

eF API 3.0 - Servizi per l'interoperabilità applicativa - V3

Indice cambiamenti		
Versione	Modifica apportata	Reviewer
1.0.0	Stesura iniziale	RP
1.0.1	- aggiunto precisazione formato date iso 8601 - aggiunto (*) per indicare un parametro obbligatorio - specificato i possibili valori di returnCode	RP
1.1.0	- Aggiunta documentazione CALENDAR 2 - Migliorie descrittive risorse jax-rs	RP
1.1.1	Aggiunta sezione contenente i parametri di configurazione a database per dominio	RP
1.1.2	Aggiungo Anchor per "parametri di configurazione"	RP
1.1.3	Correzione descrittore method get/v1	RP
1.1.4	Aggiunti jndi di riferimento dei db	RP
1.1.5	Aggiungi progetti SoapUI aggiornati	RP
1.2.0	Specifica Gestione errori	RP
1.2.1	Aggiunta content-type per metodi, aggiornamento imprecisione su gestione eccezioni di Exception	RP
1.2.2	Aggiungo progetto Postman	RP
1.2.3	Aggiunto supporto ordinamento records	RP
1.3.1	Aggiunta funzionalità di download risorse	RP
1.5.0	Aggiunta documentazione per servizi lookup/by-user e set-status	RP
1.6.0	Aggiunta documentazione per nuovi servizi di utilità a partire dal qrCode	RP
1.6.1	Aggiunta documentazione per nuovo servizio esposto "riapri"	RP
1.6.2	Aggiungo parametro opzionale "requestStatusDescription" alla "setCompletedRequestStatus"	RP
1.6.2	SWF aggiornabile autonomamente dal cliente; modificato HTTP method da PUT a POST	RP
1.7.0	Spostamento annotazioni EMF in common; introduzione EFTL© support; aggiornamento archetipo; bug fix on moduleTag empty	RP
1.7.1	Cambiamento semantica di INOLTRATA in list by status; aggiunto INOLTRATA_SWF_NON	RP
1.7.2	Esposizione servizio di processazione documenti EFTL© EFTL©	RP
2.2.0	Aggiunto alla Lookup la possibilità di unire più step di ricerca	RP
2.4.0	Introduzione invio comunicazione formale all'utente	RP
2.4.1	Clonazione di una request	RP
2.4.5	Aggiunta del nuovo concetto di versionamento; aggiunta versione V1.1 per i servizi di interoperabilità ove richiesto; correzione typo <i>mutipleForms</i> in multipleForms ; aggiunto nodo/property moduleCategoryTag .	RP
2.4.6	Aggiunta servizi JWT	LC
2.4.7	Aggiunta servizi Codice Fiscale e progetto di Validazione	LC
3.0.0	Porting infrastruttura a JDK 17 LTS (JDK 21 LTS ready), Jakarta 10 e Payara 6 (Genera questo documento, MA rimane trasparente all'utenza; nessuna nuovo funzionalità) API V3	RP
3.0.1	Aggiunta servizi Codice Fiscale e progetto di Validazione	LC
3.0.2	Aggiunta servizi JWE	LC

3.3.0	Aggiunta nuovi servizi lookup-status (deprecata versione v1 e rivista in v1_1) e lookup-period (since v1_1); aggiunto moduleId a request/.../get (struttura v1_1) API-252 - Look Up By Date - Su tutte le domande REVIEW API-255 - Implementazione chiamata API - ADD RWE2 - Request message REVIEW API-273 - GET - Inserire URL di accesso al modulo nel JSON CHIUSO	RP
3.3.0	Aggiunto servizio per indicare una diversa chiave da usare nel controllo firma di un JWT	LC
3.4.0	Aggiunto servizio di validazione indirizzo email	LC
3.5.0	Servizio per ricevute login SSO	LC
3.7.1	Reso ufficiale formato XML ove era referenziato "per il futuro", perchè implementato dalla v3.3.0	RP
3.7.2	Per le chiamate: - Lookup by-user-and-period - Lookup by-period Aggiornato il metodo da "POST" a "GET"	LB
3.9.1	Dichiarazione v1.2 CDM per le funzionalità lookup request (by user e non)	RP
3.9.1	Per la chiamata: - Lookup by-user Rimosso markupwiki con refuso nell'intestazione della sezione Per le chiamate: - Lookup by-status - Lookup by-period - Lookup by-user - Lookup by-user-and-period Aggiunta specifica sull'utilizzo del parametro "swfOptionMode"	LB

- [INTRODUZIONE](#)
- [DINAMICA UTENTE DELLA COMPILAZIONE](#)
- [REQUISITI](#)
- [ACCESSO E SCAMBIO](#)
- [MODELLO DI IMPIEGO](#)
- [Gestione delle versioni strutturali](#)
- [Release Notes](#)
- [Autenticazione](#)
 - [Login](#)
 - [Logout](#)
 - [SSO Receipt](#)
- [JWT](#)
 - [Creazione](#)
 - [Lettura e validazione](#)
 - [Lettura e validazione con diversa chiave](#)
- [JWE](#)
 - [Creazione](#)
 - [Creazione da String](#)
 - [Lettura e validazione](#)
 - [Lettura e validazione in String](#)
- [Utility](#)
 - [GetUrl](#)
 - [Codice Fiscale](#)
 - [Indirizzo Email](#)
- [Recupero delle istanze inoltrate](#)
 - [Lookup by-status](#)
 - [Lookup by-date \(deprecato per rimozione futura, since v1_1: utilizzare Lookup-by-period\)](#)
 - [Lookup by-period](#)
 - [Lookup by-user](#)
 - [Lookup by-user-and-period](#)
 - [Get request identifier](#)
 - [Get request](#)
 - [Get request by QRCode](#)
 - [Get Document request](#)
 - [Get User Infos of existing request](#)
- [Inserimento/verifica di codici/informazioni applicative nel modulo](#)
- [Comunicazioni di cambio stato e/o richieste di integrazione attraverso la console utente eF](#)
- [FUNZIONE RIAPERTURA DOMANDA 2.0](#)

- [FUNZIONE CLONAZIONE DI UNA DOMANDA 2.0](#)
- [COMUNICAZIONI FORMALI](#)
 - [Invio comunicazione formale all'utente](#)
- [FUNZIONE CALENDAR 2.0](#)
 - [Calendar 2: get appointments](#)
 - [Calendar 2: set-acquired appointment](#)
- [FUNZIONE EFTL© - Elixforms© Tag Language](#)
 - [EFTL©: process document](#)
- [GESTIONE ERRORI](#)

INTRODUZIONE

La piattaforma **elixForms©** (eF*) consente all'ente di realizzare modelli **web forms** per l'inoltro di istanze in modo *generico* e secondo un *pattern* definito.

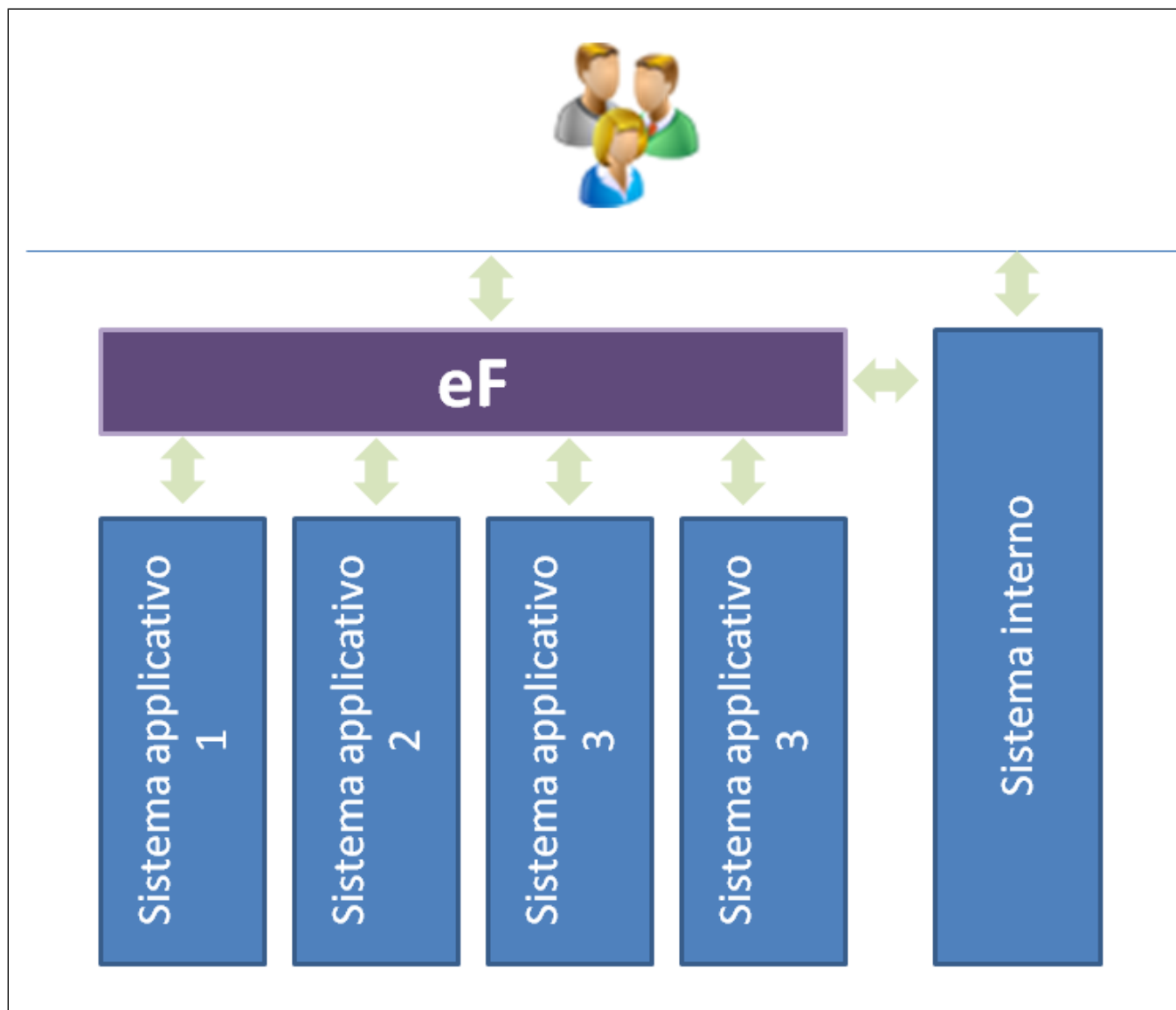


Identificazione -> Compilazione -> Sottoscrizione/Convalida -> Eventuale pagamento -> Inoltro (con protocollazione)

Secondo questo modello ogni istanza dell'utente può essere liberamente modellata, sia nella struttura dati da richiedere sia nei controlli da effettuare.

Il sistema consente, inoltre di guidare l'utente raccogliendo informazioni direttamente o acquisendole da più sistemi "esterni", verificandole su sistemi terzi o allegandole sotto forma di documenti (anche firmati digitalmente).

Secondo questo modello, l'ente può implementare uno strato di costruzione di raccolta di istanze digitali "standard", contenenti sia i dati sia la documentazione allegata, anche se riferito a diverse esigenze e sistemi applicativi.



eF* fornisce tutte le funzionalità di supporto online alla gestione del colloquio mirato con l'utenza, a supporto della corretta e completa compilazione /costruzione (multi-lingua, help desk, riaperture, protocollazioni multiple).



Il modello eF* (elixForms©) implementa la visione dell'agenda digitale dalla produzione dell'istanza all'autorizzazione full-digital secondo quanto previsto dal CAD.

Secondo questo modello, **eF***, infatti, è in grado di fornire una serie di servizi per l'interoperabilità applicativa, volti ad interfacciare sistemi applicativi specifici sia interni che di terze parti.

DINAMICA UTENTE DELLA COMPILAZIONE

eF* consente di modellare e compilare le istanze divise per step, liberamente implementabili.

Ogni step deve essere compilato e validato per poter essere chiuso ed inoltrabile.

Solo se tutti gli step saranno chiusi, l'istanza potrà essere inoltrata.

La validazione avviene tramite l'esecuzione di plugin standard di piattaforma (Es. sui controlli formali ed incrociati) e quelli specifici di integrazione che chiamano sistemi applicativi esterni.

REQUISITI

Ogni sistema esterno può quindi avvalersi di tali servizi per integrare o costruire la componente di alimentazione istanze in modalità *document exchange* senza l'interazione con l'utente, ma con l'acquisizione delle sue istanze come fascicoli digitali secondo il modello ad esempio della fatturazione elettronica. Risolta da **eF*** la dinamica di interazione con l'utente per la raccolta di dati corretti, l'applicazione potrà contare su dati strutturati secondo una struttura specifica per istanza, realizzata e mantenuta dall'ente in maniera rapida (modifiche della struttura dati, controlli, ecc.), ma secondo un preciso pattern che consente la repentina identificazione di come e dove i dati saranno strutturati.

