UNIONCAMERE
Unione italiana delle Camere di commercio, industria, artigianato e agricoltura
-
Ministero della Transizione Ecologica
DIREZIONE GENERALE PER L'ECONOMIA CIRCOLARE (ECi)
"Divisione IV - Pianificazione, tracciabilità e vigilanza sulla gestione dei rifiuti"
"Comitato Nazionale dell'Albo dei gestori ambientali"

Registro Elettronico Nazionale
Tracciabilità dei Rifiuti
Laboratorio Sperimentale Prototipazione Funzionale

DECRETO-LEGGE 14 dicembre 2018, n. 135
Disposizioni urgenti in materia di sostegno e semplificazione per le imprese
e per la pubblica amministrazione.

"Art. 6 - Disposizioni in merito alla tracciabilità dei dati ambientali inerenti rifiuti"
(come modificato in sede di conversione dalla Legge 11 febbraio 2019, n. 12)

Applicazione delle linee di indirizzo AGID
Modello di interoperabilità tecnica
Utilizzo, configurazione e test del servizio

Versione:	03-00
Data:	22-04-2022
Identificatore:	20220422_ModInterop.RENTRI.v03.docx
Redattori:	Diego Fenza
Approvato da:	Maurizio Zampieri
Contatti:	itec@ecocerved.it

Sommario

Avvertenza

1. NOVITÀ INTRODOTTE RISPETTO ALLA PRECEDENTE EMISSIONE

Versione/Release n°:	*03-00*	Data Versione/Release:	*22-04-2022*
Nome documento:	20220422_ModInterop.RENTRI.v03.docx		
Descrizione modifiche:	Aggiunti messaggi di errore e migliorata l'indicazione su come valorizzare il parametro x5c.		
Motivazioni:	Terza emissione.		

Versione/Release n°:	*02-00*	Data Versione/Release:	*22-06-2021*
Nome documento:	20210621_ModInterop.RENTRI.v02.docx		
Descrizione modifiche:	Eliminata la necessità di trasferire al sistema RENTRI, in fase di configurazione, il certificato X509 da utilizzare nell'interoperabilità		
Motivazioni:	Seconda emissione.		

2. RIFERIMENTI

2.1. Documenti

[d1] Modello di interoperabilità – Regole tecniche
00_linea_di_indirizzo_interoperabilita_tecnica.pdf (agid.gov.it)
https://www.agid.gov.it/sites/default/files/repository_files/00_linea_di_indirizzo_interoperabilita_tecnica.pdf
01_pattern_interazione_0.pdf (agid.gov.it)
https://www.agid.gov.it/sites/default/files/repository_files/01_pattern_interazione_0.pdf
02_pattern_sicurezza.pdf (agid.gov.it)
https://www.agid.gov.it/sites/default/files/repository_files/02_pattern_sicurezza.pdf
03_profili_di_interoperabilita.pdf (agid.gov.it)
https://www.agid.gov.it/sites/default/files/repository_files/03_profili_di_interoperabilita.pdf
04_raccomandazioni_di_implementazione.pdf (agid.gov.it)
https://www.agid.gov.it/sites/default/files/repository_files/04_raccomandazioni_di_implementazione.pdf

2.2. Link

[L0]	Linee Guida Tecnologie e standard sicurezza API | Linee Guida Tecnologie e standard per la sicurezza dell'interoperabilità tramite API dei sistemi informatici (italia.it) https://docs.italia.it/AgID/documenti-in-consultazione/lg-sicurezza-interoperabilita-docs/it/bozza/index.html
[L1]	Linee Guida sull'interoperabilità tecnica | Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni (italia.it) https://docs.italia.it/italia/piano-triennale-ict/lg-modellointeroperabilita-docs/it/bozza/index.html
[L2]	Representational State Transfer - Wikipedia https://it.wikipedia.org/wiki/Representational_State_Transfer
[L3]	RFC 7519 - JSON Web Token (JWT) (ietf.org) https://tools.ietf.org/html/rfc7519
[L4]	RFC 7807 - Problem Details for HTTP APIs (ietf.org) https://tools.ietf.org/html/rfc7807

2.3. Termini e definizioni

Termine	*Definizione*
AgID	Agenzia per l'Italia digitale.
UI	Interfaccia utente.

