Descrizione API

Upload Fatture

Le tipologie di documenti inviabili sono definite "Flussi" sono visibili qui. Ogni flusso ha i suoi stati.

L’upload di un xml non valido rispetto allo schema xsd avrà come riposta un errore 400 (la fattura non viene quindi salvata su TSDigital).

Per i flussi SDIPA, SDIPR e SELFINV verranno effettuati anche i controlli da parte dello SDI e il mancato superamento impedirà l’upload con un errore 400. Anche in questo caso la fattura non viene salvata su TSDigital.

Nell'ambiente di test è possibile inviare notifiche pilotando l'esito di risposta SDI. Vedi Mock Sdi.

E' quindi importante gestire correttamente gli errori in risposta poiché in futuro questi controlli potrebbero essere estesi.

Upload Fattura

> POST/v2/invoices

Esempio chiamata:

```
curl --location --request POST 'https://b2bwrite-api-test.agyo.io/api/v2/invoices' \
--header 'Authorization: Bearer' \
--header 'accept: application/json;charset=utf-8' \
--header 'User-Agent: Postman' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
--header 'X-Request-ID: Codice Univoco (cuid)' \
--header 'X-Correlation-ID: Codice Univoco (cuid)' \
--header 'X-Item-ID: ItemID Azienda' \
--header 'X-User-ID: ID Chiave Tecnica' \
--header 'Content-Type: application/json;charset=utf-8' \
--data-raw '{
"transmitterId": "ItemId Azienda",
"senderId": "ItemId Azienda",
"flowType": "SDIPR",
```

```
"fileName": "nome file XML",
"content": "file XML in base64"
}'
```

Esempio risposta upload OK (Status Code 201):

```
{
    "hubId": "5e980d7f9715a77a998d0047",
    "link": {
        "rel": "self",
        "href": "https://b2bread-api-test.agyo.io/api/v2/invoices/5e980d7f9715a77a998d0047"
    }
}
```

> In caso di risposta positiva 201 potrebbe essere necessario fino a qualche secondo affinché la fattura sia disponibile in lettura, questo per via dei tempi di replica dell'informazione sulle diverse repliche del database. Per questioni di performance il server ritorna il 201 quando il dato è replicato sulla metà + 1 delle repliche, questo consente di garantire il salvataggio del dato e una minor attesa del client.

> Una volta effettuato l'upload evitare di chiedere immediatamente lo stato ma attendere almeno 5 minuti prima di tentare aggiornamenti di stato. Il processo asincrono sebbene sia spesso immediato può richiedere più tempo in situazioni di alto carico. Chiamate a vuoto non fanno altro che alzare il carico

Esempio di risposta upload XML non valido, controllo effettuato per tutti i flussi (Status Code 400):

```
{
    "status": "BAD_REQUEST",
    "code": "400",
    "timestamp": "16-04-2020 07:48:51",
    "message": "SCARTATO - XML della fattura non conforme alla linea 14 e colonna 33. Motivo
dello scarto: cvc-complex-type.2.4.a: contenuto non valido che inizia con l'elemento
\"CessionarioCommittente\". È previsto un elemento \"{CedentePrestatore}\".",
    "subErrors": null
}
```

Esempio risposta mancato superamento controlli SDI, controlli effettuati solo per i flussi SDIPA, SDIPR, SELFINV (Status Code 400):

```
{
    "status": "BAD_REQUEST",
    "code": "400",
    "timestamp": "16-04-2020 07:54:45",
    "message": "Fattura non valida. Errori: [00400 - Sulla riga di dettaglio con aliquota IVA pari a zero deve essere presente il campo Natura]",
    "subErrors": null
}
```


Esempio risposta upload XML FPA12 ma indicando nel payload flusso SDIPR:


```
{
    "status": "BAD_REQUEST",
    "code": "400",
    "timestamp": "16-04-2020 08:01:05",
    "message": "Non è possibile utilizzare il formato trasmissione FPR12 quando il flusso è SDIPA",
    "subErrors": null
}
```

Controllo XML

> POST/v2/invoices/validate

Questo endpoint permette di effettuare la validazione di un xml sia sulla base dello schema xsd sia sulla base dei controlli formali effettuati dal Sistema di Interscambio.

