AGID | Agenzia per l'Italia Digitale

# Linee Guida Modello Interoperabilita

***Release Bozza in consultazione***

**italia**

**22 giu 2023**

# Indice

Questo documento raccoglie il testo delle *Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni*, e dei Documenti Operativi ad essa allegati, adottate con Determinazione AgID n. 547 del 1 ottobre 2021 e disponibili sul sito AgID[8]

[8] https://www.agid.gov.it/sites/default/files/repository_files/linee_guida_interoperabilit_tecnica_pa.pdf

CAPITOLO 1

# Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni

## 1.1 Introduzione

Le Linee Guida individuano le tecnologie e gli standard che le Pubbliche Amministrazioni devono tenere in considerazione durante la realizzazione dei propri sistemi informatici, al fine di permettere il coordinamento informativo e informatico dei dati tra le amministrazioni centrali, regionali e locali, nonché tra queste e i sistemi dell'Unione Europea, con i gestori di servizi pubblici e dei soggetti privati.

Le Linee Guida assicurano l'aggiornamento rispetto alla:

- evoluzione della tecnologia;
- aderenza alle indicazioni europee in materia di interoperabilità;
- adeguatezza alle esigenze delle pubbliche amministrazioni e dei suoi utenti;
- adozione da parte di tutti i soggetti, pubblici e privati;
- adeguatezza dei necessari livelli di sicurezza.

Le Linee Guida contribuiscono alla definizione del Modello di Interoperabilità della PA (in breve ModI), focalizzandosi sulle tecnologie e le loro modalità di utilizzo, per assicurare lo scambio di dati tra le PA, e tra queste e i soggetti privati; in esse sono definiti i contesti di interazione e integrazione tra le PA, i cittadini e le imprese.

La definizione del ModI è coerente con il nuovo European Interoperability Framework (in breve EIF) oggetto della Comunicazione COM (2017) 134 della Commissione Europea del 23 marzo 2017[1], al fine di assicurare l'interoperabilità nel contesto Europeo e per l'attuazione del Digital Single Market (Mercato Unico Digitale).

Nell'ottobre 2005 il CNIPA (oggi Agenzia per l'Italia digitale - AgID) ha pubblicato un insieme di documenti che costituivano il riferimento tecnico per l'interoperabilità fra le PA. Tali documenti delineavano il quadro tecnico-implementativo del Sistema pubblico di cooperazione (SPCoop), il framework di interoperabilità a livello applicativo adottato dalle PA.

Le Linee Guida, e più in generale il ModI, completano il processo di aggiornamento del SPCoop avviato con la determinazione AgID 219/2017 - «Linee guida per transitare al nuovo modello di interoperabilità»[2].

[1] Cf. https://eur-lex.europa.eu/legal-content/IT/TXT/?qid=1584086617794&uri=CELEX:52017DC0134

[2] Cf. https://www.agid.gov.it/sites/default/files/repository_files/upload_avvisi/linee_guida_passaggio_nuovo_modello_interoperabilita.pdf

**Le Linee Guida considerano i soli servizi digitali (di seguito e-service) realizzati da una Pubblica Amministrazione per assicurare l'accesso ai propri dati e/o l'integrazione dei propri processi attraverso l'interazione dei suoi sistemi informatici con quelli dei fruitori.**

Le Linee Guida contribuiscono alla definizione del Modello di Interoperabilità della PA (in breve ModI) relativamente agli e-service provvedendo a:

- definire le modalità di integrazione tra le PA, e tra queste, cittadini e imprese, armonizzando le scelte architetturali di interoperabilità delle PA;
- individuare le scelte tecnologiche che favoriscano lo sviluppo, da parte delle PA, cittadini e imprese, di soluzioni applicative innovative che semplifichino e abilitino l'utilizzo dei dati e dei servizi digitali;
- promuovere l'adozione dell'approccio «API first» per favorire la separazione dei livelli di backend e frontend, con logiche aperte e standard pubblici che garantiscano ad altri attori, pubblici e privati, accessibilità e massima interoperabilità di dati e servizi digitali;
- privilegiare standard tecnologici, de iure e de facto, che soddisfino l'esigenza di rendere sicure le interazioni tra le PA, e tra queste, cittadini e imprese;
- favorire l'interazione tra PA, e tra queste, cittadini e imprese, attraverso un approccio Contract-First.

Il ModI è coerente con European Interoperability Framework (EIF) che fornisce orientamenti alle PA Europee su come operare le iniziative relative al tema dell'interoperabilità attraverso una serie di raccomandazioni atte a stabilire relazioni tra le varie organizzazioni, razionalizzare i processi volti a sostenere i servizi digitali e assicurare che le norme esistenti e quelle nuove non pregiudichino gli sforzi di interoperabilità[3].