REST	Representational state transfer – architettura per la trasmissione di dati su http senza ulteriori livelli, quali ad esempio SOAP.
JWT	JSON Web Token – standard open ([L3] RFC 7519) per creare una collezione di dati, con firma e/o crittografia opzionale, dove il corpo (payload) contiene un JSON con una o più attestazioni (claim).
JWS	JSON Web Signature - formato standard per la firma di dati, in particolare del JSON Web Token.
CA	Autorità di certificazione.
Base 64	Sistema di codifica che consente la traduzione di dati binari in stringhe di testo.
X.509	Standard utilizzato per definire il formato dei certificati a chiave pubblica.
API	Application programming interface: insieme di procedure atte all'espletamento di un dato compito.
Epoch time	Rappresentazione del tempo nei sistemi operativi UNIC vista come la differenza in secondi rispetto alla mezzanotte (UTC) del 1º gennaio 1970 (detta epoca).

3. OBIETTIVI

In questo documento verranno indicate le modalità, relativamente ai pattern sicurezza, che permettono l'accesso ai servizi interoperanti e la verifica dell'integrità dei messaggi scambiati.

Inoltre, verrà indicata la modalità di riconoscimento dei fruitori che dovranno accedere al sistema attraverso l'interoperabilità applicativa.

Per quanto riguarda l'interfaccia dei servizi esposti si rimanda alla consultazione della specifica OpenApi e alla relativa visualizzazione UI (Swagger).

4. PREREQUISITI

Il fruitore del servizio, ossia il soggetto che interagisce tramite l'interoperabilità con il sistema, deve essere stato preventivamente accreditato al sistema, affinché possa essere riconosciuto quando si presenterà per via applicativa.

L'accreditamento avviene attraverso la procedura web esposta dal portale RENTRI, con la quale si identificano le imprese tenute all'iscrizione, e si configurano gli utenti abilitati ad operare.

L'interoperabilità applicativa si basa sullo scambio di header nella richiesta http, i quali sono firmati digitalmente utilizzando un certificato di tipo X.509 rilasciato da una CA riconosciuta, al fine di garantire l'adeguato livello di sicurezza ed integrità della trasmissione dei dati.

Il riconoscimento del fruitore del servizio applicativo nel sistema RENTRI avviene mediante l'identità digitale utilizzata per la sottoscrizione dei pacchetti di dati scambiati.

4.1. Certificato

Il certificato utilizzato dal fruitore deve corrispondere ad un certificato rilasciato da una CA qualificata in conformità al regolamento eIDAS.

Il certificato può essere intestato a vari soggetti:

- All'impresa, ossia l'identificativo indicato nel campo "*subject*" del certificato deve corrispondere al codice fiscale (o alla partita IVA) dell'impresa/ente iscritta al RENTRI;
- Al legale rappresentante, o comunque ad altra persona con poteri di rappresentanza dell'impresa iscritta al RENTRI;

- A persona delegata secondo le modalità fornite attraverso apposita procedura web nell'area riservata agli iscritti al RENTRI.

Può essere utilizzato il certificato della CNS intestata alla persona (legale rappresentante o suo delegato), oppure un dispositivo HSM nel quale custodire il certificato, oppure un servizio di firma remota, purché il certificato sia emesso da una CA riconosciuta da AgID, o riconosciuta da altro ente comunitario secondo il regolamento eIDAS.

4.2. Riconoscimento dei fruitori

Per l'utilizzo dei servizi interoperanti sia in ambiente dimostrativo che in ambiente effettivo, occorre che il fruitore del servizio sia riconosciuto dal sistema.

