MEXAL

PASSCOM

MANUALE WEBAPI

28/10/2021 v1.1

PASSEPARTOUT®
Software gestionale

WEBAPI v1.1

Sommario

1. Introduzione 2

1.1. Cosa è 2

1.2. Glossario 2

1.3. Come si usa 2

2. Formato della http request 3

2.1. Installazioni locali 3

2.2. Installazioni LIVE 6

3. comandi e scambio dei dati 7

3.1. Tipo di comandi e spazio dei nomi 7

3.2. Metodo GET 8

3.3. Metodo POST 9

3.4. Metodo PUT 10

3.5. Metodo DELETE 10

3.6. Entità ricerca 10

3.7. Entità allegati 11

3.8. Webapi per la richiesta di servizi 12

3.9. Riassunto dei comandi RESTful 13

4. Accesso ai dati del gestionale 14

Esempio gestione ordini clienti 14

5. CHANGELOG 17

V1.1 : dalla Versione gestioneale 2021I – V816 17

1. INTRODUZIONE

1.1. COSA È

WebApi è, come suggerisce il nome, una collezione di API Passepartout basate su tecnologia Web tramite protocollo https. Il servizio è fruibile utilizzando richieste https secondo lo stile architetturale RESTful: **R**epresentational **S**tate **T**ransfer. Sono implementati tutti i metodi CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) per mezzo dei metodi http **POST**, **GET**, **PUT**, **DELETE**.

Secondo il paradigma REST il servizio è fortemente orientato alle entità (gli archivi del gestionale) le quali saranno raggiungibili mediante path specifici.

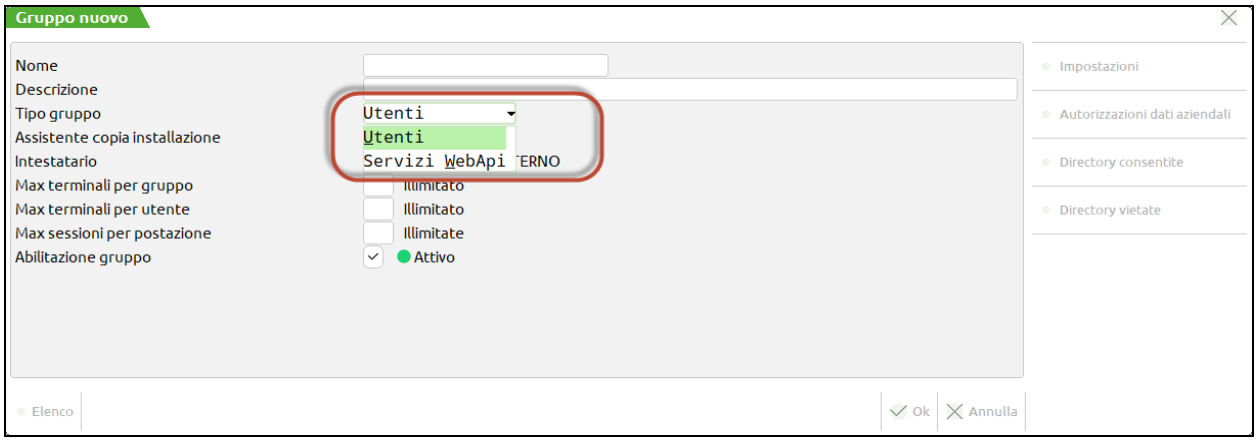
1.2. GLOSSARIO

Di seguito la terminologia ed il significato dei termini utilizzati in questo manuale.

- *risorsa*: oggetto presente sul gestionale Mexal/Passcom su cui possono essere effettuate operazioni
- *entità*: risorsa. Entità e risorsa verranno trattati come sinonimi all'interno di questo manuale
- *http*: Hypertext Transfer Protocol
- *https*: Hypertext Transfer Protocol over Secure Socket Layer. Nel corso di questo manuale potrà capitare di riferirsi indistintamente ad http o https per definire alcuni concetti. WebAPI gestirà solo richieste tramite protocollo sicuro https.
- *http request*: azione (POST, GET, PUT, DELETE) fatta su una risorsa identificata da un dato url
- *header*: parte della *http request* in cui vengono impostati metadati necessari al completamento della richiesta stessa
- *body*: corpo di una *http request* in cui sarà presente il dato da scambiare
- *query string*: è la parte di un URL che contiene dei dati da passare in input ad un servizio
- *JSON*: **J**ava**S**cript **O**bject **N**otation, è un formato standard adatto all'interscambio di dati tra applicazioni client/server
- *installazione*: ci si riferisce al programma gestionale Mexal/Passcom
- *ambiente live*: installazione presente in server farm Passepartout
- *ambiente locale*: installazione presente su un server locale alla rete del cliente
- *front-end:* parte di un programma responsabile dell'acquisizione dei dati di ingresso
- *back-end:* parte di un programma che elabora i dati acquisiti
- *cookie:* è una stringa di testo di piccole dimensioni inviata da un web server ad un web client e poi rimandata indietro dal client al server; tipicamente è utilizzata per implementare meccanismi di identificazione di un client presso un server