Payload di richiesta


```
{
"content": "xml della fattura in base64"
}
```


Payload di risposta


```
{
  "sdiErrors": [
    "string"
  ],
  "xsdError": "string"
```

```
}
```

Se l'xml è corretto e valido rispetto ai controlli formali SdI si otterrà in risposta un json in cui l'array `sdiErrors` è vuoto ed il campo `xsdError` è null.

```
{
    "sdiErrors": [],
    "xsdError": null
}
```

Ovviamente in caso fallisca la validazione xsd non sarà possibile effettuare i controlli formali, si avrà pertanto una risposta in cui sarà valorizzato solo il campo `xsdError`. Esempio:

```
{
    "sdiErrors": null,
    "xsdError": "XML della fattura non conforme alla linea 5 e colonna 28. Motivo dello scarto: cvc-complex-type.2.4.a: contenuto non valido che inizia con l'elemento \"CedentePrestatore\". È previsto un elemento \"{DatiTrasmissione}\"."
}
```

Di seguito un payload ottenuto in risposta in seguito al controllo di un xml valido rispetto allo schema xsd ma formalmente errato:

```
{
    "sdiErrors": [
        "00423 - Il valore del campo PrezzoTotale non risulta calcolato secondo le regole definite nelle specifiche tecniche",
        "00422 - Il valore del campo ImponibileImporto non risulta calcolato secondo le regole definite nelle specifiche tecniche",
        "00421 - Il valore del campo Imposta non risulta calcolato secondo le regole definite nelle specifiche tecniche"
    ],
    "xsdError": null
}
```

Simulazione invio a SDI

SDI Mock

In ambiente di test è disponibile un Mock dei servizi SDI che permette la simulazione dei differenti tipi di notifiche.

le notifiche vengono consegnate dopo un determinato delay in caso di notifiche multiple, fra una notifica e l'altra vi è un attesa pari al NOTIFICATION_DELAY configurato sul server (ora a 3 minuti). I tipi di notifiche supportate sono i seguenti :

- ACCETTATO
- ACCETTATO_CON_RITARDO
- RIFIUTATA
- SCARTATO
- NESSUNA_RISPOSTA
- NON_RECAPITATO

Il tipo deve essere specificato in un tag <Causale>?</Causale> all'interno della fattura

> Se non viene specificato nulla nel tag causale la fattura viene considerata come in stato ACCETTATO

Forward della fattura

Se si vuole che il mock una volta ricevuta la fattura faccia il forward e la consegni quindi all'azienda indicata come cessionario occorre specificare i seguenti codici destinatari:

Per le SDIPR:

`AGYOFWD`

Per le SDIPA

`UFPRT0`

> Occorre specificare nel cessionario in fattura piva o cf del cessionario presente su digital

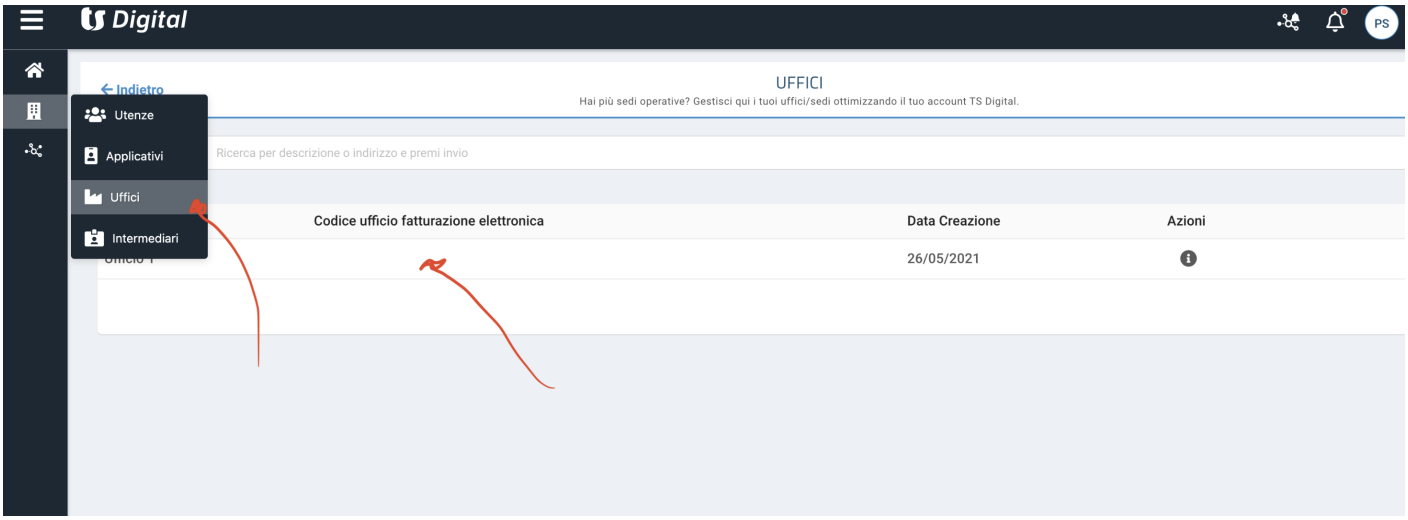
Forward della fattura con assegnazione su ufficio specifico

utilizzare:

`AGYOFW` + una lettera. ad ES. `AGYOFWA`, `AGYOFWB`

per fare in modo che la fattura venga assegnata all'ufficio corretto va poi associato l'ufficio al codice destinatario da portale.

(Attenzione. non è più fattibile assegnare un codice a piacere da portale)



Caricamento notifiche

E' possibile allegare notifiche ad una fattura in due modi:

Contemporaneamente al caricamento fattura

https://b2bwrite-api-test.agyo.io/api/swagger-ui/index.html?configUrl=/api/v3/api-docs/swagger-config#/InvoicesV2/uploadInvoice

> POST /v2/invoices

> Il filename deve essere strutturato seguendo il naming SDI

```
...

"attachments": [
    {
      "fileName": "string",
      "content": "base64"
    }
  ],

...
```

Dopo il caricamento fattura

https://b2bwrite-api-test.agyo.io/api/swagger-ui/index.html?configUrl=/api/v3/api-docs/swagger-config#/InvoicesV2/addNotification

> PATCH /v2/invoices/{hubId}/addNotification

Attenzione che il metodo HTTP è PATCH in quando va a modificare un entità già presente

> Il filename deve essere strutturato seguendo il naming SDI

```
{
  "notifications": [
    {
      "fileName": "string",
      "content": "base64"
    }
  ]
}
```

Conservazione Digitale (CCT)

Al caricamento di una fattura con già le notifiche allegate il processo segue il flusso normale, per quanto riguarda le notifiche aggiunte in un secondo momento verrà prevista la generazione automatica di un evento per tentarne la conservazione

ItemOverview

Questa API consente di verificare rapidamente la presenza di fatture ricevute, inviate e scartate per una o più aziende a partire da un dato timestamp.

Il comportamento è identico alla vecchia companyOverview, è stata aggiornata la logica relativa alla paginazione (adeguandola al continuationToken) e la nomenclatura dei campi per renderla coerente con l'itemId. In query string è possibile indicare la dimensione della pagina attraverso il parametro size (valore massimo ammesso 1000) è consigliato usare valori più bassi come 100 e sfruttare il meccanismo del cotinuation token per recuperare le pagine successive. questo permette migliori performance e riduce il rischio di rallentare il db.

I valori di *lastTimestampActive* e *lastTimestampPassive* nel payload di richiesta devono essere espressi in millisecondi e non possono essere antecedenti al 01/01/2019 nè successive al timestamp corrente.

Funzionamento

Aggiornamento fatture passive

Quando TSDigital assegna una fattura ricevuta dal Sistema di Interscambio viene prodotto un evento che, **quando verrà processato**, aggiornerà le informazioni del campo *lastTimestampPassive* con il timestamp di assegnazione della fattura.

Esempio:

- TSDigital assegna la fattura all'azienda con codice fiscale ABCDEF e itemId adcf9209-fe9b-4b37-9b68-8238da8fd6bb alle 15:30 del 15/04/2020 (UTC), corrispondente al timestamp 1586964600000
- Viene prodotto l'evento
- TSDigital processa l'evento precedente alle 17:00 del 15/04/2020 (UTC), corrispondente al timestamp 1587081600000
- Il valore di *lastTimestampPassive* per l'itemId adcf9209-fe9b-4b37-9b68-8238da8fd6bb sarà dunque 1586964600000 (orario di assegnazione della fattura) e non 1587081600000 (orario in cui è stato processato l'evento)

Aggiornamento fatture attive/scartate

Vale quanto detto in precedenza, con l'unica differenza che l'evento di aggiornamento dell'itemOverview viene prodotto a partire da un cambio stato della fattura e produrrà un cambiamento per i valori di *lastTimestampActive* e *lastTimestampRejected*.