**Le Linee Guida affrontano il livello di interoperabilità tecnica prevista nell'EIF[4] relativamente agli Interoperable Digital Public Services.**

Il ModI assicura l'adozione degli standard di interoperabilità mandatori individuati dalla Comunità Europea (ad esempio INfrastructure for SPatial InfoRmation in Europe[5]) e sostiene le iniziative di standardizzazione promosse dalla Commissione Europea (ad esempio Solution Architecture Template for e-Procurement[6], Solution Architecture Template for Open Data[7]).

## 1.2 Ambito di applicazione

Le Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni sono redatte in ottemperanza:

- alla lettera a comma 2 dell'articolo 14-bis del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale), che individua le funzioni dell'Agenzia per l'Italia Digitale (AgID) tra cui: «emanazione di Linee [...] di indirizzo [...] sull'attuazione e sul rispetto delle norme di cui al presente Codice, anche attraverso l'adozione di atti amministrativi generali, in materia di [...] interoperabilità e cooperazione applicativa tra sistemi informatici pubblici e quelli dell'Unione europea»;
- alla lettera b comma 3-ter dell'articolo 73 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale), in cui è indicato che il Sistema pubblico di connettività è costituito da «linee guida e regole per la cooperazione e l'interoperabilità».

Le Linee Guida sono emanata ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

### 1.2.1 Soggetti destinatari

Le Linee Guida sono destinate ai soggetti di cui all'articolo 2, comma 2 del CAD, così come indicato dall'articolo 75 dello stesso. I destinatari la attuano nella realizzazione dei propri sistemi informatici che fruiscono o erogano dati e/o servizi digitali ad altri soggetti.

[3] Cf. https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/eif-european-interoperability-framework-0
[4] Cf. https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/3-interoperability-layers#3.6
[5] Cf. https://joinup.ec.europa.eu/collection/inspire
[6] Cf. https://joinup.ec.europa.eu/solution/sat-e-procurement
[7] Cf. https://joinup.ec.europa.eu/solution/sat-open-data

Per gli e-service implementati dalle Pubbliche Amministrazioni prima dell'emanazione delle Linee Guida, al fine di assicurare la convergenza al ModI, si richiede di:

- assicurare per i nuovi fruitori l'applicazione di modalità di fruizione conformi alle presenti Linee Guida;
- prevedere, a valle di una valutazione di impatto che consideri la sicurezza dei servizi e l'effetto sui fruitori, la dismissione delle modalità difformi alle presenti Linee Guida.

Le Linee Guida sono rivolte ai soggetti privati che devono interoperare con la Pubblica Amministrazione per fruire di dati e/o servizi tramite sistemi informatici.

## 1.3 Riferimenti e sigle

### 1.3.1 Note di lettura del documento

Conformemente alle norme ISO/IEC Directives, Part 3 per la stesura dei documenti tecnici le presenti Linee Guida utilizzano le parole chiave «DEVE», «DEVONO», «NON DEVE», «NON DEVONO», «DOVREBBE», «NON DOVREBBE», «PUÒ» e «OPZIONALE», la cui interpretazione è descritta di seguito.

- **DEVE** o **DEVONO**, indicano un requisito obbligatorio per rispettare le Linee Guida;
- **NON DEVE** o **NON DEVONO**, indicano un assoluto divieto delle specifiche;
- **DOVREBBE** o **NON DOVREBBE**, indicano che le implicazioni devono essere comprese e attentamente pesate prima di scegliere approcci alternativi;
- **PUÒ** o **POSSONO** o l'aggettivo **OPZIONALE**, indica che il lettore può scegliere di applicare o meno senza alcun tipo di implicazione o restrizione la specifica.

### 1.3.2 Struttura

Le Linee Guida includono i seguenti Documenti operativi, che individuano gli standard tecnologici e le loro modalità di utilizzo al fine di fruire e/o erogare dati e/o servizi digitali per il tramite dei propri sistemi informatici.

Documenti operativi

- Documento operativo - Pattern di interazione
- Documento operativo - Pattern di sicurezza
- Documento operativo - Profili di interoperabilità
- Documento operativo - Raccomandazioni di implementazione

Al fine di assicurare l'allineamento costante delle Linee Guida alle continue evoluzione tecnologica, l'aggiornamento dei Documenti operativi è realizzato attraverso Circolari emanate dall'AgID.

### 1.3.3 Riferimenti Normativi

Sono riportati di seguito gli atti normativi di riferimento del presente documento.

