: ID del PDV così come riportato all'interno dell'IPdV in fase di versamento
- **returns:** un file .tsd contenente il rapporto del pacchetto di versamento firmato e marcato al momento in cui è stato generato

4.12 Check spazio di conservazione

E' possibile conoscere le caratteristiche e lo stato dello spazio messo a disposizione del Produttore tramite il metodo *spaceStatus*.

I parametri per la chiamata sono:

- **endpoint**: /spaceStatus
- **method**: GET
- **params**: ticket: ticket recuperato in fase di autenticazione
- **returns:** un oggetto Json contenente lo username dell'utente produttore, il numero dei documenti, lo spazio disco totale, lo spazio disco occupato e la data di riferimento del calcolo

4.13 Ultimo accesso

Tra i dati di controllo è possibile conoscere l'istante di ultimo accesso di un determinato utente.

I parametri per la chiamata sono:

- **endpoint**: /lastAccesss
- **method**: GET
- **params**: ticket: ticket recuperato in fase di autenticazione
- **returns**: un oggetto Json contenente lo username dell'utente, data e ora dell'ultimo tentativo di accesso e l'esito dell'operazione

5. Interfaccia PEC/Mail

Tra i canali di versamento il più immediato risulta quello dell'invio dei documenti da conservare tramite una mail.

Docfly prevede questa possibilità mettendo a disposizione un indirizzo PEC per la ricezione di documenti da mettere in conservazione.

In ambiente di collaudo la PEC di riferimento è: test.docfly@test.pec.aruba.it

Per usufruire di tale tipologia di servizio l'utenza del mittente delle email deve essere stato preventivamente censito a sistema, in maniera da consentire al sistema di conservazione di avere una chiave per l'identificazione del versante e il rispetto delle corrette regole di accesso. In particolare deve trattarsi della PEC del Responsabile della Conservazione di un produttore a sistema.

I documenti da inviare in conservazione devono essere allegati alla PEC con le seguenti convenzioni:

- tutti i documenti devono essere contenuti all'interno di un unico file .zip
- non verranno presi in considerazione il subject e il corpo della mail
- i file devono essere tutti sulla radice del file .zip. Eventuali file contenuti in folder annidati non vengono presi in considerazione
- il nome dello zip contiene tutte le informazioni per individuare il PdA contenente i versamenti. In particolare sarà della forma ***<archivio>#<nome pda>*** dove il carattere **#** separa le due informazioni
- la conservazione del PDA è atomica, ovvero lo zip contiene tutti i file utili per il versamento: tutti gli xml di tutti i versamenti e tutti i documenti relativi. In particolare non sarà possibile completare versamenti con nuovi invii PEC. In questo caso il completamento può essere fatto solo tramite gli altri canali (web, web service, ftp)
- l'invio di un file zip con lo stesso nome di uno precedentemente inviato non è consentito. Il mittente viene avvertito di questa eventualità sempre tramite PEC
- una volta inviato il file a DocFly la procedura di conservazione rimane identica a quella degli altri canali con relative notifiche di errore, rapporti, etc.

6. Interfaccia FTP

Uno dei più comuni e più vecchi protocolli di scambio di file è FTP. Si pone come uno dei servizi messi a disposizione sin dalla nascita di internet e nel tempo è cresciuto in flessibilità grazie alla possibilità di creare connessioni sicure, nella sua declinazione FTPS, e al recupero di download interrotti.

Per sua natura FTP prevede la possibilità di operare l'upload sia di un numero elevato di file, sia di file di grandi dimensioni.

Docfly sfrutta questa natura consentendo al sistema versante di produrre grosse moli di dati e demandando temporaneamente la loro elaborazione per la messa in

conservazione. Questo accade grazie a un sistema di code che disaccoppia l'operazione di versamento da quella di elaborazione per la conservazione.

Di seguito sono riportati i dati utili per l'integrazione.

Ambiente	Descrizione	URL
Collaudo	DOCFLY_FTP_URL	ftps://collaudo.docfly.it/
Produzione	DOCFLY_FTP_URL	ftps://ftp.docfly.it/

NOTA BENE: per raggiungere correttamente il servizio FTP è necessario che siano accessibili dalla propria rete le porte 990 e quelle da 60000 a 64096

Nella configurazione di tale canale Docfly provvede, in sede di attivazione dell'archivio, alla creazione della opportuna struttura dati per il ricevimento dei versamenti.

In particolare ogni Produttore avrà un suo spazio privato in cui troverà già pronte tante cartelle quanti sono gli archivi attivi a sua disposizione.

Di seguito è descritto il processo di versamento tramite tale canale.

La connessione al server FTP di DocFly avviene più esattamente su protocollo FTPS che consente la crittografia dei dati in comunicazione tra i due sistemi per motivi di sicurezza.Può essere quindi utilizzato qualsiasi client che supporti questo protocollo.

L'accesso può essere effettuato sia da un sistema automatico sia tramite interfaccia utente. Esistono infatti client utilizzabili in entrambe le modalità.