Funzionamento delle API

Per semplicità prendiamo in esame un payload di richiesta relativo ad un solo item così definito:

```
{
    "items": [
        {
            "itemId": "ABCDEF",
            "lastTimestampActive": 1583143700735, // 2 Marzo 2020, 10:08:20 UTC
            "lastTimestampPassive": 1583244701735 // 3 Marzo 2020, 14:11:51 UTC
        }
    ]
}
```

L'itemId *ABCDEF* su database contiene queste informazioni:

```
{
    "itemId": "ABCDEF",
    "taxId": "XYZ",
    "lastTimestampActive": 1586159941109, // 6 Aprile 2020, 07:59:01 UTC
    "lastTimestampPassive": 1583251200000, // 3 Marzo 2020, 16:00:00 UTC
    "lastTimestampRejected": 1584378112720, // 16 Marzo 2020, 17:01:52 UTC
    [...]
}
```

I flag saranno calcolati così:

```
boolean active = lastTimestampActive > lastTimestampActiveFromRequest;
boolean rejected = lastTimestampRejected > lastTimestampActiveFromRequest;
boolean passive = lastTimestampPassive > lastTimestampPassiveFromRequest;

// Utilizzando i dati di esempio:

// 6 Aprile 2020, 07:59:01 UTC > 2 Marzo 2020, 10:08:20 UTC => true
boolean active = 1586159941109 > 1583143700735

// 16 Marzo 2020, 17:01:52 UTC > 2 Marzo 2020, 10:08:20 UTC => true
boolean rejected = 1584378112720 > 1583143700735

// 3 Marzo 2020, 16:00:00 UTC > 3 Marzo 2020, 14:11:51 UTC => true
boolean passive = 1583251200000 > 1583244701735
```

e produrranno quindi il seguente payload di risposta:

```
[...]
{
  "active": true,
  "passive": true,
  "rejected": true,
    "lastTimestampActive": 1586159941109, // 6 Aprile 2020, 07:59:01 UTC
    "lastTimestampPassive": 1583251200000, // 3 Marzo 2020, 16:00:00 UTC
    "lastTimestampRejected": 1584378112720, // 16 Marzo 2020, 17:01:52 UTC
  "itemId": "ABCDEF",
  "taxId": "XYZ"
}
[...]
```

Utilizzo

> Ad esclusione della prima volta in cui si utilizza l'API per garantire il corretto comportamento è **fondamentale** utilizzare come valori di lastTimestampActive e lastTimestampPassive in request quelli ottenuti dalla response precedente di company overview.

Effettuare una prima chiamata all'API utilizzando un valore valido a piacere per i campi *lastTimestampActive* e *lastTimestampPassive*, ad esempio 1577836800000 (1 Gennaio 2020, 00:00:00 UTC).

Ottenuta la prima response è opportuno memorizzare i valori di *lastTimestampActive* e *lastTimestampPassive* che verranno utilizzati per la successiva chiamata di company Overview.

Riprendendo l'esempio di prima ed ipotizzando che la situazione delle fatture sia rimasta invariata per l'itemId ABCDEF la nuova chiamata avrà come payload di request:

```
{
    "items": [
        {
            "itemId": "ABCDEF",
    "lastTimestampActive": 1586159941109, // 6 Aprile 2020, 07:59:01 UTC ottenuto dalla
response precedente
    "lastTimestampPassive": 1583251200000, // 3 Marzo 2020, 16:00:00 UTC ottenuto dalla
response precedente
        }
    ]
```

```
}
```

e come response

```
[...]
{
  "active": false,
  "passive": false,
  "rejected": false,
    "lastTimestampActive": 1586159941109, // 6 Aprile 2020, 07:59:01 UTC
    "lastTimestampPassive": 1583251200000, // 3 Marzo 2020, 16:00:00 UTC
    "lastTimestampRejected": 1584378112720, // 16 Marzo 2020, 17:01:52 UTC
  "itemId": "ABCDEF",
  "taxId": "XYZ"
}
[...]
```

Request/Response

Esempio di payload di richiesta con una post a *https://b2bread-api-test.agyo.io/api/v2/items/overview?size=1*

```
{
    "items": [
        {
            "itemId": "ABCDEF00G00H000L",
            "lastTimestampActive": 1583143700735,
            "lastTimestampPassive": 1583143700735
        },
        {
            "itemId": "6f51950a-c982-40b7-99a3-405dd1328684",
            "lastTimestampActive": 1583143700735,
            "lastTimestampPassive": 1583143700735
        }
    ]
}
```