Tabella 1.1: Riferimenti Normativi

| | |
|---|---|
| [CAD] | decreto legislativo 7 marzo 2005, n. 82 recante «Codice dell'Amministrazione Digitale»; NOTA – Il D. Lgs. 82/2010 è noto anche con l'abbreviazione «CAD» |
| [EIF] | European Interoperability Framework (EIF) |
| [CE 2008/1205] | Regolamento (CE) n. 1205/2008 della Commissione del 3 dicembre 2008 recante attuazione della direttiva 2007/2/CE del Parlamento europeo e del Consiglio per quanto riguarda i metadati |
| [D.lgs. 196/2003] | Codice in materia di protezione dei dati personali |
| [UE 679/2016] | Regolamento (UE) 2016/679 del 27 aprile 2016 relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali (in breve GDPR) |
| [UE 910/2014] | Regolamento (UE) n. 910/2014 del 23 luglio 2014 in materia di identificazione elettronica e servizi fiduciari per le transazioni elettroniche nel mercato interno (in breve eIDAS) |
| [DUE 2019/1024] | Direttiva (UE) 2019/1024 del Parlamento Europeo e del Consiglio del 20 giugno 2019 relativa all'apertura dei dati e al riutilizzo dell'informazione del settore pubblico |

### 1.3.4 Termini e definizioni

Tabella 1.2: Termini e definizioni

| | |
|---|---|
| [AgID] | Agenzia per l'Italia Digitale |
| [API] | Application Programming Interface |
| [API-First] | L'API-first è un approccio in cui le PA considerano le API come mezzo principale per perseguire i propri obiettivi, interagendo con i propri stakeholder sin dalla fase di progettazione. Come indica il CAD art. 64-bis, comma 1-bis quindi, le interfacce applicative (API appunto) devono essere progettate e/o evolute in maniera interoperabile, a prescindere dai canali di erogazione del servizio che sono individuati logicamente e cronologicamente dopo la progettazione dell'API |
| [BP] | WS-I Basic Profile - (Web Services Interoperability Specification) |
| [CAD] | Codice Amministrazione Digitale, D.lgs. 7 marzo 2005, n. 82 |
| [Contract-First] | Contract-first è un approccio che prevede di dare seguito all'interazione di più sistemi informatici definendo le API condivise attraverso un Interface Description Language (IDL) |
| [EIF] | European Interoperability Framework |
| [Enti Capofila] | Gli enti capofila sono pubbliche amministrazioni che si propongono nel ModI quali soggetti responsabili delle attività di gestione sul Catalogo degli e-service, delle API e degli accordi di interoperabilità nelle veci di altre Pubbliche Amministrazioni |
| [Erogatore] | Uno dei soggetti di cui all'articolo 2, comma 2 del CAD che rende disponibile e-service ad altre organizzazioni, per la fruizione di dati in suo possesso o l'integrazione dei processi da esso realizzati |
| [e-service] | I servizi digitali realizzati ai sensi del CAD art. 1, comma 1, lettera n-quater) da un erogatore per assicurare l'accesso ai propri dati e/o l'integrazione dei propri processi attraverso l'interazione dei suoi sistemi informatici con quelli dei fruitori, trovano attuazione nell'implementazione di API |
| [Fruitore] | Un'organizzazione che utilizza gli e-service messi a disposizione da un dei soggetti di cui all'articolo 2, comma 2 del CAD |
| [HTTP] | Hypertext Transfer Protocol |
| [IDPS] | Interoperable Digital Public Services |
| [JWT] | JSON Web Tokens |
| [ModI] | Modello di Interoperabilità delle Pubbliche Amministrazioni Italiane |
| [PA] | Pubblica Amministrazione Italiana |
| [QoS] | Quality of Service |
| [REST] | Representational State Transfer |
| [RPC] | Remote Procedure Call |
| [SLA] | Service Level Agreement |
| [SLI] | Service Level Indicator |
| [SLO] | Service Level Objective |
| [SOAP] | Simple Object Access Protocol |
| [UML] | Linguaggio di modellazione unificato (Unified Modeling Language) |
| [W3C] | World Wide Web Consortium |
| [WS-*] | Lo stack degli standard emanati relativi alle tecnologie SOAP, tra cui SOAP, WSDL, WS-Security, WS-Addressing e WS-I |
| [WSDL] | Web Services Description Language |
| [XML] | eXtensible Markup Language |
| [XML-RPC] | XML-Remote Procedure Call |

## 1.4 Principi generali

### 1.4.1 Interazioni

L'ambito di applicazione delle Linee Guida, in coerenza con il ModI, comprende i tre tipi di interazioni previste nell'EIF che vedono coinvolte Pubbliche Amministrazioni, cittadini e imprese.

Le interazioni prevedono che i soggetti coinvolti possano svolgere la funzione di **erogatore di servizi, quando il soggetto mette a disposizione servizi digitali utilizzati da altri soggetti**, e la funzione di **fruitore di servizi, quando il soggetto utilizza i servizi digitali messi a disposizione da un altro soggetto**.