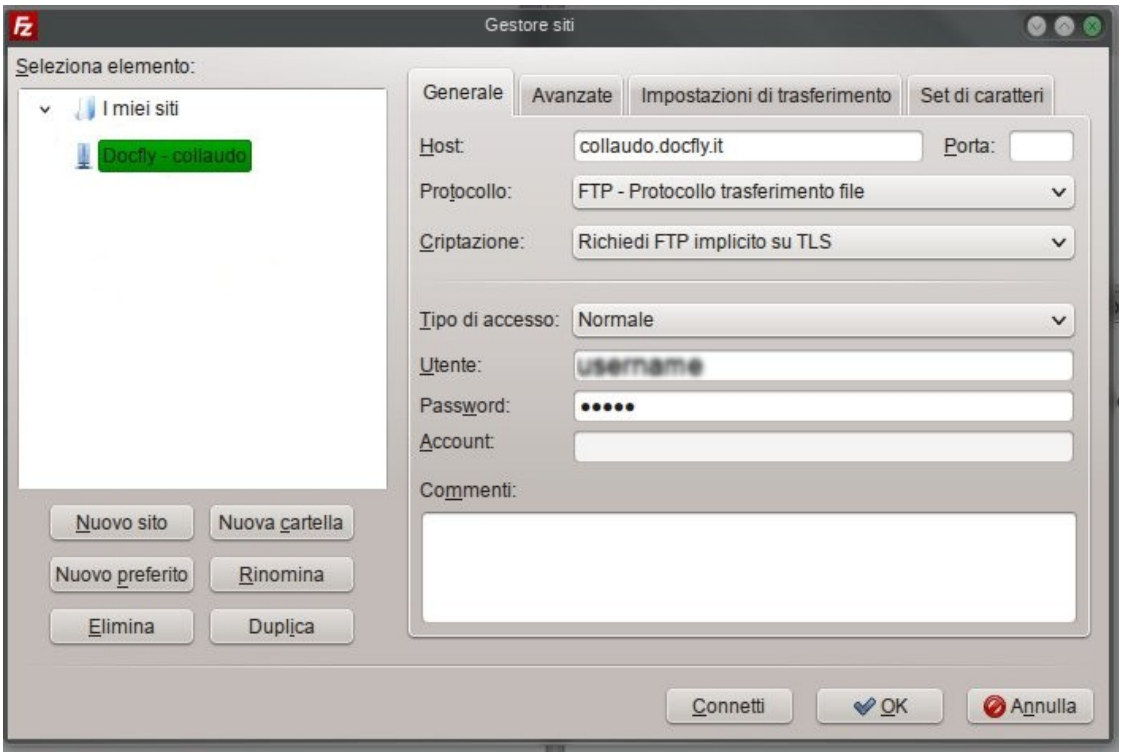
Tra i più diffusi forse il più noto è Filezilla (https://filezilla-project.org/) che ha la peculiarità di essere multipiattaforma, ovvero eseguibile sui sistemi operativi più diffusi.

In alto a destra è visibile l'icona che dà accesso alla rubrica di connessioni

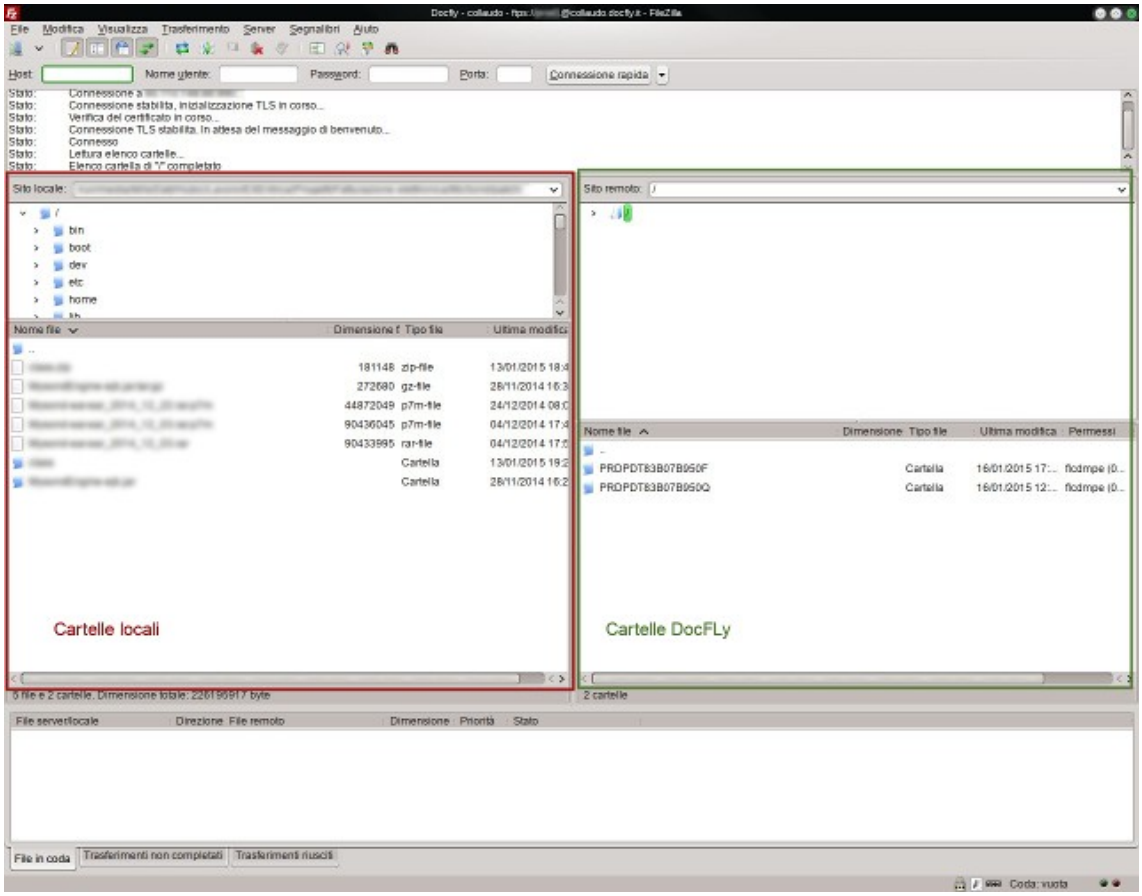
I dati di connessione da impostare sono:

- Host: come da tabellina precedente
- Protocollo: *FTP*
- Criptazione: *Richiedi FTP implicito su TLS*
- Tipo di accesso: *Normale*

- Utente: *vostra_username*
- Password: *vostra_password*



A connessione avvenuta l'utente è composta da due parti: sulla sinistra sono visibili le cartelle del proprio computer, sulla destra lo spazio DocFly personale.



In tale spazio è inizialmente visibile l'elenco degli archivi in cui è possibile versare. Come di consueto, ogni archivio è identificato dal codice fiscale del titolare dell'archivio stesso.

Una volta acceduto all'archivio corretto il processo di versamento si sviluppa nei seguenti passi:

- creazione del PdA, ovvero della cartella che conterrà tutti i versamenti
- accesso alla cartella del PdA
- upload degli IPdV. Docfly a cadenze predefinite verifica l'esistenza di nuovi PdA e/o di nuovi IPdV da processare. Nel caso dovesse trovarne, li elabora e si predispone alla ricezione dei documenti da conservare. **NOTA BENE:** nel caso si rendesse necessario modificare le caratteristiche del versamento, ad esempio per correggere alcuni metadati o aggiungere/rimuovere documenti dal versamento, è sufficiente fare l'upload di un nuovo IPdV corretto che riporti lo stesso *pdvid* riportato nel l'IPdV considerato obsoleto.
- upload dei documenti da conservare. A cadenze predefinite il sistema controlla l'esistenza di documenti da conservare, così come dichiarati negli IPdV del passo precedente

- una volta ricevuti tutti i documenti di tutti gli IPdV di un PdA, DocFly procede automaticamente con la loro messa in conservazione

Se l'IPdV non dovesse essere valido il file viene rinominato aggiungendo un suffisso *.novalidate*.

Se l'IPdV risulta corretto viene rinominato in *.processed* e il sistema attende la fine del caricamento di tutti i file inerenti il versamento.

Per ogni documento versato, se questo risulta corretto dopo la validazione, viene rinominato aggiungendo un suffisso *.processed*, altrimenti viene rinominato con suffisso *.novalidate*.

Nel caso di errori durante la validazione dell'IPdV o di uno qualsiasi dei documenti viene creato all'interno della cartella di archiviazione un file *docfly_disposed.log* contenente tutte le informazioni dell'errore.

Nello specifico, separati dal carattere **:** si ritrovano:

- data di processamento
- nome del file
- messaggio di errore

NOTA BENE: è probabilmente utile approfondire l'argomento del **controllo dei versamenti**.
Il file *docfly_dispose.log* consente di verificare in maniera autonoma le varie situazioni di errore relative all'elaborazione dei documenti versati. Tale file ha un senso durante la fase di caricamento degli stessi, mentre perde di efficacia una volta che la conservazione va a buon fine.
Una volta quindi che il PDA viene conservato, a intervalli regolari l'intero spazio dedicato al PDA dal sistema FTP viene cancellato, compreso il file *docfly_dispose.log*.

7. Appendice A: caratteristiche dell'IPdV

L'IPdV è un file in formato XML che descrive un PdV, ovvero le specificità di una determinata classe documentale.

L'IPdV deve avere un naming ben preciso, del tipo

IPDV-<custom del Pacchetto di Versamento>.xml

dove la parte custom è definibile dall'utente e utile a identificare univocamente il file dell'IPdV sul file system.