Esempio di payload di risposta:

```
{
    "_embedded": {
```

```
        "itemOverviewList": [
            {
                "active": true,
                "passive": true,
                "rejected": true,
                "lastTimestampActive": 1586159941109,
                "lastTimestampPassive": 1583243470858,
                "lastTimestampRejected": 1584378112720,
                "itemId": "ABCDEF00G00H000L",
                "taxId": "ABCDEF00G00H000L"
            }
        ]
    },
    "_links": {
        "first": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/items/overview?size=1&sort=agyoCompanyId,desc"
        },
        "self": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/items/overview?size=1&sort=agyoCompanyId,desc"
        },
        "next": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/items/overview?continuationToken=33dbhiloNf1U8m9EPLLkKceN2NROiLWgzLNh6mnbF
LD5MDEXZ9jWsKRnn5622zJqRbP44Twq4oh_XrYO2zsulG3tDO5ipcarnPv6gjQk6xqnd0x_BShZwfqTmItxsAlz&size=1
&sort=agyoCompanyId,desc"
        }
    },
    "page": {
        "size": 1,
        "hasNext": true,
        "continuationToken":
"33dbhiloNf1U8m9EPLLkKceN2NROiLWgzLNh6mnbFLD5MDEXZ9jWsKRnn5622zJqRbP44Twq4oh_XrYO2zsulG3tDO5ip
carnPv6gjQk6xqnd0x_BShZwfqTmItxsAlz"
    }
}
```


Nel payload di richiesta sono presenti due item ed in queryString è stata richiesta la pagina di un elemento ci aspettiamo dunque la presenza di ulteriori pagine, come confermato dal payload di

risposta `(page.haseNext=true)`. Possiamo accedere alla pagina successiva utilizzando il link presente in `_links.next` (non costruire il link a mano ma utilizzare quello ottenuto in risposta). In questo caso effettueremo una post a *https://b2bread-api-test.agyo.io/api/v2/items/overview?continuationToken=33dbhiloNf1U8m9EPLLkKceN2NROiLWgzLNh6mnbFLD5MDEXZ9jWsKRnn5622zJqRbP44Twq4oh_XrYO2zsulG3tDO5ipcarnPv6gjQk6xqnd0x_BShZwfqTmItxsAlz&size=1&sort=agyoCompanyId,desc*

Payload di risposta:


```
{
    "_embedded": {
        "itemOverviewList": [
            {
                "active": true,
                "passive": false,
                "rejected": false,
                "lastTimestampActive": 1583243700735,
                "lastTimestampPassive": 1563982802310,
                "itemId": "6f51950a-c982-40b7-99a3-405dd1328684",
                "taxId": "03200510166"
            }
        ]
    },
    "_links": {
        "self": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/items/overview?continuationToken=33dbhiloNf1U8m9EPLLkKceN2NROiLWgzLNh6mnbF
LD5MDEXZ9jWsKRnn5622zJqRbP44Twq4oh_XrYO2zsulG3tDO5ipcarnPv6gjQk6xqnd0x_BShZwfqTmItxsAlz&size=1
&sort=agyoCompanyId,desc"
        }
    },
    "page": {
        "size": 1,
        "hasNext": false,
        "continuationToken": null
    }
}
```


Dal payload di risposta notiamo che non ci sono ulteriori pagine `(page.haseNext=false)`.

Lettura Fatture

Le chiamate per la lettura delle fatture si dividono in due step.

Questo non vuol dire però che sono complementari, ovvero, le chiamate di lettura fatture (step 2) possono essere effettuate anche senza lo step 1.

Perché è importante però utilizzare lo step 1?

Lo step 1, ovvero la chiamata di **ItemOverview**, è importante perché ci consente di verificare rapidamente se ci sono delle nuove fatture.

Questo permette di non sovraccaricare i nostri server perché se chiamando l'ItemOverview (step 1) ci restituisce che non ci sono delle nuove fatture è inutile procedere con lo step 2.

Dopo aver effettuato la chiamata di ItemOverview possiamo procedere a chiamare le api di lettura per ogni item che ha fatture.

Lista Fatture

> GET/v2/invoices

Parametri da impostare:

- **ownerId:** da popolare con l'identificativo dell'azienda (UUID/CF/P.iva) Se l'azienda ha più di un ufficio e si vogliono recuperare le fatture di tutti gli uffici è possibile aggiungere al valore di ownerId il suffisso **-ALL**.
- **active:** da popolare con *true* o *false.* True se si vuole ottenere la lista delle fatture attive. False se si svuole ottenere la lista. delle fatture passive;
- **lastTimestampFrom:** da popolare con la data espressa in millisecondi. Ad eccezione della prima volta che può essere popolato con un data a piacere le volte successive deve essere popolato con il "*lastTimestamp*" della fattura più recente ricevuta dalla response.