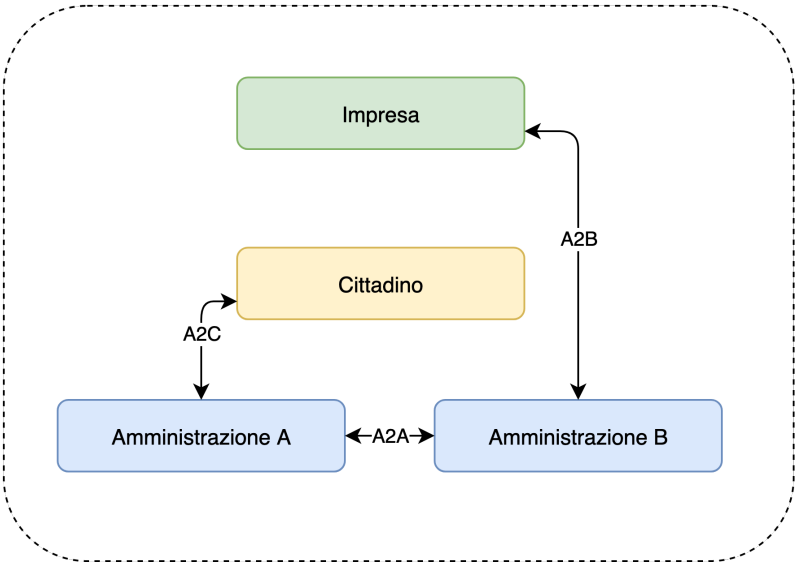


*Figura 1 - Ambito di applicazione del modello di interoperabilità*

I soggetti fruitori possono utilizzare i servizi digitali in maniera trasparente all'erogatore, attraverso:

- una soluzione software attivata da un attore umano (user agent/human);
- un sistema applicativo automatico (server/machine), anche allo scopo di definire nuovi servizi a valore aggiunto.

### 1.4.2 Application Programming Interface (API)

Con Application Programming Interface (API) si indica ogni insieme di procedure/funzionalità/operazioni disponibili al programmatore, di solito raggruppate a formare un insieme di strumenti specifici per l'espletamento di un determinato compito. Spesso, con tale termine si intendono le librerie software disponibili in un certo linguaggio di programmazione. Una buona API fornisce una «scatola nera», cioè un livello di astrazione che evita al programmatore di conoscere i dettagli implementativi dell'API stessa. Questo permette di ri-progettare o migliorare le funzioni all'interno dell'API, senza cambiare il codice che si affida ad essa. La finalità di un'API è di ottenere un'astrazione a più alto livello, di solito tra lo strato sottostante l'API e i suoi consumatori (client).

Con Web API si indicano le API rese disponibili al client attraverso Internet (prevalentemente sul Web, che si basa sul protocollo HTTP).

Per il World Wide Web Consortium (W3C), un web service è qualsiasi software disponibile su Internet che standardizza la sua interfaccia tramite la codifica eXtensible Markup Language (XML). Un client interroga un servizio web inviando una richiesta in formato XML; il servizio web ritorna una risposta utilizzando l'analogo formato. Client e web service comunicano attraverso una rete che li connette e sfruttano generalmente il protocollo applicativo HTTP. I web service si basano principalmente su standard come XML-Remote Procedure Call (XML-RPC) e Simple Object Access Protocol (SOAP). Quindi, un web service è un possibile modo di realizzare una Web API. A partire dalla seconda metà degli anni 2000, creando possibili confusioni, il termine Web API è stato utilizzato come alternativa a web service per indicare altri approcci/protocolli/tecnologie, come le API REpresentational State Transfer (REST) per realizzare API senza utilizzare XML-RPC e SOAP.

**Nell'ambito delle Linee Guida, accettando la nomenclatura in uso a livello europeo e più in generale nel contesto internazionale, si utilizza il termine generico API per indicare indifferentemente le Web API, i web service e le API REST, lasciando al contesto in cui lo stesso è utilizzato la declinazione del significato esplicito.**

Le Linee Guida individuano le modalità con cui le Pubbliche Amministrazioni implementano le proprie API, quale elemento tecnologico di base del ModI, attraverso cui le Pubbliche Amministrazioni rendono disponibile gli e-service utilizzati dai sistemi informatici di altre Pubbliche Amministrazioni, cittadini e imprese.

### 1.4.3 Qualità dei servizi

Il concetto di qualità di servizio, anche comunemente chiamata Quality of Service (QoS), fa riferimento alla descrizione non funzionale di una API, cioè alla capacità di quest'ultima di soddisfare le aspettative dei fruitori.

Assicurare la QoS nell'ambito Internet ai fini dell'interoperabilità è una sfida critica a causa della natura dinamica e impredicibile del contesto applicativo. Cambiamenti negli schemi di traffico, la presenza di transazioni critiche, gli effetti dei problemi di rete, le performance dei protocolli e degli standard di rete richiedono una definizione precisa della QoS offerta da una API.