Dato che l'IPdV è strettamente correlato a una classe documentale non è possibile fornire un dettaglio esatto del suo contenuto. In questo manuale vengono descritte le caratteristiche comuni a ogni IPdV. Per semplicità riporteremo anche un semplice esempio.

Si tenga presente che è sempre possibile ricavare un template dell'IPdV di una qualsiasi classe documentale attraverso l'opportuno link nell'interfaccia web.

Al suo interno sono contenute caratteristiche proprie e del PdV in sé e dei documenti che lo compongono.

Tali caratteristiche sono indicate tramite meta-dati. Ogni meta-dato altro non è che un'associazione nome/valore utile a descrivere in maniera succinta ma sufficientemente completa il contenuto di un documento di un certo tipo.

Tipicamente i meta-dati si distinguono in due categorie:

- meta-dati di sistema, che hanno ragion d'essere in virtù di quanto prevede la norma o necessari alla corretta gestione da parte del sistema
- meta-dati documentali, utili alla catalogazione dei materiali e successivamente utilizzati per la ricerca degli stessi o per la loro esibizione

Ogni metadato viene descritto attraverso uno o più tag nell'IPdV.

NOTA BENE: è utile sottolineare che gli ambienti di collaudo e produzione sono due installazioni distinte per cui i dati utilizzati in ambiente di collaudo potrebbero differire da quelli in ambiente di produzione. E' il caso ad esempio delle classi documentali che contengono al loro interno delle nomenclature (nome della classe, nomi dei metadati, eventuali ID interni) che sono esclusivi della singola installazione. E' bene quindi prevedere in fase di sviluppo la parametrizzazione di tali dati in maniera da agevolare successivamente la migrazione degli applicativi di integrazione verso l'ambiente di produzione.

Relativamente al Pacchetto di Versamento è di fondamentale importanza il tag ***pdvid*** che rappresenta un identificativo univoco della richiesta di versamento. Nei sistemi automatici questo id viene normalmente prodotto da un sistema documentale.

Poiché ogni Pacchetto di Versamento può essere formato da un insieme di documenti omogenei per tipologia (= classe documentale), è necessario indicare nell'indice la tale tipologia. Questa informazione va inserita nel tag ***docClass***.

Successivamente l'indice elenca i metadati relativi a ogni singolo documento che partecipa il versamento.

A questo scopo bisogna introdurre nell'xml una sezione ***files*** che contiene un elenco di tag ***file***, uno per ogni documento da mettere in conservazione.

Alcuni meta-dati sono comuni a qualsiasi tipologia di documento e devono essere introdotti obbligatoriamente:

- ***docid*** assegna al documento un identificativo univoco, dal lato di chi versa, al singolo documento. Normalmente tale dato viene ottenuto da un sistema documentale
- ***filename*** indica il nome del documento, comprensivo di eventuale estensione, così come viene memorizzato su file system
- ***mimetype*** indica il tipo di documento, nel senso informatico del termine, secondo la RFC 2046[5].. Le tipologie di documento ammesse per la classe documentale scelta vengono definite in sede di accordo tra il Responsabile del Sistema di Conservazione e il Produttore. **Nel caso di file firmati (tipicamente con estensione .p7m) il mimetype da indicare è quello del file sbustato.** Se ad esempio si volesse conservare il file *miofile.pdf.p7m* il mimetype corretto da indicare sarebbe *application/pdf*, se invece si volesse conservare *miofile.txt.p7m* il mimetype corretto da indicare sarebbe *text/plain*.
- ***closingDate*** indica la data di chiusura del documento, ovvero la data di ultima modifica precedente alla messa in conservazione
- ***hash*** contiene l'impronta del documento. Il tag è corredato dall'attributo *algorithm* che specifica l'algoritmo di calcolo dell'impronta. DocFly accetta al momento impronte prodotte esclusivamente tramite **SHA-256 codificate in base64**.

Il meta-dato ***hash*** è particolarmente importante in quanto identifica in maniera univoca e certa, nei limiti di quanto indicato nel Manuale della Conservazione, il documento e viene considerato la chiave per sincerarsi che i documenti in ingresso al Pacchetto di Archiviazione siano effettivamente attesi e siano quindi quelli corretti.

Nel momento dell'elaborazione di un documento in ingresso a un PdA DocFly calcola il suo hash e, se questo non coincide con un'impronta indicata in un PdV di quel PdA, lo scarta.

Altrimenti procede con altre validazioni e in caso di esito positivo associa al documento i restanti meta-dati rilevati nell'IPdV.

Il resto dei meta-dati di un documento sono variabili in numero e tipologia a seconda della classe documentale di appartenenza.

Ci sono tipicamente due tipi di meta-dati: *singlemetadata* e *complexmetadata*.

I primi sono di tipo flat, ovvero contengono un solo valore, i secondi raggruppano insiemi di meta-dati.