1.3. COME SI USA

WebApi è disponibile sia in ambiente locale che live dopo aver attivato la suite MDS tramite YouPass. L'attivazione del servizio abiliterà in Mexal/Passcom un gruppo particolare di utenti "Servizi WebApi". Solo gli utenti appartenenti a gruppi di questo tipo potranno utilizzare solo ed esclusivamente i servizi WebApi.



WebApi locale: è possibile accedere su installazione locale specificando l'indirizzo del server più la porta su cui è in ascolto il servizio. Il servizio deve essere abilitato lato server Mexal/Passcom tramite il menu *Servizi->Configurazioni->Configurazione moduli->WebApi*. In questa videata sarà possibile verificare la corretta abilitazione del modulo e la porta assegnata al servizio. L'accesso sarà possibile specificando nell'header della richiesta https le credenziali di accesso più altri parametri per identificare l'azienda e la data di esercizio.

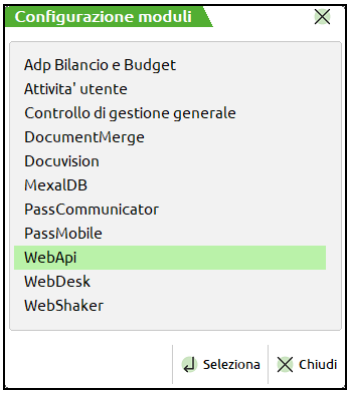
WebApi Live: in ambiente live, se attivato da codice contratto, il servizio è già disponibile e raggiungibile tramite l'indirizzo https://services.passepartout.cloud/. In questo caso l'autenticazione avviene sempre tramite credenziali di accesso nell'header della richiesta https (come per le installazioni locali) ma usando come *base address* **https://services.passepartout.cloud/webapi/risorse** (vedi sezione dedicata)

Il gruppo Servizi WebAPI gestisce un numero massimo di 3 utenti: ogni utente del gruppo Servizi WebApi avrà a disposizione un pool di massimo 5 servizi contemporanei con un time-out di 5 minuti per ogni servizio. Questo vuol dire che lo stesso utente WebAPI potrà accedere contemporaneamente solo da 5 client differenti, le successive richieste verranno rifiutate fino a che uno dei servizi del pool non si libera.

2. FORMATO DELLA HTTP REQUEST

2.1. INSTALLAZIONI LOCALI

Per utilizzare WebApi in installazioni locali è necessario attivare il modulo dal menu Servizi - Configurazioni - Configurazione moduli – WebApi.



Le informazioni necessarie per l'accesso sono:

URL = <indirizzo_server>
PORTA = <porta_server_https>
USER = <utente_di_mexal_passcom>
PWD = <password_utente_mexal_passcom>

Se la richiesta prevede l'utilizzo di un comando che coinvolge una determinata azienda, è necessario fornire anche le seguenti informazioni:

AZIE = <azienda>
SOTTOAZ = <sottoazienda> (opzionale)
ANNO_AZ = <anno_di_apertura>
MAGAZZINO=<numero_magazzino_default> (opzionale)

NOTA: se non viene utilizzato il numero di magazzino, tutte le operazioni che coinvolgono il magazzino utilizzeranno come magazzino di default il numero 1.

Se sull'installazione locale è attivo anche il controllo delle credenziali di sistema operativo è necessario utilizzare anche queste credenziali

USER_SYS = <utente_di_sistema>
PWD_SYS = <password_utente_di_sistema>

L'indirizzo per raggiungere WebApi in locale dovrà essere formato in questo modo:

https://URL:PORTA/webapi/risorse/

Esempi:

PUT https://miodominio:9004/webapi/risorse/clienti/codice_cliente (aggiorna il cliente: dati nel corpo della richiesta)

POST https://miodominio:9004/webapi/risorse/articoli (crea un nuovo articolo: dati nel corpo della richiesta)

GET https://miodominio:9004/webapi/risorse/articoli (legge tutti gli articoli)

DELETE https://miodominio:9004/webapi/risorse/articoli/codice_articolo (cancella un articolo)

Le credenziali di accesso USER e PWD devono essere formattate in questo modo:

- concatenare USER e PWD formando la seguente stringa: "USER:PWD" es. "ADMIN:MEXAL"
- fare un encoding in base64 della stringa ottenuta al passo precedente

Nell'header della richiesta https dovranno poi essere inserite le informazioni ottenute dai passi precedenti in questo modo:

```
'Authorization': 'Passepartout <credenziali_b64>'
'Content-type': 'application/json'
'Coordinate-Gestionale': 'Azienda=<AZIE> SottoAzienda=<SOTTOAZ> Anno=<ANNO_AZ> Magazzino=<MAGAZZINO>'
```

Nel caso l'installazione gestisca anche l'ulteriore autenticazione tramite credenziali di sistema operativo (login=1) devono essere aggiunte nella chiave 'Authorization' codificate in base64 analogamente a quanto fatto per le credenziali dell'utente Mexal/Passcom.

L'header in questo caso avrà una struttura di questo tipo:

```
'Authorization': 'Passepartout <credenziali_b64> <credenziali_sistema_b64>'
'Content-type': 'application/json'
'Coordinate-Gestionale': 'Azienda=<AZIE> SottoAzienda=<SOTTOAZ> Anno=<ANNO_AZ> Magazzino=<MAGAZZINO>'
```

Il Content-type (opzionale) prevede come charset di default utf-8 e non è necessario specificarlo. Se però si vuole specificare anche il charset è necessario che sia utf-8: 'Content-type': 'application/json;charset=utf-8'. Qualsiasi altro tipo di charset verrà rifiutato con un errore.

Di seguito due esempi di come deve essere formattato un header di una richiesta https su installazioni locali.

Caso di autenticazione con solo le credenziali dell'utente dell'installazione:

```
{
        'Authorization': 'Passepartout QURNSU46TUVYQUw=',
        'Content-type': 'application/json',
        'Coordinate-Gestionale': 'Azienda=DEM Anno=2021 Magazzino=2'
}
```

Caso di autenticazione in cui sono necessarie anche le credenziali di sistema operativo (login=1):

```
{
        'Authorization': 'Passepartout QURNSU46TUVYQUw= QURNSU46QURNSU4=',
        'Content-type': 'application/json',
        'Coordinate-Gestionale': 'Azienda=DEM Anno=2021'
}
```

Nel caso di installazioni locali, l'accesso alle risorse tramite protocollo sicuro https avviene utilizzando un certificato autoprodotto (self-signed). Le librerie utilizzate dall'applicazione client dovranno quindi poter gestire questo caso.

In alternativa, se non fosse possibile utilizzare i certificati self-signed, è necessario dotare l'installazione di certificati validi (non self signed). Questo è possibile farlo inserendo i certificati tramite il menu Servizi - Configurazioni - Configurazione moduli – WebApi nella sezione Configurazione Server (in modo del tutto analogo a quanto avviene per WebDesk).

2.2. INSTALLAZIONI LIVE

Per poter accedere in ambiente Live, i parametri necessari per l'accesso sono i seguenti:

https://services.passepartout.cloud/webapi/risorse

DOMINIO = <dominio_della_installazione_live>
USER = <utente_di_mexal_passcom>
PWD=<password_utente_mexal_passcom>

Se la richiesta prevede l'utilizzo di un comando che coinvolge una determinata azienda, è necessario fornire anche le seguenti informazioni:

***AZIE** = <azienda>*
***SOTTOAZ** = <sottoazienda> (opzionale)*
***ANNO_AZ** = <anno_di_apertura>*
***MAGAZZINO**=<numero_magazzino_default> (opzionale)*

Come per le installazioni locali, le credenziali di accesso USER e PWD devono essere formattate in questo modo:

- concatenare USER e PWD formando la seguente stringa: "USER:PWD" (es. "ADMIN:ADMIN")
- fare un encoding in base64 delle credenziali

Nell'header della richiesta https dovranno essere inserite le informazioni ottenute dai passi precedenti, dove rispetto alle installazioni locali nell'elemento Authorization va specificato anche il dominio nella forma *Dominio=<MIO_DOMINIO>*

'Authorization': 'Passepartout <credenziali_b64> Dominio=<DOMINIO>'
'Content-type': 'application/json'
'Coordinate-Gestionale': 'Azienda=<AZIE> SottoAzienda=<SOTTOAZ> Anno=<DATA_AZ>',

Di seguito un esempio di come deve essere formattato un header di una richiesta https in ambiente live.

```
{
        'Authorization': 'Passepartout QURNSA46QQ== Dominio=MIODOMINIO',
        'Content-type': 'application/json',
        'Coordinate-Gestionale': 'Azienda=DEM Anno=2021'
}
```