Gli elementi chiave a supporto della QoS possono essere riassunti come segue:

- Disponibilità. La probabilità che una API sia disponibile e funzionante in un istante casuale. Associato al concetto di disponibilità è quello di Time-To-Repair (TTR), cioè il tempo necessario a ripristinare una API una volta che questa diventa indisponibile. La disponibilità di una API dovrebbe potere essere verificata tramite l'esposizione di una API di monitoraggio, dedicata e a basso impatto (quindi a elevata disponibilità);
- Accessibilità. Misura la capacità di una API di essere contattabile in un qualunque istante di tempo;
- Prestazioni. Le prestazioni vengono misurate solitamente rispetto a due valori: il throughput e la latenza. Il throughput rappresenta il numero di richieste soddisfatte in un dato intervallo. La latenza rappresenta la quantità di tempo che passa tra l'invio di una richiesta e la ricezione di una risposta. Una API con buone prestazioni ha un elevato throughput e una bassa latenza;
- Affidabilità. Rappresenta la capacità di una API di funzionare correttamente e consistentemente, fornendo la stessa QoS a dispetto di malfunzionamenti di diversa natura. Di solito è espressa in termini di fallimenti in un dato lasso di tempo;
- Scalabilità. L'abilità di servire in modo consistente, efficiente e performante le richieste all'aumentare o al diminuire del loro numero. È strettamente connessa al concetto di accessibilità;
- Sicurezza. La sicurezza implica aspetti quali confidenzialità, integrità, autorizzazione e autenticazione;
- Transazionalità. Ci sono alcuni casi (ad es., API stateful) in cui è necessario assicurare l'esecuzione transazionale di una operazione. La capacità di una operazione di rispettare questa proprietà è parte della QoS.

Gli erogatori di API devono prendere tutte le iniziative necessarie a mantenere i requisiti di QoS richiesti dal caso d'uso. Questo include anche l'utilizzo di buone pratiche, ad esempio, per assicurare prestazioni e scalabilità, il risparmio della banda è una condizione fondamentale.

### 1.4.4 Service Level Agreement – SLA

L'integrazione può coinvolgere numerose organizzazioni e erogatori esterni di API. Al fine di accordarsi sulla QoS, gli erogatori e i fruitori di API utilizzano quelli che vengono definiti Service Level Agreement (SLA), ovvero accordi sul livello di servizio. Un SLA può contenere le parti seguenti:

- Scopo. Le ragioni che hanno portato alla definizione del SLA;
- Parti. I soggetti interessati dal SLA e i rispettivi ruoli (ad es., l'erogatore e il fruitore della API);
- Periodo di validità. L'intervallo di tempo, espresso mediante data, ora di inizio, data e ora di fine, per il quale si ritiene valido un particolare termine di accordo all'interno degli SLA;
- Perimetro. Quali sono operazioni interessate dallo specifico SLA;

- Service Level Objectives (SLO), ovvero obiettivi sul livello di servizio. I singoli termini di accordo all'interno di un SLA. Di solito, sono definiti utilizzando dei Service Level Indicators (SLI), ovvero indicatori sul livello di servizio, che quantificano i singoli aspetti di QoS (ad es., la disponibilità);
- Penalità. Le sanzioni che si applicano nel caso che l'erogatore dell'interfaccia di servizio non riesca ad assicurare gli obiettivi specificati nel SLA;
- Esclusioni. Gli aspetti della QoS non coperti dal SLA;
- Amministrazione. I processi mediante i quali le parti possono monitorare la QoS.

**Gli SLA possono essere statici o dinamici. Negli SLA dinamici, i SLO** (con associati SLI) variano nel tempo e i periodi di validità definiscono gli intervalli di validità di questi ultimi (ad es., in orario lavorativo i SLO possono essere differenti da quelli imposti durante la notte). La misurazione dei livelli di QoS all'interno di un SLA richiedono il tracciamento delle operazioni effettuate in un contesto infrastrutturale multi-dominio (geografico, tecnologico e applicativo).

### 1.4.5 Dominio di interoperabilità

Nell'ambito delle presenti Linee Guida, per dominio di interoperabilità si indica uno specifico contesto in cui più Pubbliche Amministrazioni e/o soggetti privati hanno l'esigenza di scambiare dati e/o integrare i propri processi per dare seguito al disposto normativo.

Ogni dominio di interoperabilità è caratterizzato da:

- i soggetti partecipanti, le Pubbliche Amministrazioni e gli eventuali soggetti privati (cittadini e imprese);
- i sistemi informatici dei soggetti partecipanti che scambiano dati e/o integrano i propri processi;
- l'insieme di API implementate per garantire le interazioni tra i sistemi informatici dei soggetti partecipanti;
- i criteri di sicurezza che le singole API forniscono per assicurare transazioni tra i soggetti partecipanti conformi alla norma.