ACCESSO E SCAMBIO

L'accesso ai servizi avviene in modalità Web Services restful in modalità HTTPS.

Ogni sistema applicativo terzo può utilizzare per l'accesso un token di sicurezza ottenuto con credenziali fornite dall'ente. E' possibile incrementare il livello di sicurezza implementando il colloquio su canali sicuri diretti punto-punto (VPN).

MODELLO DI IMPIEGO

Sono 3 i livelli di interazione tra i sistemi applicativi ed **eF*** necessari per la corretta interoperabilità:

1. **Recupero delle istanze inoltrate**

Il sistema applicativo deve ricevere le istanze, recuperandole in modalità pull da eF

2. **Inserimento e verifica di codici o informazioni applicative nel modulo**

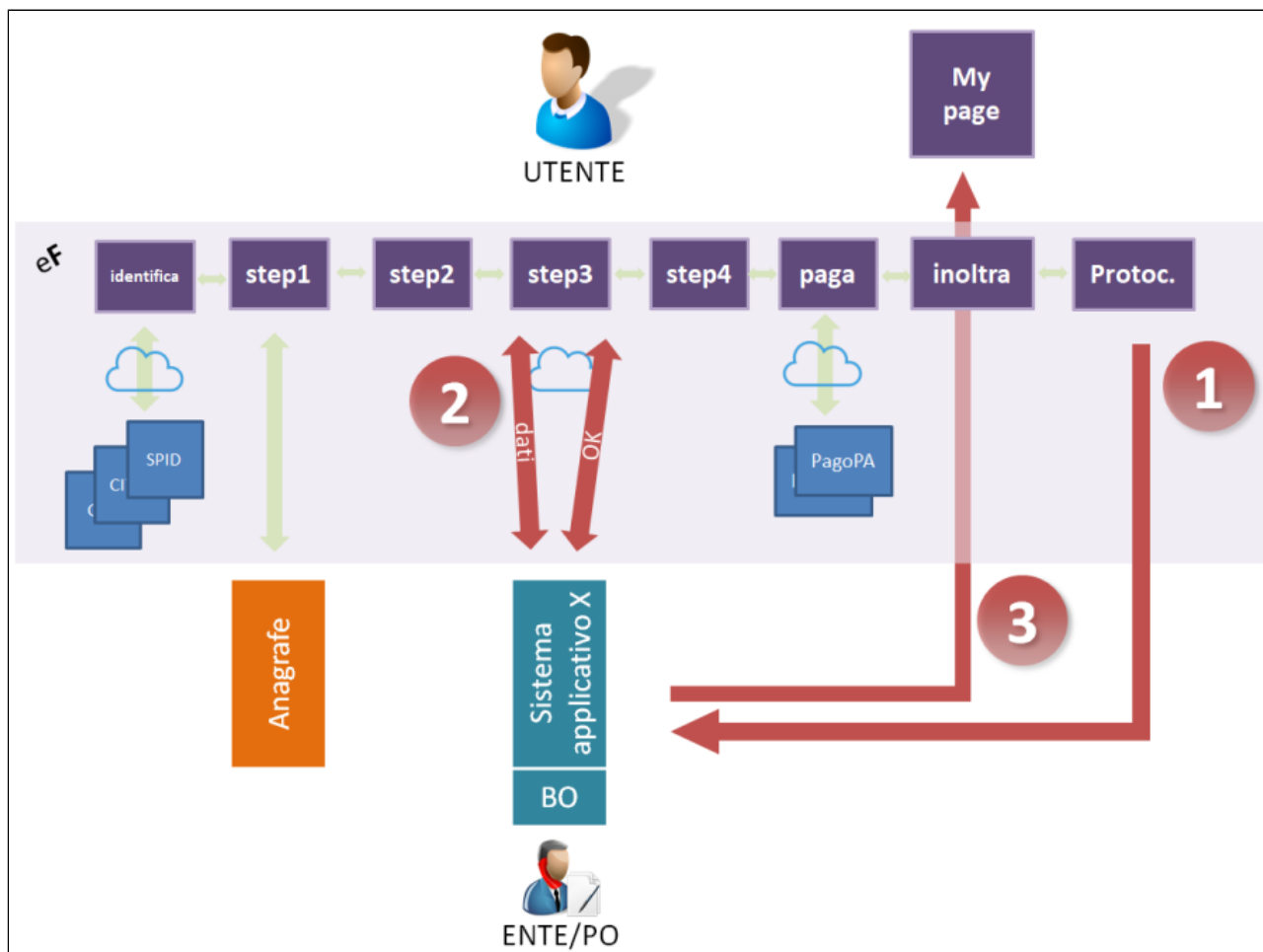
eF* si occupa della corretta compilazione proponendo codifiche e dati propri del sistema applicativo esterno in tempo reale, evitando ridondanze e sincronizzazioni

3. **Comunicazioni di cambio stato e/o richieste di integrazione attraverso la console utente eF**

Il sistema applicativo può interagire con la console utente eF, se in uso, per segnalare esiti o cambi di stato della pratica

4. **Eliminazione istanze online**

La corretta acquisizione delle istanze online e dei relativi allegati, consente di effettuare le periodiche eliminazioni delle copie online delle istanze, in ottemperanza a quanto previsto dal regolamento sulla privacy.



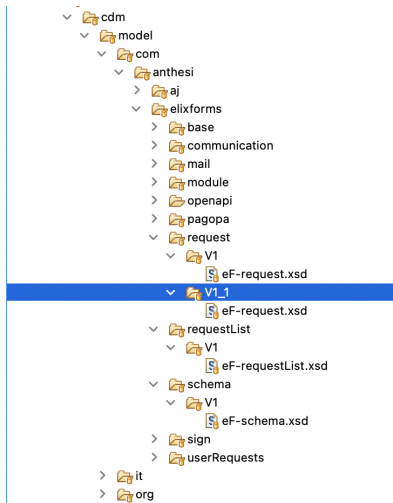
Gestione delle versioni strutturali

La gestione delle versioni nelle chiamate API di interoperabilità nasce sin dalla creazione degli "endpoints" grazie al path info ".../v1" - suffisso comune a tutte le chiamate API di interoperabilità - come ultimo componente del percorso REST.

Il concetto di "Version" è legata a modifiche strutturali della risposta che le **resources** REST, o altri tipi di servizi, espongono: le entità in risposta seguono, infatti, una logica di evoluzione ben definita e controllata da un processo di rilascio noto come **Common Data Model** (da ora in poi anche denominato **CDM**, ossia modello dati di interoperabilità comune), secondo il quale vengono definite tramite il linguaggio xml schema (xsd) le **strutture** di INTEROPERABILITÀ che avranno le entities restituite, non solo dalle resources REST esposte, ma anche da altre eventuali funzionalità di natura SOA esterna o interna che sia.

Common Data Model

Grazie agli xsds disponibili in CDM ed alla loro caratteristica di trasferibilità indipendente dalla piattaforma in uso, infatti, è possibile generare il sorgente delle entities in linguaggi di programmazione diversi, assolvendo qualunque necessità, pur mantenendo la certezza della struttura così generata, ossia la certezza di mantenere la compatibilità di "binding" con un servizio che sfrutta lo stesso CDM.



Il CDM di Anthesi® è disponibile in file compresso [cdm.zip](#), distribuibile a chi necessita generare il codice API per Elixforms®, ma a breve sarà esposto pubblicamente in **cdn** di Anthesi® in modo che la risorsa xsd sia raggiungibile tramite il suo reale namespace dichiarativo.

CDM "Versioning" e risorse "Deprecated"

Le versioni CDM delle API Elixforms® seguono il seguente concetto, ripreso dalle più comuni "best practices" a riguardo:

v<major>.<minor>.<patch> (v1.1.0 per es.)

e la numerazione segue la nota catena di aggiornamento

- **major:** aggiornamenti evolutivi che provocano la necessità di modifiche al codice ai clients del/i servizi interessati per
 - una "breaking change", termine noto in programmazione per definire un cambiamento potenzialmente NON retrocompatibile con la struttura CDM precedente;
 - cambio strutturale di piattaforma
 - evolutive con impatto sull'esposizione (cambio url e/o pathinfo etc)
 - introduzione, modifica o eliminazioni di target namespaces nel CDM
- **minor:** modifiche di struttura per le quali vengono
 - o aggiunti o rimossi nodi
 - o che prevedano aggiornamenti evolutivi
 - o, ancora, eliminazione di elementi **deprecati** (dichiarati come vetusti, ma temporaneamente lasciati per non provocare problemi di retrocompatibilità)
- **patch:** modifiche minori come errori ortografici, nodi in posizioni sbagliate e via dicendo

Come si può notare nell'esploso CDM qui sopra, la risorsa **request** per esempio consta di una version **V1** e una sua evolutiva **V1_1**, forme sincopate per intendere versioni 1.0.0 e 1.1.0.

Il ciclo di aggiornamento del CDM in Elixforms® prevede che vengano sempre mantenute **massimo 3 versioni contemporaneamente:**

1. una versione corrente, detta anche di **default**
2. una versione obsoleta per la retrocompatibilità, in modo da favorire in breve lasso di tempo il passaggio alla nuova versione dei clients senza provocare disservizi, ma **deprecata**.
3. una versione **deprecata 4 removal:** versione precedente alla versione (2), sempre per retrocompatibilità, che verrà **rimossa** all'emissione della prossima nuova versione.

Le versioni precedenti vengo dichiarate deprecate all'uscita di una nuova versione quantomeno **minor**, che include tutte le patches della versione precedente, o di una versione **major** che contiene tutte le modifiche e evolutive della minor precedente.

L'attenzione dei customers a tali aggiornamenti e rimozioni sarà notificato dai canali ufficiali Anthesi®, in modo tale che possano reagire per tempo all'eventuale aggiornamento dei pacchetti client.

Relazione tra CDM e versioni REST

Grazie all'architettura del **CDM**, si crea il legame tra il suffisso (".../V1") e la versione V_1 del **CDM**, infatti, qualunque client API invoca, sino al Settembre 2023, in maniera dichiarativa la versione 1.0.0 (versione strutturale) delle API 2.0 (versione commerciale).

Release Notes

Versione 1.1 (http://www.elixforms.it/cdm/model/request/V1_1, etc)

Si è deciso di evolvere ulteriormente il concetto di **Versioning** e del **"motore"** (i.e. l'implementazione) che lo gestisce per le risorse REST esposte, in modo da renderlo più flessibile nell'uso e, allo stesso tempo, meno impattante per la manutenzione evolutiva dei clients.

- Viene definita la nuova modalità di Versioning ove la versione **NON dipende più dal pathinfo**, ma **dall'header "Accept"**. Sfruttando infatti la RFC-2295 (<https://datatracker.ietf.org/doc/html/rfc2295>) è possibile utilizzare il "vendor" che trasporterà non solo l'informazione sulla **negoiazione del contenuto**, ma anche della sua **versione**.
 Si definisce il maniera "statica" il vendor: **it.elixforms.api.vX.X.X(+)contentType** (dove X è una cifra che definisce la versione rispettivamente major, minor o patch)
 La versione sarà quindi così specificata:
 - se l'header **Accept** **NON** è **definito**: viene restituito il contenuto in formato **json** e versione di default 1.1.0
 - se l'header **Accept** non ha vendor e versione, viene trattato in maniera classica sulla negoziazione del contenuto:
 - se "accept: application/json": viene restituito il formato json con versione di default 1.1.0
 - se "accept: application/xml": viene restituito il formato xml con versione di default 1.1.0
 - se l'header **Accept** ha vendor e versione:
 - se "accept: application/it.elixforms.api.v1.1.0+json": viene restituito il formato **json** con versione di default **1.1.0**
 - se "accept: application/it.elixforms.api.v1.1.0+xml": viene restituito il formato **xml** con versione di default **1.1.0**
 - se "accept: application/it.elixforms.api.v1.0.0+json": viene restituito il formato **json** con versione **1.0.0** (stessa versione strutturale di quella invocata con pathinfo **"/v1"**, ma in nuova modalità)
 - se "accept: application/it.elixforms.api.v1.0.0+xml": viene restituito il formato **xml** con versione **1.0.0** (stessa versione strutturale di quella invocata con pathinfo **"/v1"**, ma in nuova modalità)
- Dalla versione 1.1.0, viene eliminato il path **/services** vecchio **".../eF/services/api..."**, nuovo path **".../eF/api..."**
- Viene **deprecata for removal** la **V_1** ("**/v1** in pathinfo"), ma continuerà ad esistere sino a sua rimozione, come spiegato nel paragrafo precedente
- Viene imposta programmaticamente la versione di default alla **versione V1_1** che nella struttura di **CdmRequestType** produce le seguenti modifiche:
 - correzione sintattica del nodo **"mutipleForms"** in **"multipleForms"**
 - aggiunta del nodo **"moduleCategoryTag"**
- Rimozione del suffisso **"/v1"** dal PathInfo delle risorse CDM di **request** per la versione **1.1.0**;
- Rimozione del suffisso **"/v1"** dal PathInfo delle risorse CDM di **request** per nuova modalità (utilizzo di accept) in **v1.0.0**, deprecata per creare la retro-compatibilità (mantenuto chiaramente in versione originale).