3. COMANDI E SCAMBIO DEI DATI

Tutti gli esempi e le descrizioni dei comandi da questo punto in poi avranno lo stesso formato, sia per installazioni locali che live, lasciando sottointeso il *base address*. Per semplicità quindi tratteremo il path in questo modo:

https://base_address/webapi/risorse/<nome_risorsa>

dove *base_address* potrà essere:

- <indirizzo_server>:<porta_https> nel caso di installazioni locali
- services.passepartout.cloud nel caso di installazioni live

Per tutti i casi di autenticazione citati, l'interazione con Mexal/Passcom avviene mediante i seguenti metodi http:

POST: utilizzato esclusivamente per creare una nuova entità (es. creazione di un nuovo cliente)

GET: utilizzato per leggere una entità o una collezione di entità (es. la lista di tutti i clienti)

PUT: utilizzato esclusivamente per aggiornare una entità (es. modifica di una descrizione di un articolo)

DELETE: utilizzato per cancellare una entità (es. eliminazione di un cliente)

I metodi POST e PUT prevedono anche un *payload* contenuto nel *body* della richiesta https in formato JSON in cui saranno inseriti i dati da inserire o aggiornare, mentre per i comandi di tipo GET e DELETE un eventuale contenuto viene ignorato.

3.1. TIPO DI COMANDI E SPAZIO DEI NOMI

Mexal/Passcom prevede due tipi di comandi:

- Comandi che richiedono necessariamente l'apertura di una azienda in un dato anno
- Comandi sovra-aziendali che possono essere eseguiti anche senza l'accesso ad una particolare azienda

Nel primo caso, nell'header di tutte le richieste https devono essere specificati anche i parametri di accesso alla particolare azienda (o sottoazienda), come indicato nel capitolo 2.

Nel secondo caso tali parametri possono essere omessi in quanto non verranno controllati dal servizio WebAPI.

Tutti i nomi sono in alfabeto minuscolo senza indicazione sul formato del dato ed eventuali nomi composti saranno separati dal carattere *underscore* (_).

Esempi:

_ARDES$ = ardes

_PCLIS = pclis

I campii di tipo *array* possono arrivare ad un massimo di 3 livelli. Ogni elemento di un campo di tipo array è identificato a sua volta da un elemento di tipo array in cui sono specificati gli indici dei vari livelli ed il valore in ultima posizione.

Ad esempio per un campo array di 3 livelli, per identificare un elemento, la struttura sarà di questo tipo:

```
"campo_array": [
        [1,2,1,"a"],        // valore "a" in posizione 1 nel primo livello, 2 nel secondo e 1 nel terzo
        [2,1,4,"b"]         // valore "b" in posizione 2 nel primo livello, 1 nel secondo e 4 nel terzo
]
```

Per i nomi delle variabili, il tipo di dato ed il significato usare il parametro il metodo GET con il parametro *info=true* nella query string.

3.2. METODO GET

Secondo l'architettura REST i comandi di tipo GET non devono specificare alcun contenuto nel corpo della richiesta http. In ogni caso anche se presente deve essere ignorato.

Per capire quali sono le entità accessibili è disponibile una entità *HELP* utilizzabile con metodo *GET* che restituisce un JSON con tutte le entità ed i metodi disponibili.

GET https://base_address/webapi/risorse/help

Verrà tornato un JSON con l'elenco di tutti gli end-point gestiti. Di default verranno restituiti il path, la descrizione ed il metodo disponibile. Se si vogliono ulteriori informazioni è possibile usare il parametro *extended* in query string

GET https://base_address/webapi/risorse/help?extended=true

```
{
    "next": true/false,                          // specifica se la risorsa gestisce il parametro next
    "filter": true/false,                        // specifica se la risorsa gestisce i filtri
    "regexp": "<path_risorsa>",                  // path e la regular expression associata alla risorsa
    "descrizione": "<descrizione_risorsa>",      // descrizione della risorsa
    "coordinate": true/false,                    // specifica se per l'accesso alla risorsa servono le coordinate aziendali
    "versione": "<versione_gestionale>",         // specifica da che versione contenitore (Mexal/Passcom) è disponibile la risorsa
    "method": "<metodo_http>",                   // specifica il metodo http disponibile per la
    "max": true/false,                           // specifica se la risorsa gestisce il parametro max
    "chiavi": [ <elenco_chiavi> ],               // specifica eventuali chiavi per referenziare gli elementi di una risorsa
    "location": true/false,                      // specifica se la risorsa ritorna un tag location nell'header
    "fields": true/false                         // specifica se la risorsa gestisce il parametro fields
}
```

Facendo GET su un'entità senza specificare altri parametri, il JSON di risposta conterrà tutti i record con tutti i campi dell'archivio.