### 1.4.6 Logging

Il logging riveste un ruolo fondamentale nella progettazione e nello sviluppo di API. Le moderne piattaforme middleware, oltre ad integrare meccanismi di logging interni, possono connettersi ad API esterne che permettono la raccolta (log collection), la ricerca e la produzione di analitiche, utili tra l'altro all'identificazione di problemi e al monitoraggio del sistema e della QoS. L'utilizzo di log collector permette di centralizzare non solo i log relativi all'utilizzo delle API, ma anche quelli di eventuali altri servizi digitali e componenti di rete (ad es., proxy e application-gateway). Ai fini di non ripudio, i messaggi applicativi possono essere memorizzati insieme alla firma digitale, ed archiviati nel rispetto della normativa sulla conservazione e sulla privacy. L'erogatore deve documentare in dettaglio il formato e le modalità di tracciatura, consultazione e reperimento delle informazioni. L'erogatore non deve tracciare nei log segreti quali password, chiavi private o token di autenticazione. L'erogatore deve tracciare un evento per ogni richiesta, contenente almeno i seguenti parametri minimi:

- istante della richiesta;
- identificativo del fruitore e dell'operazione richiesta;
- tipologia di chiamata;
- esito della chiamata;
- ove applicabile, identificativo del consumatore o altro soggetto operante la richiesta comunicato dal fruitore - è cura del fruitore procedere alla codifica e l'anonimizzazione, ove necessario;
- ove applicabile, un identificativo univoco della richiesta, utile a eventuali correlazioni.

### 1.4.7 Pattern e profili di interoperabilità

Le Linee Guida individuano:

- pattern di interoperabilità, ovvero la definizione di una soluzione a una esigenza di scambio di messaggi e informazioni, declinata in una specifica tecnologia. Si suddividono in:
  - pattern di interazione, puntualizzano le modalità tecniche per implementare i modelli di scambio dei messaggi (anche detti message exchange patterns)[1], necessari all'interazione tra i sistemi informatici di erogatori e fruitori;
  - pattern di sicurezza, individuano le modalità tecniche per assicurare che i pattern di interazione rispettino specifiche esigenze di sicurezza (autenticazione e autorizzazione delle parti, confidenzialità delle comunicazioni, integrità dei messaggi scambiati, ...) negli scambi realizzati;
- profili di interoperabilità, la combinazione di più pattern per descrivere le esigenze di specifici domini di interoperabilità, quale ad esempio il non ripudio delle comunicazioni e/o dei messaggi scambiati.

**I pattern e profili di interoperabilità individuati nei Documenti operativi delle Linee Guida sono utilizzati dalle Pubbliche Amministrazioni nell'implementazione delle proprie API**.

**Le Pubbliche Amministrazioni selezionano i pattern e/o i profili di interoperabilità sulla base delle specifiche esigenze del dominio di interoperabilità a cui partecipano**.

### 1.4.8 Catalogo delle API

Le Linee Guida individuano il Catalogo delle API (in breve, Catalogo) quale componente, unica e centralizzata, che assicura alle parti coinvolte nel rapporto di erogazione e fruizione la consapevolezza sulle API disponibili, e per esse, i livelli di servizio dichiarati.

La presenza del Catalogo è funzionale a:

- facilitare l'interoperabilità tra le Pubbliche Amministrazioni e i soggetti privati interessati;
- contenere la spesa delle Pubbliche Amministrazioni, riducendo la replicazione di API;
- assicurare la dichiarazione degli SLO da parte dell'erogatore sulle singole API pubblicate;
- manifestare, ove presenti, gli impegni tra erogatori e fruitori relativi all'utilizzo delle API (SLA).

Il Catalogo, fatti salvi i principi comuni che saranno emanati dall'Agenzia per l'Italia Digitale, al fine di normalizzare le tecnologie utilizzate a livello nazionale, tiene conto della:

- Specificità dei territori e dei diversi ambiti entro cui le Pubbliche Amministrazioni operano attraverso la determinazione di specializzazioni dei contenuti del Catalogo, prevedendo aggregazioni di API a livello territoriale (ad es. su base regionale) e/o relativamente agli ambiti tematici entro cui le Pubbliche Amministrazioni operano (ad es. giustizia). Tale scelta è ulteriormente giustificata dalla opportunità di favorire momenti di aggregazione di soggetti omogenei che determini la creazione di API comuni, nonché la condivisione di metodologie per la loro progettazione e il loro sviluppo.
- Esigenza di assicurare la governance del Catalogo, quale presupposto per garantire una semantica univoca e condivisa, per evitare ridondanze e/o sovrapposizioni in termini di competenze e contenuti (de-duplicazione).
- Esigenza di assicurare una descrizione formale delle API che, attraverso l'utilizzo degli Interface Description Language (IDL) indicati, permetta di descrivere le API indipendente dal linguaggio di programmazione adottato dall'erogatore e dai fruitori.