NOTA BENE: *è a discrezione del sistema DocFly generare meta-dati di tipo singlemetadata o complexmetadata.*

Un meta-dato di tipo *simple* è indicato nell'IPdV dal tag ***singlemetadata*** e normalmente contiene due sotto-tag, ***name*** e ***value***. Il primo contiene il nome del meta-dato, il secondo il valore attuale relativo al documento in esame.

Di seguito è mostrato un esempio che riepiloga quanto finora descritto:

```
<?xml version="1.0"?>
<PDV>
        <pdvid>3222</pdvid>
        <docClass namespace="conservazione.doc">doc</docClass>
        <files>
                <file>
                        <docid>1234</docid>
                        <filename>Test Conservazione.txt</filename>
                        <mimetype>text/plain</mimetype>
                        <closingDate>2014-04-14T23:59:59Z</closingDate>
                        <hash algorithm="SHA-256">

        <value>bwN48hpJX1wTJHMX0Vjp1R2kWlv2j8LzZuRQ3q/cgwI=</value>
                        </hash>
                        <metadata>
                                <mandatory>
```

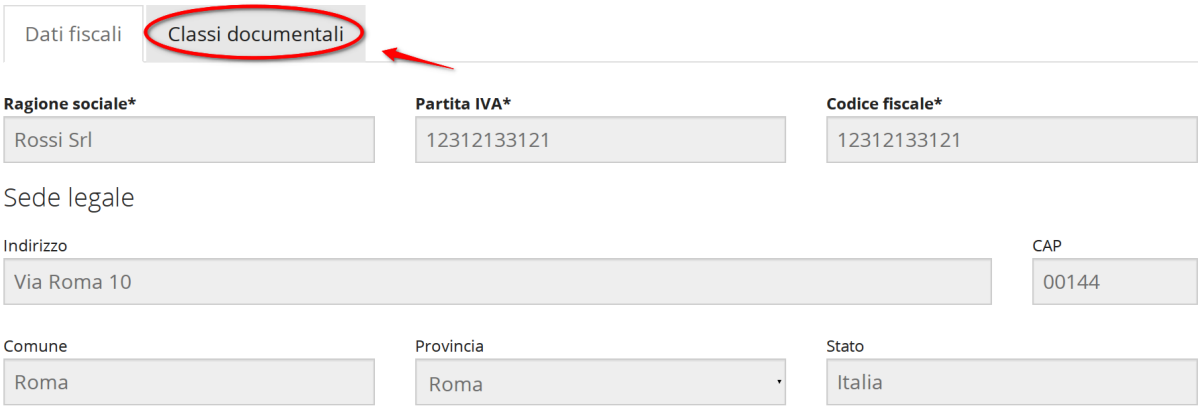
```
                                <singlemetadata>

        <namespace>conservazione.doc</namespace>
                                        <name>oggettodocumento</name>
                                        <value>Contratto Roma Salv</value>
                                </singlemetadata>
                                <complexmetadata namespace="conservazione.doc"
name="soggettoproduttore" namespaceNode="conservazione.soggetti"
nodeName="soggetto">
                                        <singlemetadata>

        <namespace>conservazione.soggetti</namespace>
                                                <name>nome</name>
                                                <value>Pippo</value>
                                        </singlemetadata>
                                        <singlemetadata>

        <namespace>conservazione.soggetti</namespace>
                                                <name>cognome</name>
                                                <value>Claudio</value>
                                        </singlemetadata>
                                        <singlemetadata>

        <namespace>conservazione.soggetti</namespace>
                                                <name>codicefiscale</name>
                                                <value>RSSMRA85T10A562S</value>
                                        </singlemetadata>
                                </complexmetadata>
                                <complexmetadata namespace="conservazione.doc"
name="destinatario"
```

```
                                    namespaceNode="conservazione.soggetti"
nodeName="soggetto">
                                    <singlemetadata>

        <namespace>conservazione.soggetti</namespace>
                                            <name>nome</name>
                                            <value>Salvo</value>
                                    </singlemetadata>
                                    <singlemetadata>

        <namespace>conservazione.soggetti</namespace>
                                            <type>String</type>
                                            <name>cognome</name>
                                            <value>Roma</value>
                                    </singlemetadata>
                                    <singlemetadata>

        <namespace>conservazione.soggetti</namespace>
                                            <type>String</type>
                                            <name>codicefiscale</name>
                                            <value>RSSSSA85T10A562S</value>
                                    </singlemetadata>
                                </complexmetadata>
                            </mandatory>
                        </metadata>
                    </file>
        </files>
</PDV>
```