GET https://base_address/webapi/risorse/articoli

Se invece si vuole recuperare il singolo record bisogna specificare l'ID/Codice della particolare entità

GET https://base_address/webapi/risorse/articoli/11AA55

Nota: la differenza sostanziale tra i comandi GET che tornano una collezione di elementi rispetto a quelli che tornano un singolo elemento sta nel fatto che nel primo caso il risultato potrebbe avere alcuni campi in meno. Inoltre gli array sono tutti al massimo di 2 dimensioni, mentre nel secondo caso gli array possono avere fino a un massimo di 3 dimensioni. Per i dettagli sui campi disponibili usare info=true in query string.

È possibile operare dei filtri usando le *query string.*

Per specificare i soli campi che si vuole vengano restituiti bisogna utilizzare l'attributo *fields*

GET https://base_address/webapi/risorse/articoli/11AA55?fields=arcod,ardes

Nota: è sempre consigliato estrarre i soli campi necessari in modo da velocizzare l'elaborazione lato server, soprattutto nel caso che la risorsa da gestire sia molto corposa.

È possibile specificare il numero massimo di record restituiti nel caso si richieda una collezione di risorse. In *query string* sarà necessario utilizzare l'attributo *max* (massimo 100)

GET https://base_address/webapi/risorse/articoli?fields=arcod,ardes&max=10

La risposta, oltre ai record letti, in questo caso contiene anche un elemento che identifica quale sarà il successivo record da cui partire per eventuali successive letture.

```
{
  [
    {
      "arcod": "123ABC",
      "ardes": "Pacco pignoni"
    },
    {
      "arcod": "123456",
      "ardes": "Cambio SRAM"
    },
    {
      "arcod": "ABCDEF",
      "ardes": "Gomma MAXISS ARDENT"
    }
  ],
  "next": "0F45AD69"
}
```

Di default il metodo GET legge al massimo 100 record (come se fosse impostato max=100). Per archivi più lunghi di 100 record verrà quindi sempre restituito un elemento *next* indicante il prossimo record da cui partire per completare la lettura.

È possibile specificare il numero di record da cui partire se è stato restituito un *next* in un precedente metodo GET

GET https://base_address/webapi/risorse/articoli?fields=arcod,ardes&next=0F45AD69

Per ogni risorsa è possibile ottenere un elenco di tutti i campi disponibili ed il loro significato. È sufficiente specificare nel path della query string l'attributo *info=true*, in questo caso tutte le altre query string (se presenti) verranno ignorate.

GET https://base_address/webapi/risorse/articoli?info=true

Nel caso di GET su una collezione di entità è possibile impostare anche filtri più evoluti oltre all'utilizzo di quelli specificati in *query string* (*fields*, *max* e *next).*

I filtri dovranno essere impostati nel corpo della richiesta in formato JSON. Dato però che, come precedentemente indicato, il metodo GET non prevede l'utilizzo di alcun contenuto nel body della http request, i filtri saranno utilizzati sfruttando il metodo POST.

Ogni entità ha a disposizione una risorsa denominata *ricerca* su cui sarà possibile specificare opportuni filtri nel body della request.

POST https://base_address/webapi/risorse/articoli/ricerca (vedi entità RICERCA)

3.3. METODO POST

Il metodo POST deve essere utilizzato esclusivamente per la creazione di nuove entità. Nel path non deve essere mai specificato il codice della risorsa da creare. Nel body della richiesta https deve essere presente un JSON con i dati da inserire. Nel caso debba essere specificato un codice per la creazione della risorsa (vedi anagrafica articoli) bisogna aggiungerlo nel body della richiesta https.

Se la creazione avviene con successo verrà tornato un *http status code 201 (Created).*

POST https://base_address/webapi/risorse/clienti

```
{
  "pccod": "501.AUTO", //codice creato in automatico sul mastro indicato
  "pcdes": "Cliente di prova",
  "pcnaz": "I",
  // altri campi obbligatori
}
```

Creazione con codice specificato dall'utente

POST https://base_address/webapi/risorse/articoli

```
{
```

```
    "arcod": "ARTICOLO1",
    "ardes": "Articolo di prova"
    //altri campi obbligatori
}
```

Il *response* di una *http request* conterrà nell'header della risposta un tag *location* comprendente il path della nuova entità appena creata.