Come impostare la chiamata:

Eseguo la prima chiamata con data a piacere, ad esempio 24/01/2022
Dalla response ottengo tutte le fatture che hanno subito aggiornamenti successivi alla data impostata.
Le fatture vengono restituite dalla più recente alla meno recente.

Il giorno seguente ho bisogno di sapere se ci sono altri aggiornamenti sulle fatture?
Quello che bisogna fare è fare è impostare nella chiamata il timestamp dell'ultima fattura ottenuta dalla response precedente.

Esempio:

24/01/2022 ore 11:00 timestamp: 1643022047000

Response:

```
"…
"currentStatusName": "INVIATO",
"lastTimestamp": 1643047247000,
 "active": true,
 …"
```

25/01/2022 ore 11:00 timestamp: 1643047247000

In questo modo dalla response otterremo solo le fatture che hanno subito un aggiornamento e non nuovamente tutta la lista delle fatture.

Il comportamento ideale da seguire quindi è sincronizzare le fatture tramite questa chiamata e poi impostare delle query da parte vostra per ottenere specifici risultati.

> **E' possibile trovare la stessa fattura con *timestamp* diversi?**
> Sì è possibile perché il *timestamp* di una fattura viene modificato ogniqualvolta essa subisce un aggiornamento.

Continuation Token

Il "continuationToken" ci permette di vedere, qualora siano presenti, le altre fatture ottenute dalla response.
Il "size" della chiamata è impostato di default a 20 in modo da ottenere dalla response non più di 20 record (fatture).
Nel caso in cui siano presenti più di 20 fatture, in fondo alla response, troveremo due tag che ne danno conferma e sono "hasNext" popolato con "true" e "continuationToken" popolato con una stringa.

Esempio:

```
},
    "page": {
        "size": 20,
        "hasNext": true,
```

```
        "continuationToken":
"6g5TtcULxhoHlBd9DlTOmnTlqtnDa5Hz8zlimF7R1ahgTkS2fOYE1z3HM162btsWr8iV0xi2Pbd02nHD7XlwFX3Wtt2cA
5iTxLJ07-zpwUXV1WJ9HAd9ko3t22M3tjjc"
    }
```

Quindi per visualizzare le altre fatture presenti nella response non dobbiamo far altro che aggiungere alla chiamata il parametro “continuationToken” e popolarlo con la stringa indicata.

Esempio di chiamata:

```
curl --location --request GET 'https://b2bread-api-
test.agyo.io/api/v2/invoices?ownerId=bf01b67b-86c6-4695-b8c3-
2135aa0b8d6c&active=true&lastTimestampFrom=1546335945000' \
--header 'User-Agent: Postman' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
--header 'Content-Type: application/json' \
--header 'X-Request-ID: cuid(codice unvico)' \
--header 'X-Correlation-ID: cuid(codice univoco)' \
--header 'X-Item-ID: UUID (identificativo azienda)' \
--header 'X-User-ID: ID Chiave Tecnica' \
--header 'Accept-Language: it-IT' \
--header 'Authorization: Bearer '
```

Dettaglio Fattura

> GET/v2/invoices/{hubId}

Esempio chiamata:

```
curl --location --request GET 'https://b2bread-api-test.agyo.io/api/v2/invoices/{hubId}' \
--header 'accept: application/hal+json' \
--header 'User-Agent: Postman' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
--header 'X-Request-ID: cuid (codice univoco)' \
--header 'X-Correlation-ID: cuid (codice univoco)' \
--header 'X-Item-ID: Identificativo azienda' \
--header 'X-User-ID: ID Chiave Tecnica' \
```

```
--header 'Authorization: Bearer '
```

Download Fattura

> GET/v2/invoices/{hubId}/download

Permette il download della fattura così come ha raggiunto l'ultimo stato. Questo significa che il download di una fattura inviata a SDI scaricherà una fattura firmata XADES (a meno che l'azienda abbia il flag di invio senza firma) con terzo intermediario TS (a meno che la fattura non sia già stata firmata in upload)

Mentre per la fattura ricevuta da SDI si riceverà una fattura non firmata, quindi in semplice formato xml. Se la fattura fa parte di un lotto si riceve il solo xml della fattura a cui si fa riferimento in request in quanto essa viene scorporata dal lotto per facilità d'uso.