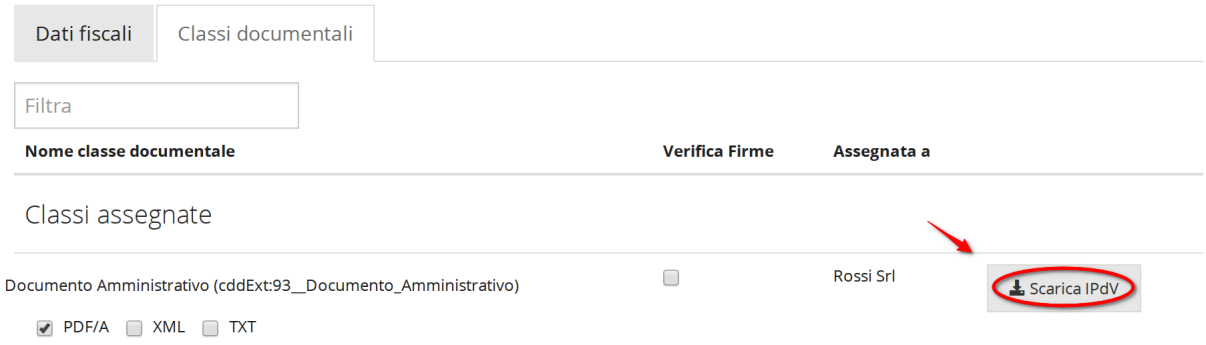
Un esenpio concreto di file IPdV relativo alla classe documentale assegnata dal sistema all'archivio può essere recuperato tramite l'interfaccia web:

- una volta entrati nel sistema cliccare sul bottone ***Visualizza*** accanto all'archivio di riferimento

- cliccare sul tab ***Classi documentali*** della pagina di dettaglio dell'archivio



- cliccare sul bottone ***Scarica IPdV*** accanto alla classe documentale di riferimento



- in questo modo si attiverà il download di un IPdV ben formato in cui vanno esclusivamente cambiati i valori definitivi da inserire a sistema.

Nella compilazione di un IPdV le regole principali di cui tenere conto sono:

- è importante che ogni IPdV abbia un naming specifico. Esso è

IPdV-<qualsiasi cosa>.xml

- i valori da aggiornare all'interno dell'IPdV hanno tutti il prefisso *@ph_*, tutti gli altri dati vanno lasciati così come sono
- di particolare rilevanza è il tag ***pdvid*** che deve essere unico all'interno dell'archivio
- tutti i riferimenti alle date vanno inseriti nello standard ISO8601. Sono ammessi i formati sotto elencati (le parti in grassetto sono parte del formato e vanno riportate così come sono):

Tipologia	Formato	Esempio
Completa	aaaa**-**mm**-**gg**T**hh24**:**min**:**ss**.**milli**Z**	2014-11-28T00:00:00.000Z
Semplificata	aaaa-mm-gg**T**hh24**:**min**:**ss	2014-11-28T00:00:00
Solo data	aaaa**-**mm**-**gg	2014-11-28

- l'impronta dei file da mettere in conservazione è un **hash in SHA256 codificato in base64**. Per verificare durante il testing che l'hash sia quello atteso da noi si può fare un test con questo strumento online ONLINE SHA1, SHA-256, SHA-384, SHA-512 HASH CALCULATOR e confrontare il proprio elaborato con il risultato che segue il prefisso SHA-256(Base64)
- l'elenco dei metadati obbligatori, nel senso che devono essere presenti nell'IPdV eventualmente anche con valore vuoto, è quello contenuto all'interno del tag

8. Appendice B: notifiche tramite callback url

Il flusso di versamento e conservazione prevede l'invio di notifiche opportune a indicare situazioni di errore o a confermare il corretto ciclo di vita dei Pacchetti di Archiviazione.

Il sistema in automatico invia tali notifiche via PEC ai corretti destinatari.

E' possibile ricevere queste notifiche, sempre in modalità push, fornendo al sistema una url a cui inviarle. La configurazione di tale url è demandata all'utente accedendo dall'interfaccia web, cosicché egli possa in totale autonomia variarla in fase di manutenzione.

Una volta effettuata l'autenticazione è sufficiente cliccare sul bottone *Visualizza* posto accanto all'archivio per cui si vuole configurare la url.

Nella pagina di gestione dell'archivio, sotto il tab *Dati fiscali* è presente la sezione *Settaggi archivio*. Una volta aperta tale sezione cliccando sull'iconcina a forma di freccia posta alla sua destra, si ha accesso alla configurazione della url di callback.

Da questo momento in poi, oltre alle consuete comunicazioni via PEC, il sistema invia tutte le notifiche presso tale url in formato json.

Le notifiche sono di vario tipo ma risultano particolarmente interessanti quelle relative alle fasi di versamento. In particolare ci sono notifiche relativamente agli errori riscontrati in fase di validazione degli IPDV e dei file versati e notifiche relative al buon esito come la generazione dei Rapporti di Versamento o o del Rapporto di Conservazione.