Questo avviene attraverso l'accreditamento e la configurazione preliminare dei soggetti registrati in RENTRI tramite le funzioni fornite dal portale web.

[n.b.] La sola registrazione nell'area di produzione non abilita automaticamente anche all'utilizzo dei servizi in area dimostrativa.

Servizio area demo:	*https://demoprototipo.rentri.it/api*
Servizio area produzione:	*https://prototipo.rentri.it/api*

5. INDICAZIONI PRELIMINARI

5.1. Architettura

Il servizio utilizza l'architettura "RESTful API" (*REpresentational State Transfer Application Programming Interface*) attraverso l'esposizione di una serie di endpoint web (delle URL) che rispondono alle richieste fatte da uno sviluppatore attraverso il protocollo HTTP.

Il principio REST stabilisce una precisa mappatura tra le tipiche operazioni CRUD (creazione, lettura, aggiornamento, eliminazione di una risorsa) e i metodi HTTP.

Metodo http	*Operazione CRUD*	*Descrizione*
POST	Create	Crea una nuova risorsa
GET	Read	Ottiene una risorsa esistente
PUT	Update	Aggiorna una risorsa o ne modifica lo stato
DELETE	Delete	Elimina una risorsa

Tabella 1 - Mappatura tra le operazioni CRUD ed i metodi http

5.2. Versionamento

L'aggiornamento e l'evoluzione dei servizi esposti avviene mediante l'indicazione di un numero di versione direttamente nel path che qualifica l'URL web.

In caso di breaking changes verrà creata una nuova versione, mentre la precedente continuerà a esistere fino alla completa dismissione. La formazione della URL per la richiesta con il numero di versione, avviene inserendo dopo l'indirizzo del servizio, la stringa seguente:

/v<Major>[.<Minor>]

dove *Major* e *Minor* rappresentano la versione del servizio.

Esempio:	*https://servizio/**v1.0**/endpoint*

5.3. Catalogo API

Il catalogo delle API viene esposto tramite la specifica OpenApi 3, ed è navigabile anche attraverso l'interfaccia Swagger apponendo dopo l'indirizzo del servizio la stringa *"/swagger"*.

Di seguito vengono indicati gli indirizzi dei servizi dell'ambiente demo e produzione.

Esempio:	*https://prototipo.rentri.it/api/swagger*

5.4. Messaggi di errore

In caso di status code della risposta diverso da 200 (*"2xx" il 2 iniziale => successo*), il sistema risponde con un oggetto JSON secondo quanto indicato dall'RFC 7807.

Alla struttura standard contenente le proprietà *type*, *title* e *status*, si aggiunge una proprietà *modelState*, contenente un dizionario, le cui chiavi rappresentano i riferimenti dei dati in cui è stato rilevato l'errore, e i cui valori contengono un array dei relativi codici di errore.

In caso di errore generico la chiave del dizionario è valorizzata a *generic*.

Codice	Descrizione
agIDInterop.missingAuthorizationBearerHeader	Mancanza dell'header Authorization Bearer
agIDInterop.missingAgIDJWTSignatureHeader	Mancanza dell'header Agid-JWT-Signature (dove richiesto)
agIDInterop.invalidToken	Token JWT non valido
agIDInterop.invalidIssuerSigningKey	Firma del token JWT non valida
agIDInterop.invalidLifetime	Verifica di validità del token fallita
agIDInterop.invalidAudience	Verifica del claim Audience fallita
agIDInterop.invalidJwtId	JWT Id non trovato o non valido
agIDInterop.notUniqueJwtId	JWT Id non unico
agIDInterop.invalidCertificate	Certificato di firma del JWT non valido
agIDInterop.invalidIssuer	Verifica del claim Issuer fallita
agIDInterop.invalidClaim	Verifica di altri claim fallita
agIDInterop.invalidDigest	Validazione del Digest header fallita
agIDInterop.invalidSignedHeaders	Claim signed_headers non valido
agIDInterop.invalidSignedHeaderDigest	Validazione del Digest signed header fallita
agIDInterop.invalidSignedHeaderContentType	Validazione del ContentType fallita (se presente)
agIDInterop.invalidSignedHeaderContentEncoding	Validazione del ContentEncoding fallita (se presente)
sys.required	Campo obbligatorio
sys.invalid	Campo non formalmente valido
sys.noData	Nessun dato
sys.outOfRange	Valore al di fuori dei limiti consentiti
sys.outOfSize	Numero di valori al di fuori del limite consentito
sys.unknownModel	Modello dati non riconosciuto
sys.invalidBase64String	Codifica base64 errata
sys.genericError	Errore generico