Creazione di un articolo

POST https://base_address/webapi/risorse/articoli

Body della richiesta https:

```
{
    "arcod": "ARTICOLO1",
    "ardes": "Articolo di prova"
    //altri campi obbligatori
}
```

Header della risposta https:

```
{
        // altri tag
        'Content-type': 'application/json; charset=utf-8',
        'Location': 'https://base_address/webapi/risorse/articoli/ARTICOLO1',
}
```

3.4. METODO PUT

Il metodo PUT deve essere utilizzato per modificare/aggiornare delle risorse già esistenti. In questo caso è obbligatorio specificare sempre il codice della risorsa nel PATH, mentre i dati devono essere specificati nel body della *http request*. Non è previsto alcun contenuto nel *body* della risposta https. In caso di successo verrà solamente restituito un *http status code 204 (No Content).*

PUT https://base_address/webapi/risorse/articoli/11AA55

Body della richiesta https:

```
{
    "ardes": "Descrizione modificata"
    //altri campi
}
```

3.5. METODO DELETE

Il metodo DELETE deve essere utilizzato per eliminare una risorsa esistente. In questo caso è obbligatorio specificare sempre il codice della risorsa nel PATH. Non è previsto alcun contenuto nel *body* della risposta https. In caso di successo verrà solamente restituito un *http status code 204 (No Content).*

DELETE https://base_address/webapi/risorse/articoli/11AA55

3.6. ENTITÀ RICERCA

La risorsa *ricerca* è una particolare risorsa comune a tutti i gruppi di risorse (es. clienti, articoli, documenti, etc.).

Questa particolare risorsa può essere utilizzata solo mediante metodo POST. Il significato è: "crea una ricerca su questa collezione di risorse". In questo modo è possibile ottenere un risultato come se fosse una GET in cui sono specificati filtri particolari nel *body* della *request* oltre quelli specificati in query string. Come per il comando GET è possibile utilizzare i filtri in query string (*fields, max e next*).

I filtri devono essere specificati secondo questo pattern

```
{
```

```
    "filtri":[
      {
          "campo":"<nome_campo>",
          "indice":"<indice_campo_array>", // facoltativo: serve nel caso il campo sia un array. Nel caso di liste, gli array sono di
                                              un solo livello quindi deve essere specificato solo un indice
          "condizione": "=,<,<=,>,>=,<>,contiene",
          "case_insensitive": true/false      // default: false
          "valore": "<valore da testare>"
      },
     // altri elementi in AND
    ]
}
```

Nel caso la condizione sia "*contiene*" può essere specificato anche se la ricerca debba essere case sensitive o meno specificando il parametro "*case_insensitive*" (default: false)

POST https://base_address/webapi/risorse/articoli/ricerca?fields=arcod,ardes

Esempio body della richiesta https

```
{
    "filtri":[
      {
          "campo":"artip",
          "condizione": "=",
          "valore": "A"
      },
      {
          "campo":"arcuc",
          "condizione": ">",
          "valore": 100
      }
    ]
}
```

I filtri sono tutti in logica AND tra loro. Nell'esempio verranno restituiti solo gli articoli di tipo merce e costo ultimo anno corrente maggiore di 100 e solo i campi codice e descrizione.

3.7. ENTITÀ ALLEGATI

Solo per gli archivi che hanno allegati (es. immagini articoli) è disponibile una entità dedicata per l'accesso ai documenti.

GET https://base_address/webapi/risorse/<categoria_risorsa>/<codice_risorsa>/allegati/<tipo_allegato>

Dove tipo allegato è il tipo di file che si vuole referenziare. Nell'header della risposta sarà specificato il content-type corretto per il file restituito (ad esempio image/jpeg per le immagini).

Esempio di letture delle tre immagini articolo (icona, catalogo, immagine):

Lettura dell'icona di un articolo

GET https://base_address/webapi/risorse/articoli/FUELEX97/allegati/icona

Lettura dell'immagine di catalogo

GET https://base_address/webapi/risorse/articoli/FUELEX97/allegati/immagine-catalogo

Lettura dell'immagine articolo

GET https://base_address/webapi/risorse/articoli/FUELEX97/allegati/immagine

3.8. WEBAPI PER LA RICHIESTA DI SERVIZI

In tutti i casi in cui l'operazione che si vuole effettuare sul gestionale non identifichi una risorsa ben precisa, ma piuttosto richieda l'esecuzione di una operazione o un insieme di operazioni che dovranno essere svolte da Mexal/Passcom, l'approccio RESTful non è più sufficiente e bisogna adottare un paradigma differente.

WebAPI mette a disposizione un ulteriore canale per la gestione di questi casi identificato con il nome *servizi.*

Le *http request* in questo caso saranno tutte di tipo *POST* e richiederanno l'esecuzione di un servizio.

Il path deve essere quindi formato in questo modo

POST https://base_address/webapi/servizi

Tutti i dati relativi all'esecuzione del particolare comando dovranno essere specificati nel *body* della *request*.