### 1.4.9 Governance del modello

L'Agenzia per l'Italia Digitale è responsabile delle attività di governance del ModI con l'obiettivo di definire, condividere e assicurare l'aggiornamento continuo dei seguenti aspetti:

[1] Cf. https://en.wikipedia.org/wiki/Messaging_pattern

- l'insieme delle tecnologie che abilitano l'interoperabilità tra le Pubbliche Amministrazioni, cittadini e imprese;
- i pattern di interoperabilità (interazione e sicurezza);
- i profili di interoperabilità.

Il rapporto tra fruitori ed erogatori è reso esplicito tramite il Catalogo. In ottemperanza al principio once-only definito nell'EU eGovernment Action Plan 2016-2020, l'erogatore si impegna a fornire l'accesso alle proprie API a qualunque soggetto che ne abbia diritto e ne faccia richiesta. Gli erogatori DEVONO descrivere i propri e-service classificando le informazioni scambiate (ove possibile collegandole ai vocabolari controllati e a concetti semantici definiti a livello nazionale e/o internazionale), e applicando etichette che ne identifichino la categoria.

Un erogatore può delegare la registrazione degli e-service all'interno del Catalogo ad un'altra Amministrazione, denominata ente capofila, relativamente a specifici contesti territoriali e/o ambiti tematici.

In prima istanza si prevede che gli enti capofila possano essere:

- a livello territoriale, le Regioni per le Pubbliche Amministrazioni Locali del territorio di riferimento;
- a livello di ambito, le Pubbliche Amministrazioni Centrali per domini di interoperabilità costituiti per specifici ambiti tematici.

Il ModI opera in assenza di elementi centralizzati che mediano l'interazione tra erogatori e fruitori. Il Catalogo delle API permette ai soggetti pubblici e privati di conoscere gli e-service disponibili e le loro modalità di erogazione e fruizione.

L'Agenzia per l'Italia Digitale ha il ruolo di:

- recepire le esigenze di interoperabilità delle Pubbliche Amministrazioni, astrarle ed eventualmente formalizzare nuovi pattern e/o profili di interoperabilità;
- coordinare il processo di definizione dei profili e pattern di interoperabilità;
- rendere disponibile il Catalogo, attraverso un'interfaccia di accesso unica per permettere a tutti i soggetti interessati, pubblici e privati, di assumere consapevolezza degli e-service disponibili;
- richiedere l'adozione dei pattern e profili di interoperabilità per l'implementazione delle API quale condizione per l'iscrizione al Catalogo, nonché controllare con continuità il rispetto dei requisiti per l'iscrizione al catalogo.

## 1.5 Progettazione dei servizi digitali e degli e-service

Nella realizzazione delle proprie funzioni istituzionali le Pubbliche Amministrazioni implementano procedimenti amministrativi, definendo processi con l'obiettivo di realizzare servizi ai cittadini e imprese efficienti ed efficaci.

Un servizio consiste in un'attività o in una serie di attività svolte in uno scambio tra un fornitore e un cliente.

Il CAD all'art.64-bis comma 1-bis prevede che le PA, i fornitori di identità digitali e i prestatori dei servizi fiduciari qualificati, «in sede di evoluzione, progettano e sviluppano i propri sistemi e servizi in modo da garantire l'integrazione e l'interoperabilità tra i diversi sistemi e servizi (..) espongono per ogni servizio le relative interfacce applicative» (API). Lo stesso art.64-bis al comma 1-quater prevede che le PA «rendono fruibili tutti i loro servizi anche in modalità digitale».

Se l'efficacia di un servizio pubblico è determinata dal soddisfacimento della norma, l'efficienza può essere incrementata utilizzando gli strumenti propri della Information and Communication Technology (ICT) in coerenza col principio «digital first».

I processi che utilizzano gli strumenti ICT sono qui detti processi digitali.

Un e-service è un servizio erogato via Internet o attraverso una rete privata tramite un processo digitale in cui sono coinvolti erogatori e fruitori. Gli e-service sono una particolare categoria di servizi in rete basati su interfacce applicative (API).

Gli e-service di interesse delle Linee Guida sono caratterizzati da:

- il descrittore dell'e-service, un'entità che descrive l'obiettivo e le modalità per usufruire del servizio;
- le API che comprendono le modalità di accesso ad un e-service, le cui implementazioni tecnologiche si basano sulla combinazione dei pattern di interoperabilità e sicurezza o l'utilizzo di profili di interoperabilità;
- gli accordi di interoperabilità in cui sono definiti i legami tra fruitore ed erogatore.