Tabella 2 - Elenco dei codici di errore più comuni.

5.4.1. *Esempio di messaggio di errore*

Esempio JSON:

```
{
  "type": "https://httpstatuses.com/400",
  "title": "Bad Request",
  "status": 400,
  "modelState": {
    "[0].dati": [
      "sys.required"
    ],
    "[0].sedeVenditore": [
      "sys.required"
    ],
    "[0].sedeAcquirente": [
      "sys.required"
    ]
  }
}
```

6. AUTENTICAZIONE

Nei prossimi capitoli si farà riferimento espressamente al documento [d1] - 02_pattern_sicurezza.pdf (agid.gov.it) delle guida AgID relativamente all'interoperabilità, ed in particolare ai pattern sicurezza: **ID_AUTH_REST_0*** e **INTEGRITY_REST_01**.

Inoltre, si farà riferimento al token JWT (JSON Web Token) ossia allo standard per la creazione di modelli di dati, con firma, dove il corpo (payload) contiene un JSON con una o più attestazioni (claim).

Si faranno riferimento anche alle relative attestazioni (claim) più comuni: iat, exp, nbf, aud, iss, …

Per l'autenticazione ai servizi viene utilizzato il pattern sicurezza delle linee guida AgID **ID_AUTH_REST_02**.

6.1. Regole di processamento

Il pattern prevede la generazione di un token JWT, costruito nelle modalità di seguito riportate, e firmato con il certificato riconosciuto in RENTRI come descritto in precedenza.

Il token JWT firmato, utilizzando lo standard JWS (JSON Web Signature), dovrà essere poi trasmesso nell'header della richiesta http *Authorization Bearer*.

6.1.1. *Costruzione dell'header*

Dovranno essere indicati i parametri:

- ***alg*** *(Algorithm)* con l'algoritmo di firma (es RS256, ES256, …);
- ***typ*** *(Type)* impostato a *JWT*;
- ***x5c*** (*spec.AgId X.509 Certificate Chain*) con un array il cui unico elemento deve essere una stringa corrispondente alla rappresentazione in Base64 del certificato X.509 utilizzato per firmare il token JWT e riconosciuto in RENTRI.

6.1.2. *Costruzione del payload*

Nel payload dovranno essere indicate le seguenti attestazioni (claim):

- ***exp*** *(Expiration Time),* ***iat*** *(Issued At)e* ***nbf*** *(Not Before)* per indicare rispettivamente la scadenza del token, il momento del rilascio del token, il momento da cui il token può ritenersi valido. I valori sono rappresentati secondo il formato '*epoch time*'.
- ***aud*** *(Audience)* impostato alla costante "*rentri.api*";
- ***iss*** *(Issuer)* impostato con l'identificativo a cui è intestato il certificato utilizzato per la firma del token JWT ed indicato nel header paramenter ***x5c***;
- ***jti*** *(JWT ID)* identificativo del token JWT, questo valore deve essere univoco per il fruitore e non ha una formattazione particolare.