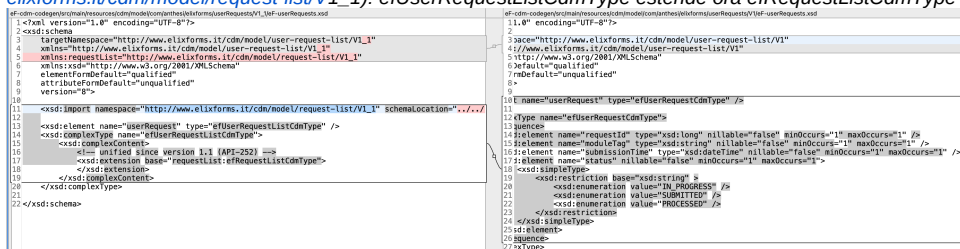
A seguire nel documento, vengono riportate le modifiche di quelle risorse il cui contenuto dipende dalla versione 1.1.0 di **CdmRequestType**, **CdmRequestListType** etc, ossia che rispondono alla nuova modalità.

Modifiche intervenute nella versione strutturale CDM 1_1

- Domain request: <http://www.elixforms.it/cdm/model/requestV1> http://www.elixforms.it/cdm/model/requestV1_1
 - efRequestCdmType**:
 - si aggiunge elemento **"moduleUrl"**: url di accesso al modulo della request
 - si aggiunge elemento **"moduleCategoryTag"**: tag della categoria del modulo sotteso dalla request
 - efStepCdmType**
 - typo fix: viene corretto l'errore ortografico dell'elemento **"mutipleForms"** in **"multipleForms"**
- Domain lookup request: <http://www.elixforms.it/cdm/model/request-list/V1> http://www.elixforms.it/cdm/model/request-list/V1_1
 - efRequestListCdmType** viene ripensata la risposta della ricerca delle request e unificata tra i vari tipi di "lookup":
 - by-date**: viene deprecata e disponibile solo in v1
 - by-status**: mantiene struttura originale in v1; viene rivista in v1_1 come segue:
 - requestTitle** diventa **moduleTitle**
 - step** viene rimosso per implementare i nuovi **requestStatus** e **swfOptionKey**: lo stato della request è ora disgiunto dall'eventuale stato di SWF (Simple workflow)
 - si introduce **submissionTime**
 - si introduce **hiddenToUser**
 - by-period**: introdotta in vece di by-date con nuova struttura di risposta del by-status (vedere documento API (per le specifiche)
- Domain lookup user request: <http://www.elixforms.it/cdm/model/user-request-list/V1> http://www.elixforms.it/cdm/model/user-request-list/V1_1

<pre> 1 <?xml version="1.0" encoding="UTF-8"?> 2 <xsd:schema 3 targetNamespace="http://www.elixforms.it/cdm/model/request-list/V1" 4 xmlns="http://www.elixforms.it/cdm/model/request-list/V1" 5 xmlns:xsd="http://www.w3.org/2001/XMLSchema" 6 elementFormDefault="qualified" 7 attributeFormDefault="unqualified" 8 version="0"> 9 10 <xsd:element name="requestList" type="efRequestListCdmType" /> 11 12 <xsd:complexType name="efRequestListCdmType"> 13 <xsd:sequence> 14 <xsd:element name="requestId" type="xsd:long" nillable="false" minOccurs="1" maxOccurs="1" /> 15 <xsd:element name="moduleTitle" type="xsd:string" nillable="false" minOccurs="1" maxOccurs="1" /> 16 <xsd:element name="moduleTag" type="xsd:string" nillable="false" minOccurs="1" maxOccurs="1" /> 17 <xsd:element name="submissionTime" type="xsd:dateTime" nillable="false" minOccurs="1" maxOccurs="1" /> 18 <xsd:element name="requestStatus" nillable="false" minOccurs="1" maxOccurs="1"> 19 <xsd:simpleType> 20 <xsd:restriction base="xsd:string"> 21 <xsd:enumeration value="IN_PROGRESS" /> 22 <xsd:enumeration value="SUBMITTED" /> 23 <xsd:enumeration value="PROCESSING" /> 24 </xsd:restriction> 25 </xsd:simpleType> 26 </xsd:element> 27 <xsd:element name="swfOptionKey" type="xsd:integer" nillable="false" minOccurs="0" maxOccurs="1" /> 28 <xsd:element name="hiddenToUser" type="xsd:boolean" nillable="false" minOccurs="1" maxOccurs="1" /> 29 </xsd:sequence> 30 </xsd:complexType> 31 32 </xsd:schema> </pre>	<pre> 1 <?xml version="1.0" encoding="UTF-8"?> 2 <xsd:schema 3 targetNamespace="http://www.elixforms.it/cdm/model/request-list/V1" 4 xmlns="http://www.elixforms.it/cdm/model/request-list/V1" 5 xmlns:xsd="http://www.w3.org/2001/XMLSchema" 6 elementFormDefault="qualified" 7 attributeFormDefault="unqualified" 8 version="0"> 9 10 <xsd:element name="requestList" type="efRequestListCdmType" /> 11 12 <xsd:complexType name="efRequestListCdmType"> 13 <xsd:sequence> 14 <xsd:element name="requestId" type="xsd:long" nillable="false" minOccurs="1" maxOccurs="1" /> 15 <xsd:element name="requestTitle" type="xsd:string" nillable="false" minOccurs="1" maxOccurs="1" /> 16 <xsd:element name="moduleTag" type="xsd:string" nillable="false" minOccurs="1" maxOccurs="1" /> 17 <xsd:element name="step" type="xsd:string" nillable="false" minOccurs="1" maxOccurs="1" /> 18 </xsd:sequence> 19 </xsd:complexType> 20 21 </xsd:schema> </pre>
--	---

- **efUserRequestListCdmType** la struttura della risposta si adegua in v1_1 al namespace di **efRequestListCdmType** (http://www.elixforms.it/cdm/model/request-list/v1_1): **efUserRequestListCdmType** estende ora **efRequestListCdmType**



Versione 1.0 (<http://www.elixforms.it/cdm/model/request/v1>, etc)

Versione strutturale come nota ad oggi, Settembre 2023 e nata nel Giugno 2019. Moltissimi sono stati gli interventi effettuati di Business Logic (BL) e migliorie tecniche nonché implementative, ma sinora non impattanti la struttura esposta

Autenticazione

Login



Login

Per accedere ai servizi e' necessario fornire le credenziali di un utente appartenente al gruppo "RWE2 – Accesso servizi Rest"

[https://\[host\]/eF/services/api/authentication/login/v1](https://[host]/eF/services/api/authentication/login/v1)

http method: POST

content-type: application/x-www-form-urlencoded o application/json

form parameters:

username(*)=<username>

password(*)=<password>

ritorna: **authToken** da utilizzare nelle chiamate successive come metodo di autenticazione.



Ogni invoke del servizio, se invocato durante una sessione attiva, invalida il precedente token assegnato anche se non ancora scaduto generandone uno nuovo; come conseguenza le chiamate usando il precedente token vengono respinte con un HTTP STATUS 403 (*Forbidden*).

Per default, configurabile, il timeout del token è di 30 minuti; il timeout viene azzerato ad ogni nuova richiesta valida autenticata (i.e. stessa modalità delle sessioni java).

Logout



Logout

Per invalidare l'**authToken** è necessario invocare la Logout

[https://\[host\]/eF/services/api/authentication/{username}/logout/v1](https://[host]/eF/services/api/authentication/{username}/logout/v1)

http method: POST con **authToken** header ("Authorization: Bearer <token>")

pathInfo parameter:

username(*)=<username>

content-type: application/x-www-form-urlencoded

SSO Receipt



Logout

Permette di recuperare la ricevuta di un login avvenuto via SSO.

https://[host]/eF/api/authentication/sso/receipts/users/{codiceFiscale}

http method: GET con authToken header ("x-api-key: <token>")

pathInfo parameter:

codiceFiscale(*)=codice fiscale dell'utente di cui cercare le ricevute

content-type: application/json



Logout

Permette di inserire la ricevuta di un login avvenuto via SSO.

https://[host]/eF/api/authentication/sso/receipts

http method: POST con authToken header ("x-api-key: <token>")

content-type: application/json

body content: la ricevuta SSO decorata in una struttura SsoReceipt

```
{
  "elixReceipt": "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?><EXT NEXT_FIELD_KEY=\"COL0022\" NE...\", // la scheda XML salvata in elixForms per l'autenticazione
  "originIp": "192.168.200.172",
  "referrerHost": "https://posteid.poste.it/",
  "requestedHost": "dev.elixpreprod.it",
  "requestId": 571181,
  "ssoData": "{\"userId\":\"1088\",\"ssoSchemaGenericId\":\"852362\",\"ssoName\":\"RWE2 - Autenticazione...\",
  // Il json mantenuto in sessione di elixForms dopo login SSO (SsoLoginInfo)
  "ssoHttpHeaders": "{\"host\":\"elixpreprod01.ovh.anthesi.com:8080\",\"cache-control\":\"max-age=0\",
  \"upgrade-insecure-requests\":\"1\",...\", // gli header della richiesta
  "ssoType": "RWE2 - Autenticazione AuthService",
  "userId": 1088
}
```

JWT

E' stata introdotta la gestione di lettura e produzione di token secondo lo standard JWT ([RFC-7519](#)).

Creazione



Creazione JWT

Crea un token JWT con informazioni standard e firmato con la chiave di cifratura di Anthesi, definita nell'apposita tabella con nome "jwt-anthesi-key". La validità del token è di 2 ore.

https://[host]/eF/services/api/authentication/jwt

http method: POST con authToken header ("x-api-key: <token>")

content-type: application/json

body content: informazioni del contenuto denominato "payload" da inserire nel JWT.

Esempio

```
{
  "iss": "servizio-cittadino-attivo",
  "sub": "mariorossi",
  "info": {
    "https://www.elixforms.it/name": "Mario",
    "https://www.elixforms.it/surname": "Rossi",
    "https://www.elixforms.it/email": "mario.rossi@example.org",
    "https://www.elixforms.it/fiscal_code": "XXX",
    "https://www.elixforms.it/login_date": "2023-08-09T11:31:30Z[UTC]",
    "https://www.elixforms.it/login_receipt": "SPIDlogin 0101001029"
  }
}
```

ritorna: stringa alfanumerica rappresentante il JWT

Lettura e validazione



Lettura

Legge il token JWT fornito in ingresso, validandone le informazioni e controllando la scadenza. La firma viene controllata utilizzando le chiavi: per sapere qualche chiave utilizzare è possibile indicare nel header del JWT il campo "kid" oppure il campo custom "<https://www.elixforms.it/jwtkeyname>" oppure il campo "iss".

[https://\[host\]/eF/services/api/authentication/jwt/payload](https://[host]/eF/services/api/authentication/jwt/payload)

http method: GET con authToken header ("x-api-key: <token>")

content-type: application/text

header-parameter: "jwt" contenente la stringa JWT da analizzare

ritorna: application/json con il payload del JWT

Esempio ritorno

```
{
  "prgRedirectUrl": null,
  "value": {
    "code": "OK",
    "description": "JWT read",
    "complete": true,
    "globalStatus": "OK",
    "resultNumber": null,
    "serviceInformationList": {
      "serviceInformation": [
        {
          "ITSystemName": "CoreJwtService$Proxy$_$$_WeldSubclass",
          "complete": true,
          "serviceVersion": "1.1.0",
          "statusDetailList": {
            "statusDetail": [
              {
                "code": "OK",
                "description": "JWT read",
                "detailStatus": "OK"
              }
            ]
          }
        }
      ]
    }
  },
  "uuid": "9e4d7d68-0588-4527-8dd7-73047040be8a",
  "version": "3.0.0.-1",
  "matchingEntitiesSize": 1,
  "jwtPayload": {
    "aud": null,
    "exp": "2024-01-02T10:17:04Z[UTC]",
    "header": {
      "partnerId": "Anthesi"
    },
    "iat": "2024-01-02T08:17:04Z[UTC]",
    "info": {
      "https://www.elixforms.it/email": "mario.rossi@example.org",
      "https://www.elixforms.it/login_receipt": "SPIDlogin 0101001029",

```

```
{
  "https://www.elixforms.it/surname": "Rossi",
  "https://www.elixforms.it/login_date": "2023-08-09T11:31:30Z[UTC]",
  "https://www.elixforms.it/name": "Mario",
  "https://www.elixforms.it/fiscal_code": "XXX"
},
"iss": "servizio-cittadino-attivo",
"jti": "AGeVyw28wHtGZFPltwxxpw",
"nbf": null,
"sub": "mariorossi"
}
}
```

Lettura e validazione con diversa chiave



Lettura

Legge il token JWT fornito in ingresso, validandone le informazioni e controllando la scadenza. La firma viene controllata utilizzando la chiave indicata.