COLLAGE SERVER REMOTO

Questo comando permette l'esecuzione di un Collage Server Remoto contenuto in una Passapp esattamente come avviene con WebShaker.

In questo caso è necessario specificare i dati necessari all'individuazione dell'etichetta Collage da eseguire all'interno della Passapp indicando *codice_app*, *nome_collage* e *etichetta_collage*

Esempio di esecuzione di un Collage Server Remoto

Richiesta

POST https://base_address/webapi/servizi

```
{
   "cmd": "esec_collage_server_remoto",
   "codice_app": "924148WEBSHAKER",
   "nome_collage": "colws",
   "etichetta_collage": "WAPI"
}
```

Risposta

```
{
            <risultato dell'elaborazione>
}
```

SVILUPPO DISTINTA BASE

Servizio per lo sviluppo della distinta base di un articolo

Richiesta

POST https://base_address/webapi/servizi

```
{
   "cmd": "sviluppo_distinta_base",
   "dati": {
      "codice_articolo": "<codice_articolo>",
      "codice_articolo_padre": "<codice_art_padre>",
      "codice_cliente": "<codice_cliente>",
      "id_riga_oc": 0,
      "quantita_taglie": [
            [1,n],
           ..
           [32,n]
         ]
   }
}
```

3.9. RIASSUNTO DEI COMANDI RESTFUL

	Metodo http	PATH	Query String	HEADER	BODY	Success http Code

Creazione	POST	https://base_address/webapi/<nome_risorsa>	NO	Authorization
				Content-Type
				Coordinate-Gestionali
					{
					<dati>
					}
						201 (Created)

Lettura	GET	https://base_address/webapi/<nome_risorsa> https://base_address/webapi/<nome_risorsa>/<codice>	fields max next	Authorization Content-Type Coordinate-Gestionali	{}	200 (OK)

Aggiornamento	PUT	https://base_address/webapi/<nome_risorsa>/<codice>	NO	Authorization
				Content-Type
				Coordinate-Gestionali
					{
					<dati>
					}
						204 (No Content)

Eliminazione	DELETE	https://base_address/webapi/<nome_risorsa>/<codice>	NO	Authorization Content-Type Coordinate-Gestionali	{}	204 (No Content)
Ricerca	POST	https://base_address/webapi/<nome_risorsa>	fields max next	Authorization Content-Type Coordinate-Gestionali	filtri:[{…}, {…}]	200 (OK)

4. ACCESSO AI DATI DEL GESTIONALE

Per gli archivi ad un solo livello (articoli, clienti, fornitori etc.) per recuperare le informazioni di tutto l'archivio è sufficiente utilizzare qualsiasi comando REST con un'unica richiesta.

Invece per quanto riguarda archivi che presentano più livelli (tipicamente i documenti) può essere necessario effettuare più chiamate per avere i dati nella loro completezza.

Prendiamo come esempio i documenti: questi archivi sono composti da una tabella master e una tabella di dettaglio (testata + righe). Per avere quindi il dettaglio completo delle testate e di tutte le righe che compongono il documento sarà necessario utilizzare più chiamate REST.

Per i documenti c'è un contenitore dedicato in cui poi andranno specificate le varie entità a cui si vuole accedere.

Il path sarà di questo tipo

COMANDO https://base_address/webapi/risorse/documenti/<risorsa>

Prendiamo come esempio i documenti di tipo *ordini clienti*.

Per referenziare i dati di testata il path sarà di questo tipo

GET https://base_address/webapi/risorse/documenti/ordini-clienti

Con questo comando viene restituita una lista di tutte le testate dei documenti.

Se invece si vogliono avere i dettagli sulle righe dei documenti bisogna utilizzare un comando di questo tipo

GET https://base_address/webapi/risorse/documenti/ordini-clienti/righe

In questo caso verranno restituite tutte le righe di tutti i documenti di tipo *ordini-clienti*. Sarà poi compito del programmatore utilizzare i filtri in modo da leggere/scrivere solo le righe di determinati documenti.

Nel caso si debba specificare una chiave, bisognare comporla inserendo la sigla, la serie e il numero del documento in questo modo

GET https://base_address/webapi/risorse/documenti/ordini-clienti/OC+1+55

Nota: i campi chiave necessari possono essere recuperati visionando la risposta dell'HELP. Per ogni entità sarà presente un elemento "chiavi" contenente tutte le chiavi necessarie per completare la richiesta.

Nota: i dati restituiti da una GET su una lista possono essere leggermente differenti da quelli restituiti dalla GET di una singola entità. Ad esempio nelle liste gli array sono tutti di un solo livello, mentre nelle singole entità gli array possono essere restituiti fino a 3 livelli.