Il formato delle notifiche via callaback è Json ed è così costituito:

- **version**: identifica il formato della notifica
- **timestamp**: ovvero l'istante in cui viene trasmessa la notifica espresso nel numero di millisecondi dalla mezzanotte del 01/01/1970
- **type**: identifica la tipologia di notifica
- **payload**: restituisce altre informazioni di dettaglio

Tra i *type* interessanti di particolare rilievo ci sono i seguenti:

- **PDVADDED**: quando un updv viene aggiunto regolarmente
- **PDVVALIDATOR**: ogni volta che c'è un errore nella validazione dell'IPdV in caricamento
- **FILEADDED**: quando fa l'auditing di tipo "file added"

P.I. 01879020517 C.F. e R.I./AR 01879020517 REA 145843 C.S. € 6.500.000 i.v.
società con socio unico

- **FILEADDED**: quando manda la mail di "Versamento incorretto" (e nel caso nel payload c'è anche l'errore)
- **PDVCOMPLETIONCHECK**: quando viene inviata la mail di "Rapporto di versamento del PdA [pdaName] relativo a [nomeIPdV]"
- **PDAGENERATOR**: quando manda la mail "Rapporto di conservazione del PdA [PdAName]"

Esempio di notifica di errore

""" Esempio di uploadQueue su DocFly """

Esempio di Rapporto di Versamento

""" Esempio di uploadQueue su DocFly """

Esempio di Rapporto di Conservazione

""" Esempio di uploadQueue su DocFly """

9. Appendice C: esempi di codice con chiamate servizio REST

NOTA BENE: i seguenti esempi sono realizzati in Python 2

ESEMPIO DI VERSAMENTO

Il seguente esempio mostra come fare a creare un PdA e versare al suo interno un IPdV e un documento.

```
""" Esempio di uploadQueue su DocFly """
import requests


def upload_queue(name_pda, filename, filepath):
""" Esempio di uploadQueue su DocFly """

    endpoint = 'https://collaudo.docfly.it/api/uploadQueu
    username = 'myusername
    password = 'mypassword'
    archive = 'codice_fiscale_archivio'

    #
    # invio file
    #

    # setting dati di invio
    params = {
        'username: username,
        'password': password,
```

```
        'owner': archive,
        'destination': name_pda,
        'filename: filename,
        'files': {
            'filedata': (filepath, open(filepath, 'rb')),
        }
    }

    files = params.pop('files')
    response = requests.post(endpoint, data=params, files=files)

    # Se ok, response.json() contiene
    # {u'status': u'success', u'data': u'Upload successfull'}

    return response.json()

if __name__ == '__main__':

    RESPONSE_IPdV = upload_queue('nome_pda', 'IPdV-qualcosa.xml', 'path_file_IPdV')

    if RESPONSE_IPdV['status'] == 'success':

        # Se ok, viene stampato
        # {u'status': u'success', u'data': u'Upload successfull'}

        print upload_queue('nome_pda', 'file_di_test.txt', 'path_file_di_test.txt')
```

ESEMPIO DI CANCELLAZIONE PDA

Il seguente esempio mostra come cancellare un PDA non ancora conservato.

NOTA BENE: l'operazione di cancellazione PDA è irreversibile

```
""" Esempio di deletePDA su DocFly """
import requests
def login(username, password):
    """ Effettua login """

    endpoint = 'https://collaudo.docfly.it/api/signin'
    params = {
        'username: username,
        'password': password,
    }
    response = requests.post(endpoint, data=params)
    json = response.json()

    ticket = json['data']['data']['ticket']
    home_node_ref = json['data']['homeNodeRef']

    return (ticket, home_node_ref)

def deletePDA(ticket, node_ref):
""" Recupera le informazioni del nodeRef """

    endpoint = 'https://collaudo.docfly.it/api/deletePDA'
    params = {
```

```
        'ticket': ticket,
        'nodeRef': node_ref,
        }
    response = requests.post(endpoint, params=params)

    # La risposta restituisce un json
    print response.json()

def logout(ticket):
    """ Esegue la signout. Ritorna True se OK, False altrimenti. """

    endpoint = 'https://collaudo.docfly.it/api/signout'
    params = {
        'ticket': ticket
    }
    return requests.delete(endpoint, params=params).json()

if __name__ == '__main__':
    # effettua il login e recupera ticket di sessione e nodeRef della home
    TICKET, HOME_NODE_REF = login('prod1', 'prod1')
    print 'Login OK: ticket=' + TICKET
    print

    # recupera tutte le informazioni del nodeRef
    deletePDA(TICKET, 'workspace://SpacesStore/7cb34802-7e24-49e0-957a-
3d2a8cd8e3d7')
    print
    # effettua il logout
```

```
    print 'Logout ' + logout(TICKET)['status']
```