*https://[host]/eF/services/api/authentication/jwt/keys/

Unknown macro: {keyName}

/payload*

http method: GET con authToken header ("x-api-key: <token>")

path-parameter: "keyName" nome della chiave in DB da usare

content-type: application/text

header-parameter: "jwt" contenente la stringa JWT da analizzare

ritorna: application/json con il payload del JWT

Esempio ritorno

```
{
  "prgRedirectUrl": null,
  "value": {
    "code": "OK",
    "description": "JWT read",
    "complete": true,
    "globalStatus": "OK",
    "resultNumber": null,
    "serviceInformationList": [{"serviceInformation": {
      "ITSystemName": "CoreJwtService$Proxy$_$_WeldSubclass",
      "complete": true,
      "serviceVersion": "1.1.0",
      "statusDetailList": [{"statusDetail": {
        "code": "OK",
        "description": "JWT read",
        "detailStatus": "OK"
      }}]
    }}]
  },
  "uuid": "9e4d7d68-0588-4527-8dd7-73047040be8a",
  "version": "3.0.0.-1",
  "matchingEntitiesSize": 1,
  "jwtPayload": {
    "aud": null,
    "exp": "2024-01-02T10:17:04Z[UTC]",
    "header": {"partnerId": "Anthesi"},
    "iat": "2024-01-02T08:17:04Z[UTC]",
    "info": {
```

```
{
  "https://www.elixforms.it/email": "mario.rossi@example.org",
  "https://www.elixforms.it/login_receipt": "SPIDlogin 0101001029",
  "https://www.elixforms.it/surname": "Rossi",
  "https://www.elixforms.it/login_date": "2023-08-09T11:31:30Z[UTC]",
  "https://www.elixforms.it/name": "Mario",
  "https://www.elixforms.it/fiscal_code": "XXX"
},
{
  "iss": "servizio-cittadino-attivo",
  "jti": "AGeVyw28wHtGZFPltwxxpw",
  "nbf": null,
  "sub": "mariorossi"
}
}
```

JWE

E' stata introdotta la gestione di lettura e produzione di token JWT cifrati secondo lo standard JWE ([RFC-7516](#)).

Creazione

Creazione JWT

Crea un token JWE con informazioni standard e cifrato con la chiave indicata per nome al momento della chiamata, definita nell'apposita tabella con nome "jwe-anthesi-key". La validità del token è di 2 ore.

[https://\[host\]/eF/services/api/authentication/jwe](https://[host]/eF/services/api/authentication/jwe)

http method: POST **con** authToken header ("x-api-key: <token>")

content-type: application/json

body content: informazioni del contenuto denominato "payload" da inserire nel JWE e della chiave da usare per decifrare (keyName).

Esempio

```
{
  "keyName" : "jwe-anthesi-key",
  "payload" : {
    "iss" : "servizio-cittadino-attivo",
    "sub" : "mariorossi",
    "info" : {
      "https://www.elixforms.it/name" : "Mario",
      "https://www.elixforms.it/surname" : "Rossi",
      "https://www.elixforms.it/email" : "mario.rossi@example.org",
      "https://www.elixforms.it/fiscal_code" : "PRTL76C01D325T",
      "https://www.elixforms.it/login_date" : "2023-08-09T11:31:30Z[UTC]",
      "https://www.elixforms.it/login_receipt" : "SPIDlogin 0101001029"
    }
  }
}
```

ritorna: stringa alfanumerica rappresentante il JWE

Creazione da String

Creazione JWE da String

Crea un token JWE considerando come payload una qualsiasi stringa e senza applicare alcuna validazione. Il token viene cifrato con la chiave indicata per nome al momento della chiamata, definita nell'apposita tabella con nome "jwe-anthesi-key". La validità del token è di 2 ore.

[https://\[host\]/eF/services/api/authentication/jwe/string](https://[host]/eF/services/api/authentication/jwe/string)

http method: POST con authToken header ("x-api-key: <token>")

content-type: application/json

body content: informazioni del contenuto denominato "payload" da inserire nel JWE e della chiave da usare per decifrare (keyName).

Esempio

```
{
  "keyName" : "jwe-anthesi-key",
  "payload" : "il-contenuto-come-string"
}
```

ritorna: stringa alfanumerica rappresentante il JWE

Lettura e validazione



Lettura

Legge il token JWE fornito in ingresso, decifrando le informazioni e controllando la scadenza. La decifratura viene eseguita utilizzando la chiave standard denominata "jwe-anthesi-key".

https://[host]/eF/services/api/authentication/jwe/payload

http method: GET con authToken header ("x-api-key: <token>")

content-type: application/text

header-parameter: "jwe" contenente la stringa JWE da analizzare

ritorna: application/json con il payload del JWE

Esempio ritorno

```
{
  "prgRedirectUrl" : null,
  "value" : {
    "code" : "OK",
    "description" : "JWT read",
    "complete" : true,
    "globalStatus" : "OK",
    "resultNumber" : null,
    "serviceInformationList" : {
      "serviceInformation" : [
        {
          "ITSystemName" : "CoreJwtService$Proxy$$_weldSubclass",
          "complete" : true,
          "serviceVersion" : "1.1.0",
          "statusDetailList" : {
            "statusDetail" : [
              {
                "code" : "OK",
                "description" : "JWT read",
                "detailStatus" : "OK"
              }
            ]
          }
        }
      ]
    }
  },
  "uuid" : "2cbdc574-de22-4461-87ca-c204ec0e9b96",
  "version" : "3.3.0.-1",
  "matchingEntitiesSize" : 1,
  "jwtPayload" : {
    "aud" : null,
    "exp" : "2024-02-07T15:54:49Z[UTC]",

```

```
{
  "header" : {
    "partnerId" : "Anthesi"
  },
  "iat" : "2024-02-07T13:54:49Z[UTC]",
  "info" : {
    "https://www.elixforms.it/email" : "mario.rossi@example.org",
    "https://www.elixforms.it/login_receipt" : "SPIDlogin 0101001029",
    "https://www.elixforms.it/surname" : "Rossi",
    "https://www.elixforms.it/login_date" : "2023-08-09T11:31:30Z[UTC]",
    "https://www.elixforms.it/name" : "Mario",
    "https://www.elixforms.it/fiscal_code" : "PRTLUCU76C01D325T"
  },
  "iss" : "servizio-cittadino-attivo",
  "jti" : "doeMdIZ6HULBcvPRAE-bzw",
  "nbf" : null,
  "sub" : "mariorossi"
}
```

Lettura e validazione in String



Lettura

Legge il token JWE fornito in ingresso, decifrando le informazioni e controllando la scadenza. La decifratura viene eseguita utilizzando la chiave standard denominata "jwe-anthesi-key".

https://[host]/eF/services/api/authentication/jwe/string/payload

http method: GET **con** authToken header ("x-api-key: <token>")

content-type: application/text

header-parameter: "jwe" contenente la stringa JWE da analizzare

ritorna: application/json con il payload del JWE considerato come una String

Esempio ritorno

```
{
  "prgRedirectUrl": null,
  "value": {
    "code": "OK",
    "description": null,
    "complete": true,
    "globalStatus": "OK",
    "resultNumber": null,
    "serviceInformationList": null,
    "uuid": "24d1629a-e8cb-4d00-bfe7-aa9a0f8069b4",
    "version": "3.0.0",
    "matchingEntitiesSize": 1,
    "jwt": "la-string-originale"}
}
```

Utility

In questa sezione vengono presentati i servizi di utilità generica

GetUrl

Porting dal vecchio RWE2 della funzionalità di chiamata URL esterno con inoltro della richiesta e recupero della risposta. In questo modo si disaccoppia il servizio dal chiamante, consentendo cambi ed aggiornamenti infrastrutturali.



Call

Nel caso di protocollo GET, i parametri di query indicati nel URL vengono riportati tali e quali al remoto. Nel caso di POST, lo stesso viene fatto con il body.

https://[host]/eF/services/api/network/call

http method: GET e POST con authToken header ("x-api-key: <token>")

content-type: qualsiasi

ritorna: direttamente il body della risposta remota

Codice Fiscale

Porting dal plugin di validazione RWE2 per creare e controllare formalmente un Codice Fiscale. Il controllo non verifica l'esistenza in nessuna anagrafe, ma solo la correttezza rispetto al formato.



Create

Creazione del codice fiscale sulla base dei dati forniti

https://[host]/eF/api/validation/fiscalCode

http method: GET con authToken header ("x-api-key: <token>")

content-type: application/text

pathInfo parameter:

name=<nome persona>

surname=<cognome persona>

gender=M/F

birthDate=<data di nascita in formato yyyy-MM-dd>

municipalityCode=<codice del comune o stato estero di nascita>

ritorna: il codice fiscale o eventuale errore



Check

Controllo della validità di un codice fiscale

https://[host]/eF/api/validation/fiscalCode/<code>

http method: GET con authToken header ("x-api-key: <token>")

content-type: application/text

urlInfo parameter:

code=<codice fiscale da controllare>

ritorna: JSON con informazioni di validazione

Esempio validazione

```
{
  "failureReason": "",
  "fiscalCode": "RSSMRA70A01F205V",
  "passed": true
}
```


Indirizzo Email

Validazione formale di un indirizzo email. Attualmente implementato secondo lo standard RFC-822 ed utilizzando la libreria Apache Commons Validator.



Create

https://[host]/eF/api/validation/email

http method: GET con authToken header ("x-api-key: <token>")

content-type: application/json

pathInfo parameter:

email=<indirizzo email da validare>

ritorna: struttura "Resul" con campo "isValid" a "true" se l'indirizzo è valido, "false" se non lo è. Un valore non fornito o vuoto per il campo "email" è considerato come NON valido.

Recupero delle istanze inoltrate

L'utente può compilare liberamente la propria istanza in modo progressivo, correggendo e riprendendo la medesima in più occasioni, fino ad arrivare alla definitiva chiusura della stessa, tramite validazione ed inoltro.

eF* versa nel sistema di protocollo l'istanza e gli allegati inoltrati secondo quanto previsto dalla norma e comunica all'utente l'esito e, se positivo, il numero di protocollo.

A supporto del corretto recupero di tutte le istanze vengono quindi messi a disposizione servizi restful con metodi specifici per operare sulle istanze.

Il flusso logico tipico per l'acquisizione delle istanze via via inoltrate da parte di un applicativo di back office prevede il periodico lancio di una funzione che:

1. Identificazione con rilascio token: [login](#)
2. Chiamata per recuperare l'elenco degli id delle istanze non ancora inoltrate: [.../lookup/by-status/](#)
3. Chiamata per ogni id di recupero dei dati della singola istanza [/{id_request}/get](#)
 - a. Identificazione dei campi attraverso le KEY oppure, per il massimo isolamento, con i TAG
 - b. Recupero degli allegati tramite url
4. Conferma di ricezione (positiva o negativa) [/{id_request}/set-acquired](#)

(*) parametro obbligatorio

Lookup by-status



Recupero delle istanze inoltrate

per ottenere l'elenco delle istanze di un modulo in base allo stato:

v1.2 (default): *https://[host]/eF/api/request/lookup/by-status*

- senza header,
- con header "accept:application/json",
- con header "accept:application/xml",
- con header "accept:application/it.elixforms.api.v1.2.0+json"
- con header "accept:application/it.elixforms.api.v1.2.0+xml"
- v1.1 (deprecated): *https://[host]/eF/api/request/lookup/by-status*
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

query parameters:

requestStatus=<lista di stati> possibili valori: **[IN_PROGRESS, SUBMITTED, PROCESSED]** per restringere il risultato alle sole richieste nel /i dato/i stato/i [...] **&requestStatus="SUBMITTED"&requestStatus="PROCESSED"]**

moduleTag=<lista di module tag> per restringere il risultato alle sole richieste del/i modulo/i aventi tag specificato [...] **&moduleTag="tag_1"&moduleTag="tag_2"]**

swfOptionKey=<lista di option keys> per restringere il risultato alle sole richieste aventi lo/gli stato/i del Simple Workflow (SWF) impostato all'indice corrispondente ad uno dei possibili valori del dominio definito dall'utente [...] **&swfOptionKey="3"&swfOptionKey="22"]**

swfOptionMode=può avere valore "IN" o "NOT IN". Se utilizzato "IN" vengono restituite tutte le request che hanno gli "swfOptionKey" dichiarati. Se utilizzato "NOT IN" vengono restituite tutte le request che non hanno gli "swfOptionKey" dichiarati. Se non dichiarato di default viene utilizzato "IN"

Viene restituito l'elenco degli [id_request] per tutte le istanze avanti le clausole in almeno uno dei valori desiderati.