Per i flussi che non passano per SDI si riceve l'xml della fattura relativa all'ultimo stato, quindi se previsto l'embedding degli allegati conterrà anche quelli.

Download del file in PDF: https://b2bread-api-test.agyo.io/api/v2/invoices/{hubId}/download?format=PDF

Download del file in ASSOSOFTWARE: https://b2bread-api-test.agyo.io/api/v2/invoices/{hubId}/download?format=ASSOSW

Download del file in XML: https://b2bread-api-test.agyo.io/api/v2/invoices/{hubId}/download?format=XML

Esempio chiamata:

```
curl --location --request GET 'https://b2bread-api-test.agyo.io/api/v2/invoices/{hubId}/download?format=XML' \
--header 'accept: application/hal+json' \
--header 'User-Agent: Postman' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
--header 'X-Request-ID: cuid (codice univoco)' \
--header 'X-Correlation-ID: cuid (codice univoco)' \
--header 'X-Item-ID: UUID (identificativo azienda)' \
--header 'X-User-ID: ID Chiave Tecnica' \
--header 'Authorization: Bearer '
```

>

GET/v2/invoices/{hubId}/download/original

Permette il download della fattura originale, quindi per una fattura inviata a SDI permetter di scaricare la fattura così come è stata caricata dall'utente.

Mentre per le passive ricevute da SDI permette di scaricare la fattura originale ricevuta, firmata XADES o PADES (p7m).

Per i flussi che non passano per SDI si riceve l'xml caricato dall'utente senza alcuna modifica.

Se la fattura in request fa riferimento ad un lotto si riceverà il lotto intero, in quanto la fattura si trova all'interno di quel file originale.

Download Notifica SDI

GET/v2/invoices/{hubId}/downloadAllMessages

Esempio Chiamata:

```
curl --location --request GET 'https://b2bread-api-
test.agyo.io/api/v2/invoices/{hubId}/downloadAllMessages' \
--header 'accept: application/hal+json' \
--header 'User-Agent: Postman' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
--header 'X-Request-ID: cuuid' \
--header 'X-Correlation-ID: uuid-v4' \
--header 'X-Item-ID: ID Azienda' \
--header 'X-User-ID: ID Chiave Tecnica' \
--header 'Authorization: Bearer '
```

Notifica di Scarto

GET/v2/invoices/{hubId}/download

Parametri:

- **messageId**: il valore da inserire in questo parametro è recuperabile dalla response della chiamata di *Dettaglio Fattura*.
  Il tag di riferimento è "*notificationId*" e si trova nella sezione "*messages*";
- **format**: scegliere il tipo di formato dello notifica (Esempio "XML").

Esempio Chiamata:

```
curl --location --request GET 'https://b2bread-api-
test.agyo.io/api/v2/invoices/{hubId}/download?messageId=Ivwy1pPrTtV1xA&format=XML' \
--header 'Authorization: Bearer ' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
--header 'User-Agent: Postman' \
--header 'X-Request-ID: cuuid' \
--header 'X-Correlation-ID: uuid-v4' \
--header 'X-Item-ID: ID Azienda' \
--header 'X-User-ID: ID Chiave Tecnica' \
--header 'Accept-Language: it-IT' \
--header 'Content-Type: application/json' \
--header 'Cookie:
visid_incap_2770748=A9AiR++qR4+3fodrMwxP8h4nWGIAAAAAQUIPAAAAAABJFK5guyJfBL1cz/JUufW5;
visid_incap_2773288=C84tkoe0SrSMtGsu/hERxgKbb2IAAAAAQUIPAAAAAAABCra5YhrEW739veNMLkYa'
```

Download Metadata

> GET/v2/invoices/{hubId}/download/metadata

Esempio Chiamata:

```
curl --location --request GET 'https://b2bread-api-
test.agyo.io/api/v2/invoices/{hubId}/download/metadata' \
--header 'Authorization: Bearer ' \
--header 'accept: application/hal+json' \
--header 'User-Agent: Postman' \
--header 'X-App-Name: VRxyz' \
--header 'X-App-Version: 1.0' \
-header 'X-Request-ID: cuuid' \
--header 'X-Correlation-ID: uuid-v4' \
--header 'X-Item-ID: ID Azienda' \
--header 'X-User-ID: ID Chiave Tecnica' \
```

ItemId (UUID)

Le API v2 di lettura e scrittura possono essere utilizzate con l'itemId da 36 (o 40 nel caso sia un uffficio) caratteri. Per garantire la retrocompatibilità e non pregiudicare il funzionamento delle API v1 l'itemId verrà convertito in maniera trasparente da TSDigital nel vecchio identificativo.