ESEMPIO DI BROWSING

Il seguente esempio mostra come fare ad autenticarsi e ricavare le informazioni della home del sistema di conservazione

```
""" Esempio di viewRepository su DocFly """
import requests
def login(username, password):
    """ Effettua login """

    endpoint = 'https://collaudo.docfly.it/api/signin'
    params = {
        'username: username,
        'password': password,
    }
    response = requests.post(endpoint, data=params)
    json = response.json()

    ticket = json['data']['data']['ticket']
    home_node_ref = json['data']['homeNodeRef']

    return (ticket, home_node_ref)

def view_repository(ticket, node_ref):
```

```
""" Recupera le informazioni del nodeRef """

    endpoint = 'https://collaudo.docfly.it/api/viewRepository'
    params = {
        'ticket': ticket,
        'nodeRef': node_ref,
        'pag: 1,
        }
    response = requests.get(endpoint, params=params)

    # La risposta restituisce un json
    print response.json()

def logout(ticket):
    """ Esegue la signout. Ritorna True se OK, False altrimenti. """

    endpoint = 'https://collaudo.docfly.it/api/signout'
    params = {
        'ticket': ticket
    }
    return requests.delete(endpoint, params=params).json()

if __name__ == '__main__':
    # effettua il login e recupera ticket di sessione e nodeRef della home
    TICKET, HOME_NODE_REF = login('prod1', 'prod1')
    print 'Login OK: ticket=' + TICKET
    print
```

```
    # recupera tutte le informazioni del nodeRef
    view_repository(TICKET, 'workspace://SpacesStore/7cb34802-7e24-49e0-957a-3d2a8cd8e3d7')
    print
    # effettua il logout
    print 'Logout ' + logout(TICKET)['status']
```

ESEMPIO DI SEARCH

Il seguente esempio mostra come fare ad autenticarsi e ricercare un documento tramite un suo metadato

```
""" Esempio di search su DocFly """
import requests
def login(username, password):
    """ Effettua login """

    endpoint = 'https://collaudo.docfly.it/api/signin'
    params = {
        'username: username,
        'password': password,
    }
    response = requests.post(endpoint, data=params)
    json = response.json()

    ticket = json['data']['data']['ticket']
    home_node_ref = json['data']['homeNodeRef']
```

```
    return (ticket, home_node_ref)


def search(params):
""" Recupera le informazioni del nodeRef """

    endpoint = 'https://collaudo.docfly.it/api/search'


    response = requests.post(endpoint, params=params)


    # La risposta restituisce un json
    print response.json()


def logout(ticket):
    """ Esegue la signout. Ritorna True se OK, False altrimenti. """


    endpoint = 'https://collaudo.docfly.it/api/signout'
    params = {
        'ticket': ticket
    }
    return requests.delete(endpoint, params=params).json()


if __name__ == '__main__':
    # effettua il login e recupera ticket di sessione e nodeRef della home
    TICKET, HOME_NODE_REF = login('prod1', 'prod1')
    print 'Login OK: ticket=' + TICKET
    print
```

```
# recupera tutte le informazioni del nodeRef


params = {
   'ticket': TICKET,
    'className': 'cddExt:1006__Documento_informatico',
    'conservazione.doc:filename': 'nomeMioFile'
}
search(params)
print
# effettua il logout
print 'Logout ' + logout(TICKET)['status']
```

ESEMPIO DI GETDISPOSE

Il seguente esempio mostra come fare a controllare se ci sono errori in un pacchetto di archiviazione

```
""" Esempio di getDispose su DocFly """
import requests
def getDispose(username, password, owner, destination):
   endpoint = 'https://collaudo.docfly.it/api/getDispose'
   params = {
      'username: username,
      'password': password,
      'owner': owner,
      'destination': destination
      }
```

```
    response = requests.get(endpoint, params=params)



if __name__ == '__main__':
    # recupera info sul PDA nomePDA
    getDispose('prod1', 'prod1', 'AAABBB11C22D333E', 'nomePDA')
    print
```

ESEMPIO DI ESIBIZIONE PDD

Il seguente esempio mostra come fare a richiedere l'esibizione di un PdD

```
""" Esempio di viewRepository su DocFly """
import requests
def login(username, password):
    """ Effettua login """

    endpoint = 'https://collaudo.docfly.it/api/signin'

    params = {
        'username: username,
        'password': password,
    }

    response = requests.post(endpoint, data=params)
    json = response.json()

    ticket = json['data']['data']['ticket']
```

```
    home_node_ref = json['data']['homeNodeRef']

    return (ticket, home_node_ref)

def distribute(ticket, node_ref):
    """ Download del nodeRef """

    endpoint = 'https://collaudo.docfly.it/api/distribute'
    params = {
        'ticket': ticket,
        'nodeRef': node_ref,
        }

    response = requests.get(endpoint, params=params)

    # Viene restituito il PdD in formato .zip
    return response

def logout(ticket):
    """ Esegue la signout. Ritorna True se OK, False altrimenti. """

    endpoint = 'https://collaudo.docfly.it/api/signout'
    params = {
        'ticket': ticket
    }

    return requests.delete(endpoint, params=params).json()
```

```
if __name__ == '__main__':
    # effettua il login e recupera ticket di sessione e nodeRef della home
    TICKET, HOME_NODE_REF = login('prod1', 'prod1')
    print 'Login OK: ticket=' + TICKET
    print

    # download del PdD
    NODE_REF = 'workspace://SpacesStore/id_unico_nodeRef'
    distribute(TICKET, NODE_REF)
    print

    # effettua il logout
    print 'Logout ' + logout(TICKET)['status']
```