Verrà restituito errore se:

- viene specificato almeno un tag modulo che non esiste
- se viene specificata una swfOptionKey senza aver dichiarato uno ed un solo tag modulo

Verrà restituito un elenco vuoto con esito OK se:

- la combinazione delle clausole specificate non individua elementi nell'insieme di ricerca
- se swfOptionKey non esiste nel dominio definito dall'utente (non vi sarà controllo di consistenza tra lo stato di ricerca e l'eventuale stato SWF, se dichiarato)
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo swfOptionKey sia stato valorizzato



Se venissero utilizzate più clausole, la logica di estrazione sarà la seguente: **[requestStatus in (value1,..., valueN) [AND moduleTag in (value1,..., valueN) [AND swfOptionKey in (value1,..., valueN)]]]**

v1.0 (deprecated 4 removal): *https://[host]/eF/api/request/lookup/by-status*

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): *https://[host]/eF/services/api/request/lookup/by-status/v1*

http method: POST con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

form parameters:

username(*)=<username>

moduleTag(*)=<un module tag>

step(0..n)=PRONTE_DA_ELABORARE possibili valori: **[INOLTRATA (default), INOLTRATA_SWF_NONE, PRONTE_DA_ELABORARE, EVASA]**

INOLTRATA: o se non diversamente specificato, vengono restituite quelle richieste INOLTRATE (non evase) senza un stato di workflow associato.

PRONTE_DA_ELABORARE: Viene restituito l'elenco degli [id_request] per tutte le istanze aventi lo stato di workflow in meta stato **PRONTE_DA_ELABORARE**.

INOLTRATA_SWF_NONE: vengono restituite quelle richieste INOLTRATE (non evase) indipendentemente che abbiano o meno stato di workflow associato alla richiesta stessa.

EVASA: vengono restituite le richieste EVASE, indipendentemente che abbiano o meno uno stato di workflow



Evolutiva

Dalla versione 2.2.0 del core API è possibile inviare una lista di possibile **step** per i quali effettuare la ricerca: il risultato sarà l'unione (esclusi i duplicati) delle richieste ricercate.

Prestare attenzione al possibile numero elevato di corrispondenze restituite e, di conseguenza, agli effetti sulla performance in funzione del numero massimo di istanze ricercabili configurate per la piattaforma e il moltiplicatore step!

~~Lookup-by-date~~ (deprecato per rimozione futura, since v1_1: utilizzare Lookup-by-period)



Recupero delle istanze inoltrate

per ottenere l'elenco delle istanze di un modulo in base allo stato e al periodo di riferimento:

v1.1 (deprecated 4 removal): *https://[host]/eF/api/request/lookup/by-date*

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated 4 removal): *https://[host]/eF/api/request/lookup/by-date*

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): *https://[host]/ef/services/api/request/lookup/by-date/v01*

http method: POST con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

form parameters:

username(*)=<username>

moduleTag(*)=<un module tag>

step(0..n)=PRONTE_DA_ELABORARE possibili valori: [INOLTRATA (default), INOLTRATA_SWF_NONE, PRONTE_DA_ELABORARE, EVASA]

INOLTRATA: o se non diversamente specificato, vengono restituite quelle richieste INOLTRATE (non evase) senza un stato di workflow associato.

PRONTE_DA_ELABORARE: Viene restituito l'elenco degli [id_request] per tutte le istanze aventi lo stato di workflow in meta stato PRONTE_DA_ELABORARE.

INOLTRATA_SWF_NONE: vengono restituite quelle richieste INOLTRATE (non evase) indipendentemente che abbiano o meno stato di workflow associato alla richiesta stessa.

EVASA: vengono restituite le richieste EVASE, indipendentemente che abbiano o meno uno stato di workflow

dateFrom(*)=2019-01-01 in formato standard ISO 8601.

dateTo(*)=2019-03-30 in formato standard ISO 8601.

L'elenco delle istanze viene ulteriormente filtrato per il periodo di riferimento.



Evolutiva

Dalla versione 2.2.0 del core API è possibile inviare una lista di possibile **step** per i quali effettuare la ricerca: il risultato sarà l'unione (esclusi i duplicati) delle richieste ricercate.

Prestare attenzione al possibile numero elevato di corrispondenze restituite e, di conseguenza, agli effetti sulla performance in funzione del numero massimo di istanze ricercabili configurate per la piattaforma e il moltiplicatore step!

Lookup by-period



Recupero delle istanze inoltrate

DRAFT

PREVIEW

per ottenere l'elenco delle istanze di un modulo in base allo stato e al periodo di riferimento:

v1.2 (default): *https://[host]/eF/api/request/lookup/by-period*

- senza header,
- con header "accept:application/json",
- con header "accept:application/xml",
- con header "accept:application/it.elixforms.api.v1.2.0+json"

- con header "accept:application/it.elixforms.api.v1.2.0+xml" v1.1 (deprecated): *https://[host]/eF/api/request/lookup/by-period*
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

form parameters:

requestStatus=<lista di stati> possibili valori:[IN_PROGRESS, SUBMITTED, PROCESSED] per restringere il risultato alle sole richieste nel /i dato/i stato/i [...&requestStatus="SUBMITTED"&requestStatus="PROCESSED"]

moduleTag=<lista di module tag> per restringere il risultato alle sole richieste del/i modulo/i aventi tag specificato [...&moduleTag="tag_1" &moduleTag="tag_2"]

swfOptionKey=<lista di option keys> per restringere il risultato alle sole richieste aventi lo/gli stato/i del Simple Workflow (SWF) impostato all'indice corrispondente ad uno dei possibili valori del dominio definito dall'utente [...&swfOptionKey="3"&swfOptionKey="22"]

swfOptionMode=può avere valore "IN" o "NOT IN". Se utilizzato "IN" vengono restituite tutte le request che hanno gli "swfOptionKey" dichiarati. Se utilizzato "NOT IN" vengono restituite tutte le request che non hanno gli "swfOptionKey" dichiarati. Se non dichiarato di default viene utilizzato "IN"

dateFrom(*)=2024-01-01 in formato standard ISO 8601.

dateTo(*)=2024-03-30 in formato standard ISO 8601.

Viene restituito l'elenco degli [id_request] per tutte le istanze avanti le clausole in almeno uno dei valori desiderati per il periodo di riferimento.

Verrà restituito errore se:

- viene specificato almeno un tag modulo che non esiste
- se viene specificata una swfOptionKey senza aver dichiarato uno e un solo tag modulo

Verrà restituito un elenco vuoto con esito OK se:

- la combinazione delle clausole specificate non individua elementi nell'insieme di ricerca
- se swfOptionKey non esiste nel dominio definito dall'utente (non vi sarà controllo di consistenza tra lo stato di ricerca e l'eventuale stato SWF, se dichiarato)
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo swfOptionKey sia stato valorizzato



Se venissero utilizzate più clausole, la logica di estrazione sarà la seguente: [requestStatus in (value1,..., valueN) [AND moduleTag in (value1,..., valueN) [AND swfOptionKey in (value1,..., valueN)]]]

Lookup by-user



Recupero delle istanze inoltrate

DRAFT

PREVIEW

per ottenere l'elenco delle istanze di un modulo in base allo stato:

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

query parameters:

requestUser(*)=<un identificativo utente (isi_users.utente aka username)> a cui appartengono le richieste da ricercare

requestStatus=<lista di stati> possibili valori:[IN_PROGRESS, SUBMITTED, PROCESSED] per restringere il risultato alle sole richieste nel /i dato/i stato/i [...&requestStatus="SUBMITTED"&requestStatus="PROCESSED"]

moduleTag=<lista di module tag> per restringere il risultato alle sole richieste del/i modulo/i aventi tag specificato [...&moduleTag="tag_1" &moduleTag="tag_2"]

swfOptionKey=<lista di option keys> per restringere il risultato alle sole richieste aventi lo/gli stato/i del Simple Workflow (SWF) impostato all'indice corrispondente ad uno dei possibili valori del dominio definito dall'utente [...&swfOptionKey="3"&swfOptionKey="22"]

swfOptionMode=può avere valore "IN" o "NOT IN". Se utilizzato "IN" vengono restituite tutte le request che hanno gli "swfOptionKey" dichiarati. Se utilizzato "NOT IN" vengono restituite tutte le request che non hanno gli "swfOptionKey" dichiarati. Se non dichiarato di default viene utilizzato "IN"

Viene restituito l'elenco degli [id_request] per tutte le istanze avanti le clausole in almeno uno dei valori desiderati.

Verrà restituito errore se:

- viene specificato almeno un tag modulo che non esiste
- se viene specificata una **swfOptionKey** senza aver dichiarato almeno un tag modulo

Verrà restituito un elenco vuoto con esito OK se:

- la combinazione delle clausole specificate non individua elementi nell'insieme di ricerca
- se **swfOptionKey** non esiste nel dominio definito dall'utente (non vi sarà controllo di consistenza tra lo stato di ricerca e l'eventuale stato SWF, se dichiarato)
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo **swfOptionKey** sia stato valorizzato

v1.0 (deprecated 4 removal): *https://[host]/eF/api/request/lookup/by-user*

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): *https://[host]/eF/services/api/request/lookup/by-user/v1*

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

query parameters:

username(*)=<username> username autenticato

requestUser(*)=<un identificativo utente (isi_users.utente aka username)> a cui appartengono le richieste da ricercare

requestStatus=<lista di stati> possibili valori:[IN_CORSO, INOLTATA, EVASA] per restringere il risultato alle sole richieste nel/i dato/i stato/i [...&requestStatus="INOLTATA"&requestStatus="EVASA"]

moduleTag=<lista di module tag> per restringere il risultato alle sole richieste del/i modulo/i aventi tag specificato [...&moduleTag="tag_1" &moduleTag="tag_2"]

swfOptionKey=<lista di option keys> per restringere il risultato alle sole richieste aventi lo/gli stato/i del Simple Workflow (SWF) impostato all'indice corrispondente ad uno dei possibili valori del dominio definito dall'utente [...&swfOptionKey="3"&swfOptionKey="22"]

Viene restituito l'elenco degli [id_request] per tutte le istanze avanti le clausole in almeno uno dei valori desiderati.

Verrà restituito errore se:

- viene specificato almeno un tag modulo che non esiste
- se viene specificata una **swfOptionKey** senza aver dichiarato almeno un tag modulo

Verrà restituito un elenco vuoto con esito OK se:

- la combinazione delle clausole specificate non individua elementi nell'insieme di ricerca
- se **swfOptionKey** non esiste nel dominio definito dall'utente (non vi sarà controllo di consistenza tra lo stato di ricerca e l'eventuale stato SWF, se dichiarato)
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo **swfOptionKey** sia stato valorizzato



Se venissero utilizzate più clausole, la logica di estrazione sarà la seguente: [requestStatus in (value1,..., valueN) [AND moduleTag in (value1,..., valueN) [AND swfOptionKey in (value1,..., valueN)]]]



Recupero delle istanze inoltrate

DRAFT

PREVIEW

per ottenere l'elenco delle istanze di un modulo in base allo stato e al periodo di riferimento:

v1.2 (default): *https://[host]/eF/api/request/lookup/by-user-and-period*

- senza header,
- con header "accept:application/json",
- con header "accept:application/xml",
- con header "accept:application/it.elixforms.api.v1.2.0+json"
- con header "accept:application/it.elixforms.api.v1.2.0+xml"

v1.1 (deprecated): *https://[host]/eF/api/request/lookup/by-user-and-period*

- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

form parameters:

requestUser(*)=<un identificativo utente (isi_users.utente aka username)> a cui appartengono le richieste da ricercare

requestStatus=<lista di stati> possibili valori: [IN_PROGRESS, SUBMITTED, PROCESSED] per restringere il risultato alle sole richieste nel /i dato/i stato/i [...&requestStatus="SUBMITTED"&requestStatus="PROCESSED"]

moduleTag=<lista di module tag> per restringere il risultato alle sole richieste del/i modulo/i aventi tag specificato [...&moduleTag="tag_1" &moduleTag="tag_2"]

swfOptionKey=<lista di option keys> per restringere il risultato alle sole richieste aventi lo/gli stato/i del Simple Workflow (SWF) impostato all'indice corrispondente ad uno dei possibili valori del dominio definito dall'utente [...&swfOptionKey="3"&swfOptionKey="22"]

swfOptionMode=può avere valore "IN" o "NOT IN". Se utilizzato "IN" vengono restituite tutte le request che hanno gli "swfOptionKey" dichiarati. Se utilizzato "NOT IN" vengono restituite tutte le request che non hanno gli "swfOptionKey" dichiarati. Se non dichiarato di default viene utilizzato "IN"

dateFrom(*)=2019-01-01 in formato standard ISO 8601.

dateTo(*)=2019-03-30 in formato standard ISO 8601.