Chiariamo con un esempio.

L'azienda XYZ ha come vecchio identificativo AAABBB86C13D205E, nuovo identificativo c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1

Payload di upload fattura

```
{
  "transmitterId":"c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1",
  "senderId":"c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1",
  [...]
}
```

Se su API v1 otterrò un errore, su API v2 invece andrà a buon fine (ovviamente a patto di avere la necessaria autorizzazione ad operare sull'azienda XYZ).

La fattura verrà assegnata da TSDigital utilizzando il vecchio identificativo AAABBB86C13D205E

Effettuando una lettura alle API v1 con parametro senderId=c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1 non otterrò nessun risultato.

Effettuando invece una lettura alle API v2 con parametro ownerId=c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1 il payload di risposta sarà:

```
{
    "_embedded": {
        "invoiceList": [
            {
                "transmitterId": "AAABBB86C13D205E",
                "senderId": "AAABBB86C13D205E",
     [...]
}
```

Identico comportamento si otterrà interrogando le API v2 con parametro ownerId=AAABBB86C13D205E

Il comportamento sarà questo fino a quando tutti i client non saranno tutti aggiornati per utilizzare l'itemId, a quel punto verranno spente le API SOAP e le API v1. A quel punto procederemo ad effettuare un aggiornamento del database pertanto interrogando le api v2 con parametro ownerId=AAABBB86C13D205E non otterrò nessun risultato, mentre utilizzando il nuovo itemId ownerId=c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1 il payload sarà:

```
{
    "_embedded": {
        "invoiceList": [
            {
                "transmitterId": "c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1",
                "senderId": "c77f35bf-fbca-4111-8d92-8a9e3ef9f1f1",
     [...]
}
```

> Per sender, transmitter ed in generale qualsiasi campo trattato ad oggi come un codice fiscale vale quanto detto per l'hubId. Non effettuare nessuna logica ma trattare il dato come una stringa di lunghezza variabile

Migrazione da API REST v1 a v2

Lettura Fatture

Differenze con le attuali V1

Le API v2 di lettura differiscono dalle v1 relativamente alla paginazione. Nella nuova versione infatti non è più presente il conteggio degli elementi totali e del numero di pagine ma un booleano che indica se è presente una pagina successiva.

V1 (deprecate)

```
  "page": {
        "size": 20,
        "totalElements": 111,
        "totalPages": 6,
        "number": 0
    }
```

V2

```
"page": {
        "size": 20,
        "hasNext": true,
        "continuationToken": "abc...xyz"
    }
```

Per ottenere la pagina successiva è sufficiente richiamare la url fornita nella risposta nell'oggetto _links:

```
 "_links": {
        "first": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/invoices?ownerId=ABCDEFGHI&size=20&sort=lastTimestamp,desc"
```

```
        },
        "self": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/invoices?ownerId=ABCDEFGHI&size=20&sort=lastTimestamp,desc"
        },
        "next": {
            "href": "https://b2bread-api-
test.agyo.io/api/v2/invoices?ownerId=ABCDEFGHI&continuationToken=5WvMRcmA_8WJtFSiemEregGggF3et
SD4QqPv1aSDXnwfjLKGZKOwq13sye5fl9NQ8799jMkJveAlTaOmC9H4Pn3Wtt2cA5iTyce_5YgEik8cNNEvvUwfz43t22M
3tjjc&size=20&sort=lastTimestamp,desc"
        }
    }
```


Come è possibile osservare esaminando questi link, nella chiamata viene aggiunto il parametro relativo al continuationToken. Questo parametro consentirà al servizio di lettura di riprendere la lettura nel punto in cui si è arrivati nella lettura precedente

Un’ulteriore differenza consiste nell’obbligatorietà del campo OwnerId

Questo campo identifica il proprietario della fattura. Confrontandolo con le V1 rappresenta il sender per la ricerca delle fatture attive e il recipient per la ricerca delle passive.

Questo campo è quindi un riferimento all’azienda con la quale si sta operando. Ed è anch’esso un AgyoItemId.