4.1. ESEMPIO GESTIONE ORDINI CLIENTI

Creazione di un ordine

Richiesta

POST https://base_address/webapi/risorse/documenti/ordini-clienti

```
{
    "mmncreriga":[
      [
          1,
          1
      ]
    ],
    "mmtpr":[
      [
          1,
```

```
            "R"
        ]
    ],
    "mmart":[
        [
            1,
            "ATT002"
        ]
    ],
    "mmali":[
        [
            1,
            "22"
        ]
    ],
    "mmqta":[
            [
                1,
                3
            ]
    ],
    "mmage":"601.00001",
    "mmser":1,
    "mmsig":"OC",
    "mmnum":0,
    "mmdat":"20210825",
    "mmcli":"501.00022"
}
```


Modifica di un ordine

Richiesta

PUT https://base_address/webapi/risorse/documenti/ordini-clienti/OC+1+2


```
{
    "mmncreriga":[
        [
            1,
            1
        ]
    ],
    "mmtpr":[
        [
            1,
            "R"
        ]
    ],
    "mmage":"601.00002"
}
```


NOTA: i dati che referenziano le righe del documento devono essere sempre tutti specificati. Le righe non dichiarate verranno eliminate. In questo esempio è stato indicato che nel documento esiste solo una riga e di tipo "R". Se il documento avesse avuto altre righe sarebbero state eliminate. ATTENZIONE!

In questo caso si vuole modificare solo il codice agente lasciando tutto il resto inalteraro. Gli indici di riga e il tipo di riga devono essere sempre specificati.

Lettura di tutti gli ordini

Richiesta

GET https://base_address/webapi/risorse/documenti/ordini-clienti

Lettura di un ordine

Richiesta

GET https://base_address/webapi/risorse/documenti/ordini-clienti/OC+1+2

Cancellazione di un ordine

Richiesta

DELETE https://base_address/webapi/risorse/documenti/ordini-clienti/OC+1+2

Lettura di tutte le righe di un particolare ordine

Richiesta

POST https://base_address/webapi/risorse/documenti/ordini-clienti/righe/ricerca

```
{
   "filtri":[
      {
         "campo":"mmsig",
         "condizione":"=",
         "valore":"OC"
      },
      {
         "campo":"mmser",
         "condizione":"=",
         "valore":1
      },
      {
         "campo":"mmnum",
         "condizione":"=",
         "valore":3
      }
   ]
}
```

Ritorna tutte le righe documento dell'OC1/3

Nota: le righe del documento, essendo una lista, possono essere modificate solo quando si esegue la PUT sul documento che le contiene specificando riga per riga i campi da modificare.

5. CHANGELOG

Di seguito saranno indicati tutti i cambiamenti di questo manuale rispetto alle versioni precedenti.

V1.1 : DALLA VERSIONE GESTIONALE 2021I – V816

Modifiche al protocollo:

- Aggiunto il parametro opzionale Magazzino nell'header della richiesta per permettere di specificare il magazzino di default
- Le liste sono state arricchite di ulteriori dati e gli array possono arrivare fino a 2 livelli invece che 1 solo livello
- Cambiato path del comando HELP e aggiunto filtro *extended* in query string
- Aggiunte nuove risorse: usare il comando HELP per tutti i dettagli
- Cambiato il path di alcune risorse: usare il comando HELP per tutti i dettagli
- Aggiunto limite al numero massimo di utenti appartenenti al gruppo WebAPI: 3

- Aggiunto servizio per lo sviluppo della distinta base
- Eliminata documentazione sull'accesso alle risorse tramite il canale *servizi*: si consiglia di utilizzare sempre l'accesso con logica REST sul path "risorse"
- Inserita entità allegati: in questa versione solo lettura immagini articoli
- Non è più necessario gestire il cookie per installazioni Live
- Nell'entità di tipo ricerca aggiunta la condizione "contiene"

Nota:

- nei prossimi rilasci di WebAPI i nomi delle chiavi dei JSON verranno cambiati gestendo un massimo di 15 caratteri. In questo modo i nomi degli elementi risulteranno più parlanti e più aderenti ad una logica RESTful. Nel caso siano già state fatte applicazioni con la nomenclatura attuale, di volta in volta verranno rilasciate delle tabelle di correlazione tra i vecchi ed i nuovi nomi dei vari archivi aggiornati. In ogni caso per i dettagli sui nuovi nomi dei campi si può usare il metodo GET con il parametro *info=true* in query string.

Correzione refusi:

- esec_collage_server_remoto: era stato erroneamente indicato esegui_collage_server_remoto