Viene restituito l'elenco degli [id_request] per tutte le istanze avanti le clausole in almeno uno dei valori desiderati per il periodo di riferimento.

Verrà restituito errore se:

- viene specificato almeno un tag modulo che non esiste
- se viene specificata una **swfOptionKey** senza aver dichiarato almeno un tag modulo

Verrà restituito un elenco vuoto con esito OK se:

- la combinazione delle clausole specificate non individua elementi nell'insieme di ricerca
- se **swfOptionKey** non esiste nel dominio definito dall'utente (non vi sarà controllo di consistenza tra lo stato di ricerca e l'eventuale stato SWF, se dichiarato)
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo swfOptionKey sia stato valorizzato



Se venissero utilizzate più clausole, la logica di estrazione sarà la seguente: [requestStatus in (value1,..., valueN) [AND moduleTag in (value1,..., valueN) [AND swfOptionKey in (value1,..., valueN)]]]

v1.0 (deprecated): *https://[host]/eF/api/request/lookup/by-user-and-period*

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): *https://[host]/ef/services/api/request/lookup/by-user-and-period/v01*

http method: POST con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

form parameters:

username(*)=<username>

requestUser(*)=<un identificativo utente (isi_users.utente aka username)> a cui appartengono le richieste da ricercare

requestStatus=<lista di stati> possibili valori: [IN_CORSO, INOLTTRATA, EVASA] per restringere il risultato alle sole richieste nel/i dato/i stato/i [...&requestStatus="INOLTTRATA"&requestStatus="EVASA"]

moduleTag=<lista di module tag> per restringere il risultato alle sole richieste del/i modulo/i aventi tag specificato [...&moduleTag="tag_1"&moduleTag="tag_2"]

swfOptionKey=<lista di option keys> per restringere il risultato alle sole richieste aventi lo/gli stato/i del Simple Workflow (SWF) impostato all'indice corrispondente ad uno dei possibili valori del dominio definito dall'utente [...&swfOptionKey="3"&swfOptionKey="22"]

dateFrom(*)=2019-01-01 in formato standard ISO 8601.

dateTo(*)=2019-03-30 in formato standard ISO 8601.

Viene restituito l'elenco degli [id_request] per tutte le istanze avanti le clausole in almeno uno dei valori desiderati per il periodo di riferimento.

Verrà restituito errore se:

- viene specificato almeno un tag modulo che non esiste
- se viene specificata una swfOptionKey senza aver dichiarato almeno un tag modulo

Verrà restituito un elenco vuoto con esito OK se:

- la combinazione delle clausole specificate non individua elementi nell'insieme di ricerca
- se swfOptionKey non esiste nel dominio definito dall'utente (non vi sarà controllo di consistenza tra lo stato di ricerca e l'eventuale stato SWF, se dichiarato)
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo swfOptionKey sia stato valorizzato



Se venissero utilizzate più clausole, la logica di estrazione sarà la seguente: [requestStatus in (value1,..., valueN) [AND moduleTag in (value1,..., valueN) [AND swfOptionKey in (value1,..., valueN)]]]

Get request identifier



Get

per ottenere l'identificativo della request inoltrata a partire dal qrCode specificato; è anche metodo di utilità da orchestrare con altri servizi per ottenere ad esempio la pratica completa con la [getRequest](#)

v1.1 (default): *https://[host]/eF/api/request/get-id*

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): *https://[host]/eF/api/request/get-id*

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): *https://[host]/eF/services/api/request/get-id/v1*

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

pathInfo parameter:

--

query parameters:

username(*)=<username>

qrCode(*)=codice <qrCode> interessato

Get request



Get

per ottenere il dettaglio sintetico della request con id specificato

v1.1 (default): **https://[host]/eF/api/request/{request_id}/get**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/get**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/{request_id}/get/v1**

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

pathInfo parameter: request_id(*) (id dell'istanza eF)

query parameters:

username(*)=<username>

recordOrder(?)=<NONE(default), NEWEST_FIRST, NEWEST_LAST> (case sensitive)

La piattaforma eF* restituisce l'intero fascicolo dell'istanza in uno dei formati standard - JSON o XML – secondo un pattern uniformato per l'intera piattaforma.

Tutti i dati del documento JSON e XML sono normalizzati.

Gli **allegati** sono resi disponibili in tag contenenti l'URL di riferimento per un facile download (*Vedi esempio allegato) o via servizio [#getDocumentRequest](#).



Set Export Tag

TAG per mapping: Il creatore del modulo può utilizzare l'attributo **setTagExport** per marcare con uno o più tags separati (da spazio, virgola o punto e virgola) i CAMPI utili alla mappatura inversa.

Con l'impostazione di TAG è preservata la possibilità di utilizzare differenti moduli (o edizione clonate di questi) preservando il mapping, anche se campi e struttura del modulo saranno differenti.

TAG MODULO: Il creatore del modulo potrà inserire per ogni modulo una scheda libera di parametri per modulo che, compilati, verranno estratti in ogni istanza in una sezione dedicata.

Get request by QRCode



Get

per ottenere il dettaglio sintetico della request a partire dal qrCode specificato

v1.1 (default): **https://[host]/eF/api/request/qrcode/{qrCode}/get**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/qrcode/{qrCode}/get**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/qrcode/{qrCode}/get/v1**

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

pathInfo parameter: qrCode(*) (codice qrCode)

query parameters:

username(*)=<username>

recordOrder(?)=<NONE(default), NEWEST_FIRST, NEWEST_LAST> (case sensitive)

La piattaforma eF* restituisce l'intero fascicolo dell'istanza in uno dei formati standard - JSON o XML – secondo un pattern uniformato per l'intera piattaforma.

Tutti i dati del documento JSON e XML sono normalizzati.

Gli **allegati** sono resi disponibili in tag contenenti l'URL di riferimento per un facile download (*Vedi esempio allegato) o via servizio [#getDocumentRequest](#).



Set Export Tag

TAG per mapping: Il creatore del modulo può utilizzare l'attributo **setTagExport** per marcare con uno o più tags separati (da spazio, virgola o punto e virgola) i CAMPI utili alla mappatura inversa.

Con l'impostazione di TAG è preservata la possibilità di utilizzare differenti moduli (o edizione clonate di questi) preservando il mapping, anche se campi e struttura del modulo saranno differenti.

TAG MODULO: Il creatore del modulo potrà inserire per ogni modulo una scheda libera di parametri per modulo che, compilati, verranno estratti in ogni istanza in una sezione dedicata.

Get Document request



Get

per ottenere la risorsa fisica di una colonna di tipo binary per una request con id specificato

v1.1 (default): **https://[host]/eF/api/request/{request_id}/get/document**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/get/document**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/{request_id}/get/document/v1**

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

pathInfo parameter: request_id(*) (id dell'istanza eF)

query parameters:

username(*)=<username>

storageId(*)=<id di stoccaggio della risorsa> (binary column "columnValue" o "key" dei binary values)

upload-timestamp(*)=<timestamp di upload della risorsa> ("upload-timestamp" della property columnValueList/columnValue della binary column)

content-type(?)=<media type della risorsa se disponibile> ("content-type" della property columnValueList/columnValue della binary column)

Get User Infos of existing request



Get

per ottenere le informazioni relative all'utenza sottesa ad una richiesta esistente è sufficiente, previo autenticazione e rilascio del token autorizzativo, eseguire la richiesta fornendo il numero identificativo

v1.1 (default): **https://[host]/eF/api/request/{request_id}/user-info/get**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/user-info/get**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/{request_id}/user-info/get/v1**

http method: GET con authToken header ("Authorization: Bearer <token>")

content-type: application/x-www-form-urlencoded

pathInfo parameter: request_id(*) (id dell'istanza eF)

query parameters:

username(*)=<username>

Inserimento/verifica di codici/informazioni applicative nel modulo

Per poter inter-operare correttamente è necessario che il sistema applicativo finale riceva dalle istanze eF dati corretti e congruenti con il proprio specifico dominio applicativo.

eF implementerà dei "comportamenti" standard attraverso plugin personalizzati di interazione con il sistema applicativo esterno.

Le informazioni recuperate e/o validate saranno, quindi, contenute nel fascicolo dell'istanza, consentendone in fase di recupero, ad avvenuta compilazione, la corretta gestione da parte del sistema esterno. (Vedi punto 1)

A questo scopo il sistema applicativo terzo deve mettere a disposizione ed esporre eventuali servizi rest che consentano:

- di **recuperare** le informazioni necessarie e corrette da inserire nel modulo (es. codici e decodifiche)
- di **recuperare** le informazioni eventualmente necessarie a **verificare** la congruità del modulo (Es. presenza in white list o non presenza in black list, ecc.).

Il modello eF prevede, infatti, due tipi di interazione:

1. **Visualizzazione:** per il recupero d'informazioni da mostrare all'utente e da inserire nel modulo. I plugin vengono attivati PRIMA della visualizzazione dello step.
 Esempio: i codici di plessi scolastici per le selezioni della scuola, i codici tariffe del sistema tributi, gli identificativi anagrafici dall'albo fornitori, ecc.
 Il modello consente interazioni sia semplici, tipo liste codice-descrizione, sia ricerche complesse che restituiscono un set di dati articolato (Es. stradario, anagrafe residenti, anagrafe immobili, tributi, ecc.)



Le chiamate del sistema esterno potranno richiedere una o più chiavi di ricerca e restituire i dati utili alla compilazione e al riuso in ingresso (Vedi punto 1)

2. **Validazione:** per effettuare controlli sui dati inseriti. È eseguito in fase di CONVALIDA del singolo step.



*Le chiamate del sistema esterno potranno richiedere una o più informazioni in input prese dinamicamente dal modulo e dovranno restituire: **TRUE o FALSE in base all'esito del controllo** ed una descrizione di errore in lingua ITA \ed opzionalmente una in lingua INGLESE\ in caso negativo*

Comunicazioni di cambio stato e/o richieste di integrazione attraverso la console utente eF

La piattaforma eF mette a disposizione anche un'interfaccia per l'utenza che inoltra le istanze (MyPage), attraverso la quale può riprendere le compilazioni in corso, effettuare richieste di supporto, integrare a richiesta l'istanza, ricevere comunicazioni informali e formali inclusive di autorizzazioni digitali. Attraverso questa interfaccia può essere anche visualizzato lo stato della propria pratica grazie ad un flag di stato.



Acquisizione delle istanze elaborate da processo esterno

per confermare che l'istanza è stata ricevuta e non andrà più trasmessa:

v1.1 (default): **https://[host]/eF/api/request/{request_id}/set-acquired**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/set-acquired**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/{request_id}/set-acquired/v1**

http method: **POST** con authToken header ("Authorization: Bearer <token>")

pathInfo parameter: request_id(*) (id dell'istanza eF)

content-type: application/x-www-form-urlencoded

form parameters:

username(*)=<username> username autenticato

outcome(*)=<esito del servizio> **possibili valori:**[OK, KO]

resultCode(*)=<codice esito processo esterno> **possibili valori:** [VAL("Validata"), IMP("Importata"), IMPW("Impostata con warning"), KO("Rifiutata")]

resultDescription=<descrizione esito processo esterno>

resultReceiptId(*)=<id pratica processo esterno di riferimento>

resultTimestamp=<timestamp di riferimento dell'elaborazione pratica dal processo esterno> in formato standard ISO 8601. Se non impostata viene inizializzata con la data/ora server corrente

properties=<mappa chiave,valore/i aggiuntivo/i in formato json> (url-encoding richiesto in UTF-8)

Viene ritornato l'esito dell'operazione di storicizzazione si **elixForms@**.