6.1.3. Esempio

```
# http request
Authorization: Bearer eyJhbGciOiJSUzI1NiIs...

# deserializzazione del token JWT
# header
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": ["MIIH9jCCA96..."]
}
# payload
{
  "jti": "44ad6ba0-eaf3-4ad1-9557-968347781112",
  "aud": "rentri.api",
  "iss": "04527551008",
  "exp": 1619774997,
  "iat": 1619774877,
  "nbf": 1619774877
}
```

7. INTEGRITÀ

Per gli endpoint con verbo http **POST** o **PUT** che prevedono un body JSON, all'header http '*Authorization Bearer*', va aggiunto l'header http '*Agid-JWT-Signature*' che permette la verifica dell'integrità del dato stesso.

Viene utilizzato il pattern sicurezza INTEGRITY_REST_01 proposto dalle linee guida AgID per l'interoperabilità.

7.1. Regole di processamento

Il pattern INTEGRITY_REST_01 è un'estensione del ID_AUTH_REST_02 e prevede anch'esso la generazione di un token JWT firmato con certificato riconosciuto in RENTRI.

Il token JWT dovrà essere poi trasmesso nell'header della richiesta http '*Agid-JWT-Signature*', in aggiunta a '*Authorization Bearer*'.

L'header del token JWT prevede gli stessi parametri del ID_AUTH_REST_02 (vedi capitolo precedente), mentre il payload dovrà contenere il claim aggiuntivo '*signed_headers*' descritto nel prossimo paragrafo.

7.1.1. Claim signed_headers

Questo claim contiene valori multipli, quindi non contiene una stringa, ma un array di coppie "chiave-valore":

- *digest* contiene il valore dell'http header *digest*, questo valore corrisponde al Base 64 dell'hash SHA 256 calcolato sul body della request, a cui deve essere apposta l'indicazione dell'algoritmo utilizzato: *"SHA-256="*;
- *content-type* (quando presente nell'http Header) contiene il valore dell'http header *content-type*;
- *content-encoding* (quando presente nell'http Header) contiene il valore dell'http header *content-encoding*.

7.1.2. Esempio

```
# http request
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsI...
Agid-JWT-Signature: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCI...
Digest: SHA-256=pMTw2VNDHuvxNP6SKK30tq/09dgT6TOWFs6zAap795I=
Content-Type: application/json; charset=utf-8

# deserializzazione del token JWT
# header
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": ["MIIH9jCCA96..."]
}

# payload
{
  "jti": "fbbc862e-be92-4c7d-90e9-b1e2da0e262e",
  "signed_headers": [
    {
      "digest": "SHA-256=pMTw2VNDHuvxNP6SKK30tq/09dgT6TOWFs6zAap795I="
    },
    {
      "content-type": "application/json; charset=utf-8"
    }
  ],
  "aud": "rentri.api",
  "iss": "04527551008",
  "exp": 1619794064,
  "iat": 1619793944,
  "nbf": 1619793944
}
```

7.2. Integrità delle risposte

In caso di risposta con successo da parte delle API (http status code 2**), anche nella *response* generata dal sistema verrà applicato l'header *Agid-JWT-Signature*, in questo modo anche il fruitore potrà verificare l'integrità delle risposte fornite.

Per tale verifica l'utente potrà applicare i seguenti passi:

- verifica della validità della firma (utilizzando il certificato nell'header *x5c*);
- verifica della validità temporale del token;
- verifica del claim *aud* impostato alla costante "*rentri.api*";
- verifica del digest SHA 256 del body della risposta.

8. ASSISTENZA

Per supporto tecnico ed ogni altra richiesta di assistenza, è necessario utilizzare la sezione "Assistenza" accessibile dalla home page del portale.

Le richieste inoltrate saranno gestite dallo stesso staff di supporto, indifferentemente che siano state inserite dall'area dimostrativa oppure pervengano dall'area di lavoro effettiva.