La ricezione di "ok" o "ko" provocherà una decisione nel workflow semplice (analogo a quello di **elixFlow@**) in base a quanto disposto in elixForms per il modulo.

- Esempio="OK","VAL","..." imposta lo stato di workflow semplice corrispondente imposta lo step ad "evasa" notifica email
- Esempio="KO","KO","..." imposta lo stato di workflow semplice corrispondente non cambia lo stato dello step



Il parametro lista di properties (chiave, valore) viene introdotto per ottenere una struttura più generica e flessibile per il futuro.



La chiamata provoca l'assunzione che l'istanza in oggetto non verrà più trasmessa, indipendentemente dall'outcome.

Inoltre, come successiva integrazione (@since 1.5.0), si è deciso di "dare la libertà" al chiamante di impostare lo stato principale e lo stato dell'eventuale **"S impleWorkflow"** (SWF) collegato al modulo - con dato TAG - di quelle richieste che **NON** si trovano "In corso" di compilazione, ossia che sono state **comp** letate e inoltrate.



Cambio stato delle istanze elaborate da processo esterno

DRAFT

PREVIEW

per confermare il cambio stato dell'istanza e/o lo stato di avanzamento dell'SWF:

v1.1 (default): **https://[host]/eF/api/request/{request_id}/set-status**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/set-status**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/{requestId}/set-status/v1**

http method: POST con authToken header ("Authorization: Bearer <token>")

pathInfo parameter: requestId(*) (id dell'istanza eF di cui si vuole impostare lo stato/i)

content-type: application/x-www-form-urlencoded

forms parameters:

username(*)=<username> username autenticato al servizio associato al token corrente

requestStatus=<nuovo stato della request> possibili valori: [SUBMITTED, PROCESSED]

requestStatusDescription=<string> descrizione "human readable" della motivazione del passaggio di stato

swfOptionKey=<indice del valore associato al SWF che si desidera impostare> possibili valori: [dominio definito dall'utente - nessun controllo di consistenza!]

properties=<mappa chiave,valore/i aggiuntivo/i in formato json> (url-encoding richiesto in UTF-8) [la gestione dei parametri esiste per dare versatilità al servizio, ma il suo utilizzo formale va concordato con l'utilizzatore]

- Nel caso la request NON risulti completata e inoltrata, l'operazione solleverà un errore.
- Non vi sarà controllo di consistenza tra lo stato principale che si vuole impostare nella richiesta e l'eventuale stato SWF, se dichiarato
- Se il servizio dovesse essere invocato senza requestStatus parameter e senza swfOptionKey, l'esito sarà OK con attributo **completed = false**
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta e l'attributo swfOptionKey sia stato valorizzato, verrà sollevata una eccezione con esito KO

Viene ritornato l'identificativo dell'operazione di storicizzazione in **elixForms®** in caso di **esito positivo (OK, completed)**.



Il parametro lista di properties (chiave, valore) viene introdotto per ottenere una struttura più generica e flessibile per necessità future individuali, senza dover rivedere la struttura del servizio.



La chiamata provoca l'assunzione che l'istanza in oggetto non sia in corso di compilazione: per questo motivo si permettono solo 2 valori per impostare lo stato attuale della richiesta.

Con le successive integrazioni (@since 1.6.2), si è deciso di "dare la libertà" al chiamante di impostare autonomamente tutte le colonne di un dato **"Simple Workflow"** (SWF) collegato al modulo - con dato TAG - di quelle richieste che **NON** si trovano "In corso" di compilazione, ossia che sono state **completate e inoltrate**.



Cambio valori SWF collegato al moduloTag della request

DRAFT

PREVIEW

per confermare il cambio stato dell'istanza e/o lo stato di avanzamento dell'SWF:

v1.1 (default): **https://[host]/eF/api/request/{request_id}/swf/{columns: .+}**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/swf/{columns: .+}**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): **https://[host]/eF/services/api/request/{requestId}/swf/{columns: .+}/v1**

http method: POST con authToken header ("Authorization: Bearer <token>")

headers:

x-api-username(*)=<username> mutuo alternativo al query param **username**, è lo username autenticato al servizio associato al token corrente

pathInfo parameter:

requestId(*) (id dell'istanza eF di cui si vuole impostare lo/gli stato/i)

columns(*) (più occorrenze di coppie **colTag=value**; come da specifica rest per parametri MATRIX, ove colTag è il tag della colonna dello schema SWF da impostare e il value è il suo relativo nuovo valore vedi esempio a corollario documento)

content-type: application/x-www-form-urlencoded

query parameters:

username(*)=<username> username autenticato al servizio associato al token corrente

- Nel caso la request NON risulti completata e inoltrata, l'operazione solleverà un errore.
- Non vi sarà controllo di consistenza tra i valori che si desidera impostare e l'eventuale stato SWF corrente, se esistente; se non esistente verrà creata una nuova scheda SWF in funzione dell'id del modulo sotteso
- Se il servizio venisse invocato **prima** che un processo di SWF sia stato associato al modulo della richiesta, verrà sollevata una eccezione con esito KO

Viene ritornato l'identificativo dell'operazione di storicizzazione in **elixForms®**



La chiamata provoca l'assunzione che l'istanza in oggetto non sia in corso di compilazione.

ATTENZIONE

NON è previsto al momento la possibilità di effettuare l'upload di files nella scheda SWF

FUNZIONE RIAPERTURA DOMANDA 2.0

@since 1.6.1, si è deciso di esporre al chiamante la possibilità di riaprire una richiesta precedentemente evasa



Riapertura delle richieste evase

DRAFT

PREVIEW

per riaprire le domande evase e riportare lo stato "evasa" e "confermata" a no, mantenendo lo stato a inoltrata:

v1.1 (default): **https://[host]/eF/api/request/{request_id}/reopen**

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): **https://[host]/eF/api/request/{request_id}/reopen**

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): ***https://[host]/eF/services/api/request/reopen/v1***

http method: POST con authToken header ("Authorization: Bearer <token>")

pathInfo parameter: nessuno

content-type: application/json

query parameters:

username(*)=<username> username autenticato al servizio associato al token corrente

post body:

richiede la "richiesta riapertura" in formato json così strutturata

```
{
  "requestId": number, <(*) id della request>
  "clientId": number, <id dell'utenza "reale", (non quella sistemistica) che ha generato la riapertura, i.e. utenza loggata>
  "reopeningReason": string, <motivazione della riapertura>
  "reopeningNotificationDate": date, <data in formato ISO: "2021-05-18T00:00:00">
  "reopenLogId": number, <se esiste un log di storicizzazione da aggiornare e si conosce il generic id della scheda; se non specificato
  logga l'operazione in nuova scheda>
  "reopeningAttachmentFileName": string, <nome del file in attach, i.e. miofile.pdf>
  "reopeningAttachmentMimeType": string, <mimetype del file in attach, i.e. "application/pdf">
  "reopeningAttachment": byte[] <bytes del file in attach>
}
```

Viene ritornato l'identificativo dell'operazione di storicizzazione in **elixForms®** in caso di **esito positivo (OK, completed)** e la struttura della request post aggiornamenti come da [get #get_request](#).



La chiamata provoca l'assunzione che l'istanza in oggetto non sia in corso di compilazione.

FUNZIONE CLONAZIONE DI UNA DOMANDA 2.0

@since 2.4.1, viene esposta al chiamante la possibilità di clonare una richiesta in stato inoltrata



Riapertura delle richieste evase

DRAFT

PREVIEW

per clonare le domande inoltrate :

v1.1 (default): [https://\[host\]/eF/api/request/{request_id}/clone](https://[host]/eF/api/request/{request_id}/clone)

- senza header,
- con header "accept:application/json",
- con header "accept:application/it.elixforms.api.v1.1.0+json"
- con header "accept:application/it.elixforms.api.v1.1.0+xml"

v1.0 (deprecated): [https://\[host\]/eF/api/request/{request_id}/clone](https://[host]/eF/api/request/{request_id}/clone)

- con header "accept:application/it.elixforms.api.v1.0.0+json"
- con header "accept:application/it.elixforms.api.v1.0.0+xml"

v1.0 (deprecated 4 removal): [https://\[host\]/eF/services/api/request/{requestId}/clone/v1](https://[host]/eF/services/api/request/{requestId}/clone/v1)

http method: **POST** con authToken header ("Authorization: Bearer <token>") o pre shared token x-api-token

pathInfo parameter: requestId(*) (id dell'istanza eF sorgente che si desidera clonare)

headers:

x-api-key(*): <token> è il token autentificativo assegnato alla piattaforma e comunicato dal reparto tecnico

x-api-username: <username> dell'utenza server2server che si collega con dato token

content-type: application/json

query parameters:

nessuno

post body:

richiede la "richiesta riapertura" in formato json così strutturata

```
{
  "clientUsername": string, <(*) username dell'utente che richiede la clonazione>
  "cloningReason": string, <(*) motivazione della copia>
  "cloningDate": dateTime, <(*) data in formato ISO: "2021-05-18T00:00:00">
  "cloneLogId": number, <generic id dello storico di clonazione eventualmente da aggiornare; se non specificato viene automaticamente generato un nuovo log (Scheda "RW2 - Clone")>
}
```

Viene ritornata la struttura della request clonata in stile "get" [#get_request](#).



La chiamata provoca l'assunzione che l'istanza in oggetto non sia in corso di compilazione.

COMUNICAZIONI FORMALI

Invio comunicazione formale all'utente

Invio di comunicazione formale all'utente

Vengono richiesti messaggio da inviare e suo riferimento a tipo di documento:

https://[host]/eF/services/api/request/{request_id}/user-formal-communication

http method: **POST** con authToken header ("Authorization: Bearer <token>") o con token autentificativo OpenAPI ("x-api-key: <token>")

pathInfo parameter: request_id(*) (id dell'istanza eF)

content-type: application/json

accept: application/it.elixforms.api.v1+json

x-api-username: <username> dell'utente abilitato alla "consume" del servizio.

body:

Payload

```
{
  "message": "test",
  "documentType": "Autorizzazione",
  "documentNumber": "123123_AZ",
  "documentDate": "2023-02-27T17:08:00",
  "attachments": [
    {
      "attachFile": "<Base 64 encoded string>",
      "attachMimeType": "application/pdf",
      "attachFileName": "test.pdf"
    },
    {
      "attachFile": "<Base 64 encoded string>",
      "attachMimeType": "image/jpeg",
      "attachFileName": "test1.jpeg"
    }
  ],
  "senderUsername": "username"
}
```

message(*) = (Stringa) <messaggio da inviare all'utente finale, intestatario della request>

documentType(*) = (Stringa) <tipo di documento> La corrispondente funzionalità in elixForms prevede un elenco preso da una property di piattaforma (globalparam.formalcommunications.available.list, i cui valori, di default possono essere i seguenti: Appuntamento, Autorizzazione, Comunicazione, Esito, Licenza, Permesso, Richiesta di integrazione

documentNumber = (Stringa) <numero del documento>

documentDate = (Data in ISO format, come da specifica jax-rs) <data del documento>

senderUsername = (Stringa) <username> di un utente presente sul DB elixforms (isi_users) da indicare come mittente; nel caso non sia specificato, verrà forzato con lo username in header "x-api-username" titolare di accesso al servizio, ma anch'esso deve essere presente sul DB elixforms (isi_users).

attachments = <Array di Attachment> [da 0 a 5 ★]

Attachment

attachFile()** = <file> rappresentato da una stringa in formato Base64, encoding del byte array del file originale in UTF-8

attachFileName()** = <nome del file> da assegnare al file in upload

attachMimeType = <mime type> del documento allegato. Se vuoto viene impostato come application/octet-stream

attachSize = <dimensione del file> dichiarata. Se non specificato, viene calcolato in funzione del file ricevuto

Sebbene attachFileName, attachMimeType siano facilmente pre-impostabili di default o per deduzione programmatica, al momento è necessario esplicitarli

(*) Campi obbligatori

() Se viene inserito il nodo, devono essere valorizzati**



Come primo metodo che nasce in fase evolutiva delle API-3, si elimina il pathinfo "/v1" e si sposta tale decodifica in header Accept con notazione standard secondo RFC2616



★ Il servizio esposto tecnicamente non ha impedimenti tecnici al numero massimo di attachments possibili. Tuttavia, più sono numerosi e pesanti, più sarà il tempo necessario per l'upload e di conseguenza per l'esecuzione del servizio stesso.

Inoltre, questa funzionalità, disponibile anche in Console Gestione Istanze in elixForms, **prevede un massimo di n. 5 allegati**. Pertanto a limite raggiunto anche il servizio risponderà con un errore.



(*) parametro obbligatorio

FUNZIONE CALENDAR 2.0

Si introduce la possibilità di invocare le funzionalità server-to-server della recente funzionalità Calendar 2.0 via Elixforms© API 2.0.

Questa modalità sfrutta al meglio le nuove funzionalità di login/logout API 2.0 con token autentificativo; unifica, inoltre, l'usabilità dei servizi esposti alla clientela.

I servizi, che in questo momento saranno esposti, soddisfano le recenti richieste di business di ottenere gli appuntamenti per un certo tag e segnalare l'acquisizione degli stessi come conferma di ricezione.

Abbiamo quindi 2 soluzioni:

Calendar 2: get appointments



get (calendar 2.0 appointments)

per ottenere l'elenco degli appuntamenti per un certo **tag** :

https://[host]/eF/services/api/calendar/appointments/get/v1

http method: GET con authToken header ("Authorization: Bearer <token>")

get parameters: username&tag

username(*)=<username>

tag(*)=<un tag>

Restituisce un elenco di appuntamenti disponibili per il tag specificato.

Calendar 2: set-acquired appointment



set-acquired (calendar 2.0 appointment)

https://[host]/eF/services/api/calendar/appointments/set-acquired/v1

http method: PUT con authToken header ("Authorization: Bearer <token>")

put parameters: username&tag

username(*)=<username>

tag(*)=<un tag>

Imposta per gli appuntamenti corrispondenti ad un certo tag specificato l'acquisizione dal sistema esterno.

FUNZIONE EFTL© - Elixforms© Tag Language

Si introduce (@since 1.7.2) la possibilità di invocare la processazione di documenti e/o files, riconducibili al formato testo quali il noto linguaggio html, tramite il motore **EFTL©** (Elixforms© Tag Language) via **Elixforms© API 2.0** su piattaforma micro-services.

I servizi, che in questo momento saranno esposti, soddisfano le recenti richieste di **interazione** tra un **contesto di business**, specifico per ogni cliente (detto "user context"), il **sistema di risoluzione di informazioni** conosciuto come **TagEngine** di Elixforms©, via costrutti già noti nell'utilizzo dei moduli quali [TAG]...[/TAG], ed un **sistema logico potente e agnostico**, denominato **EFTL©** (Elixforms© Tag Language) per tradurre un semplice ed, al tempo stesso, essenziale linguaggio a tag, identificati dai caratteri quadra, "[" e "]" (es. [MIO_TAG attr1="value 1"... attrN="value n"] ... [/MIO_TAG]), in applicazioni logiche decisionali atte a "compilare" a runtime, ossia durante la processazione e produzione di documentazione, il contesto utente completandolo con informazioni proveniente dalla modulistica Elixforms©.

Il linguaggio EFTL© è utilizzabile direttamente dai clienti di tipo amministratore Elixforms© di livello intermedio: è studiato in modo tale da poter essere scritto anche da gestori di modelli o procedimenti senza particolari conoscenze informatiche.

Il principio di base su cui si fonda, infatti, EFTL© è la sintassi dei tags [TAG]...[/TAG], già nota ad un amministratore Elixforms©, ed un modello di scrittura concettualmente simile a quello delle formule di fogli di calcolo quali, MS Excel, OSX Pages o LibreOffice, che sono molto conosciuti nel mercato business in generale. Nel presente documento vedremo, infatti, come invocare il servizio, mentre rimandiamo per la sintassi e i costrutti disponibili al seguente articolo specifico:

Essendo un servizio esposto, significa che può essere invocato da qualsivoglia contesto utente con accesso alle API 2.0 di Elixforms© e avente questa estensione attiva. Si ricora, inoltre, che questa modalità sfrutta al meglio le nuove funzionalità di login/logout API 2.0 con token autentificativo.

EFTL©: process document

Il servizio di processazione prevede l'invio di un documento, Base64 encoded, e della processazione dei soli blocchi EFTL©: ogni blocco EFTL sarà quindi sostituito con il risultato della compilazione e valutazione di ognuno di essi e quindi, di fatto, sarà accodato, as is, al resto del documento non processato ottenendo così il risultato finale desiderato.



Cambio valori SWF collegato al moduloTag della request

DRAFT

PREVIEW

per processare un documento contenente blocchi di linguaggio EFTL:

https://[/host]/eF/services/api/eftl/proces/v1

http method: POST **con** authToken header ("Authorization: Bearer <token>")

headers:

x-api-key(*):<token> è il token autentificativo assegnato alla piattaforma e comunicato dal reparto tecnico

x-ef-request-id:<id> è l'id della request che sottende la processazione del documento

content-type: application/json

post body:

```
{
  "userDocument": "PCFET0NUWVBFIgh0bWw+CjxodG1sPgo8aGVhZ.....3A+CjwvYm9keT4KPC9odG1sPg==",
  "userContext": {
    "paramName1": "paramValue1"
    ...
    "paramNameN": {
      ...
    }
  }
}
```

body di esempio (attenzione ai diversi tipi in userContext):

body sample

```
{
  "userDocument": "PCFET0NUwVBFiGh0bWw+...jwvYm9keT4KPC9odG1sPg==",
  "userContext": {
    "lang": "IT",
    "docs": [
      {
        "name": "doc1",
        "quantity": 3,
        "price": 123.3
      },
      {
        "name": "doc2",
        "quantity": 5,
        "price": 213.3
      },
      {
        "name": "doc3",
        "quantity": 10,
        "price": 321.3
      }
    ],
    "docsAsEntities": [
      {
        "eftlEntityType": "com.anthesi.elixforms.api.eftl.model.mock.EftlDoc",
        "eftlEntity": {
          "name": "doc1",
          "quantity": 3,
          "price": 123.3
        }
      },
      {
        "eftlEntityType": "com.anthesi.elixforms.api.eftl.model.mock.EftlDoc",
        "eftlEntity": {
          "name": "doc2",
          "quantity": 5,
          "price": 213.3
        }
      },
      {
        "eftlEntityType": "com.anthesi.elixforms.api.eftl.model.mock.EftlDoc",
        "eftlEntity": {
          "name": "doc3",
          "quantity": 10,
          "price": 321.3
        }
      }
    ],
    "singleEntity": {
      "eftlEntityType": "com.anthesi.elixforms.api.eftl.model.mock.EftlDoc",
      "eftlEntity": {
        "name": "doc1",
        "quantity": 3,
        "price": 123.3
      }
    },
    "simpleList": [
      "s1",
      "s2",
      "s3"
    ],
    "docsSize": 3
  }
}
```

Come si vede lo user context può contenere qualunque valore:

- uno scalare
- un'oggetto
- un array di oggetti
- un array di scalari
- il tipo EFTL contenente il tipo di oggetto da istanziare (eftlEntityType) e la sua serializzazione da applicare all'istanza (eftlEntity)
- oggetti nested dei precedenti

GESTIONE ERRORI

Nei controller dei servizi jax-rs vengono intercettate le exception di tipo **ElixFormsException**: se l'exception è di tipo **warning** o **error**, viene restituito come sempre un **WebResult** la cui entity è **null** completata con i result codes, outcome e status corrispondenti alla tipologia di eccezione.

Ed es:

```
{ "value": {
  "code": "WARNING",
  "description": "Nessuna pratica trovata per l'id 6885",
  "complete": true,
  "globalStatus": "WARNING",
  "uuid": "251bab06-0b59-4fbb-b748-4c65bdbc65ec",
  "requestSize": 0
}}
```

In caso di **eccezione generica** non gestita, invece, viene restituita una risposta **200** e result codes di tipo **ERROR**.



Generi Throwable (ossia i rimanenti Error types come ad es OutOfMemory) NON vengono intercettati e rilanciati al container perché vietato dalla specifica.

Rimangono casi che sarebbero NON gestibili da codice perché rispettanti la [specifica jax-rs](https://javaee.github.io/tutorial/jaxrs002.html) del tipo:

jax-rs 2.1 - Citazione della specifica...

If the URI path template **variable cannot be cast** to the specified type, the JAX-RS runtime returns an HTTP **400 ("Bad Request")** error to the client. If the **@PathParam** annotation **cannot be cast to the specified type**, the JAX-RS runtime returns an HTTP **404 ("Not Found")** error to the client. etc...

Altro caso simile è internal server error 500 in caso di **chiamate senza json body** per resources che "consumano" media types *application/json* ...

Tuttavia, ANCHE per i casi sopra citati, sono stati definiti degli **ExceptionMappers** che intercettano le eccezioni specifiche restituendo, rispettivamente, risultati del tipo:

- **ElixFormsException** (nel caso non sia stato correttamente intercettato nei controllers) simile al precedente esempio
- **ParamException** (un parametro qualunque sia il tipo):

```
{ "value": {
  "code": "ERROR",
  "description": "Request parameter not valid",
  "complete": false,
  "globalStatus": "ERROR",
  "resultNumber": -1,
  "serviceInformationList": { "serviceInformation": [ {
    "complete": false,
    "serviceVersion": "V1",
    "statusDetailList": { "statusDetail": [ {
      "code": "400",
      "description": "[Bad Request] - java.lang.NumberFormatException: For input string: \"688a\"",
      "detailStatus": "ERROR"
    } ]
    }
  } ]
  }
},
"uuid": "0c36536f-faee-41c4-94d9-8508e2e07632"
}
```



Sostituisce il default di specifica che rilancia un HttpStatus 404 - NotFound o 400 - Bad request

- **JaxbException** o **ProcessingException** (problemi lettura body della request, ad es body json vuoto o non sintatticamente corretto):

```
{
  "value": {
    "code": "ERROR",
    "description": "Empty body or syntax error reading json request body",
    "complete": false,
    "globalStatus": "ERROR",
    "resultNumber": -1,
    "serviceInformationList": {
      "serviceInformation": [
        {
          "complete": false,
          "serviceVersion": "V1",
          "statusDetailList": {
            "statusDetail": [
              {
                "code": "400",
                "description": "[Bad Request] - Internal error: Invalid token=EOF at (line no=1, column no=0, offset=-1). Expected tokens are: [CURLYOPEN, SQUAREOPEN, STRING, NUMBER, TRUE, FALSE, NULL]",
                "detailStatus": "ERROR"
              }
            ]
          }
        }
      ]
    },
    "uuid": "1d975696-42fd-4672-9d0a-638fec76bdb0"
  }
}
```



Sostituisce il default di specifica che rilancia una HttpStatus 500 - InternalServerError

- **MultiException** (più eccezioni rilanciate contestualmente)

```
{
  "value": {
    "code": "ERROR",
    "description": "Bad Request",
    "complete": false,
    "globalStatus": "ERROR",
    "resultNumber": -1,
    "serviceInformationList": {
      "serviceInformation": [
        {
          "complete": false,
          "serviceVersion": "V1",
          "statusDetailList": {
            "statusDetail": [
              {
                "code": "400",
                "description": "[Bad Request] - The @FormParam is utilized when the content type of the request entity is not application/x-www-form-urlencoded",
                "detailStatus": "ERROR"
              },
              {
                "code": "400",
                "description": "[Bad Request] - The @FormParam is utilized when the content type of the request entity is not application/x-www-form-urlencoded",
                "detailStatus": "ERROR"
              },
              {
                "code": "400",
                "description": "[Bad Request] - While attempting to resolve the dependencies of com.anthesi.elixforms.common.model.auth.LoginCredentials errors were found",
                "detailStatus": "ERROR"
              },
              {
                "code": "400",
                "description": "[Bad Request] - Unable to perform operation: resolve on
```

```
com.anthesi.elixforms.common.model.auth.LoginCredentials",
    "detailStatus": "ERROR"
  }
]
}
},
"uuid": "b50f7989-c620-40d9-aede-e1249ef9e2e3"
}
```

• **Exception** (tutte le RIMANENTI):

```
{ "value": {
  "code": "ERROR",
  "description": "Unmanaged exception thrown. Contact technical support.",
  "complete": false,
  "globalStatus": "ERROR",
  "resultNumber": -1,
  "serviceInformationList": { "serviceInformation": [ {
    "complete": false,
    "serviceVersion": "V1",
    "statusDetailList": { "statusDetail": [ {
      "code": "500",
      "description": "[Internal Server Error] - null",
      "detailStatus": "ERROR"
    } ]
  } ] }
  } ] },
  "uuid": "e7d49547-adcb-41c7-87b5-461da2002fe7"
}}
```



- Chiamate che sollevano una `jax-rs NotFoundException` vengono lasciate proseguire con una risposta `http "404 Not Found"`
- Chiamate che sollevano una `jax-rs NotAllowedException` vengono lasciate proseguire con una risposta `http "405 Method Not Allowed"`, ad eccezione di `"PUT"`
- Attualmente, le chiamate `jax-rs` di tipo **"PUT"** restituiscono una `http response` del tipo `"406 Not Acceptable"`.