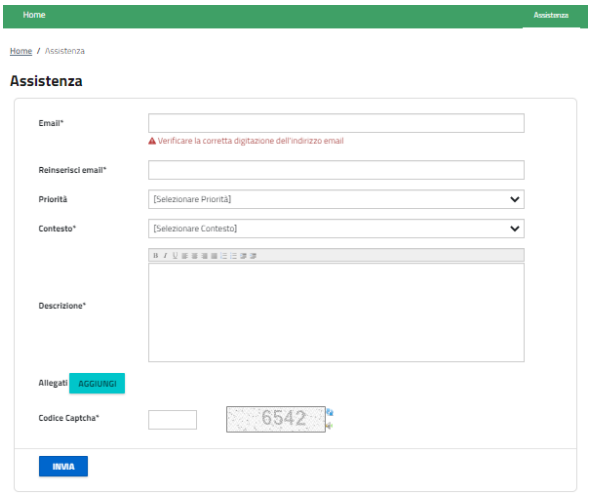

Figura 1 - Form per la richiesta di assistenza

9. ESEMPIO

Esempio di codice in Microsoft C# .NET 6 che permette di eseguire il test delle API utilizzando una CNS.

```
using Microsoft.IdentityModel.JsonWebTokens;
using Microsoft.IdentityModel.Tokens;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;


// Configurazione
var issuer = "XXXXXX00X00X000X"; // Indicare l'identificativo del certificato della CNS e legato al sub

var aud = "rentri.api"; // Per demo demorentri.api
var baseApi = "https://testprototipo.rentri.it/api"; // Per demo https://demoprototipo.rentri.it/api
                                                     // Per produzione https://prototipo.rentri.it/api
var api = $"{baseApi}/v1.0/registri/REG001D/movimenti"; // Per gli altri endpoint consultare
                                                        // https://prototipo.rentri.it/api/swagger

var jti = Guid.NewGuid().ToString(); // Id del JWT

// Dati scambiati
var content = new StringContent(@"[{""progressivo"": 1}]", System.Text.Encoding.UTF8, "application/json");

// Recupero del certificato
using var storeMy = new X509Store(StoreName.My);
storeMy.Open(OpenFlags.ReadOnly);
var cert = storeMy.Certificates.Cast<X509Certificate2>().First(c => c.HasPrivateKey &&
                                                                    c.Subject.ToUpper().Contains(issuer));

// ID_AUTH_REST_02
var tokenHandler = new JsonWebTokenHandler();
var tokenDescriptor = new SecurityTokenDescriptor {
    AdditionalHeaderClaims = new Dictionary<string, object> {
        { "x5c", new string[] { Convert.ToBase64String(cert.Export(X509ContentType.Cert)) }}
    },
    Audience = aud,
    Issuer = issuer,
    Claims = new Dictionary<string, object> { { "jti", jti } },
    SigningCredentials = new X509SigningCredentials(cert)
};

var idAuth = tokenHandler.CreateToken(tokenDescriptor);

// INTEGRITY_REST_02
using var sha256 = SHA256.Create();
var digest = $"SHA-256={Convert.ToBase64String(sha256.ComputeHash(await
                                                                    content.ReadAsByteArrayAsync()))}";
tokenDescriptor.Claims.Add("signed_headers", new List<Dictionary<string, string>> {
    new Dictionary<string, string> { { "digest", digest } },
    new Dictionary<string, string> { { "content-type", content.Headers.ContentType?.ToString()! } }
});

var integrity = tokenHandler.CreateToken(tokenDescriptor);

// Client con headers
using var cli = new HttpClient();
cli.DefaultRequestHeaders.Add("Authorization", $"Bearer {idAuth}");
cli.DefaultRequestHeaders.Add("Digest", digest);
cli.DefaultRequestHeaders.Add("Agid-JWT-Signature", integrity);

// Chiamata API
var res = await cli.PostAsync(api, content);
var response = await res.Content.ReadAsStringAsync();

Console.WriteLine(response);
```