Si assume che le Pubbliche Amministrazioni siano viste come un'entità unica da cittadini e imprese, e che, coordinandosi e collaborando, forniscano servizi.

Ne consegue che la realizzazione di un processo digitale necessita di un modello organizzativo che individui ruoli e responsabilità dei soggetti coinvolti. Nel caso in cui il servizio offerto necessiti di interazioni tra diverse Pubbliche Amministrazioni il modello organizzativo deve essere condiviso dalle stesse.

Il processo digitale determina l'insieme di e-service:

- necessari ad assicurare l'interazione tra le Pubbliche Amministrazioni (e-service di backend);
- finalizzati ad assicurare l'offerta di e-service a cittadini e imprese (e-service di frontend).

La conoscenza della disponibilità e le modalità di utilizzo degli e-service di backend e dei servizi di front-end, utilizzati direttamente da sistemi informatici, è assicurata dalla pubblicazione sul Catalogo delle API previsto nel ModI.

Tali azioni sono realizzate direttamente dagli erogatori di e-service o da un ente capofila.

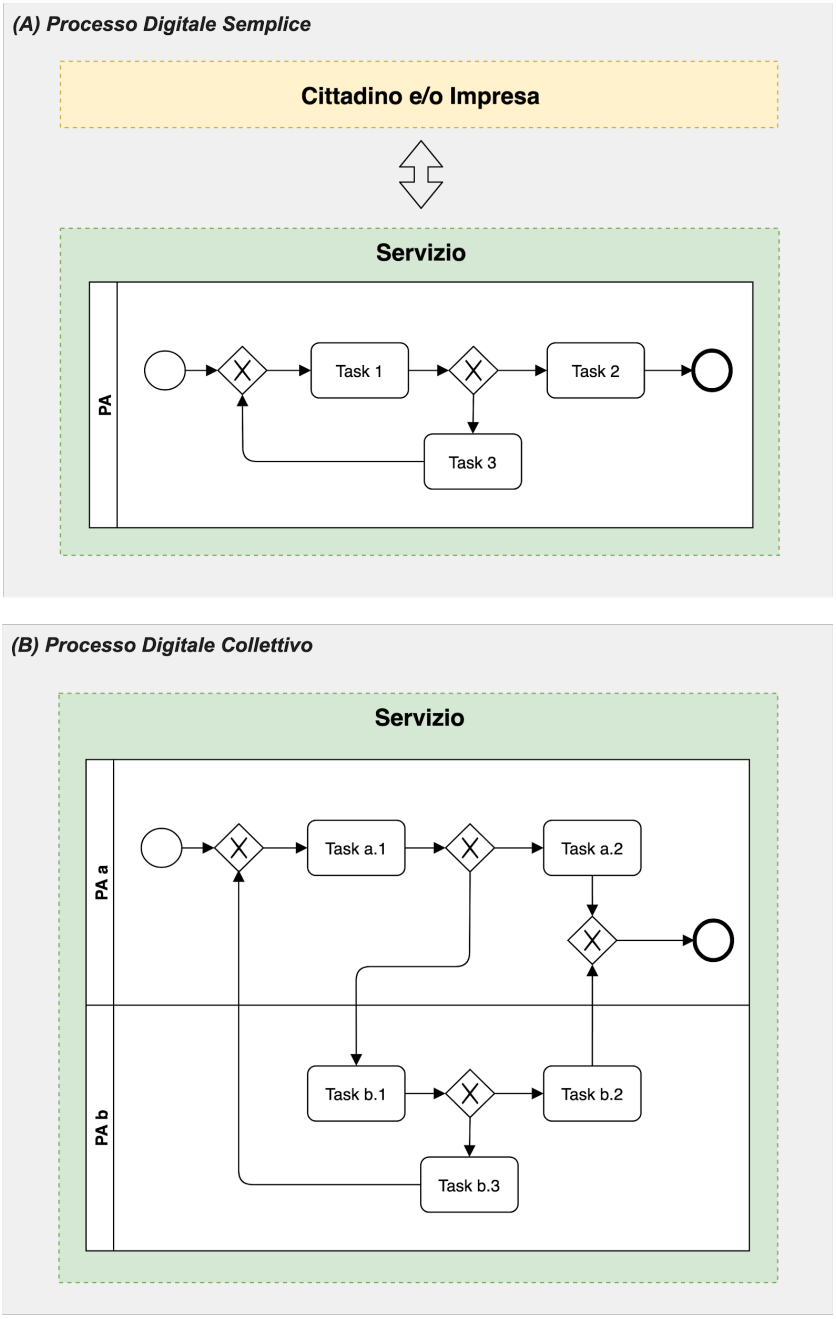


*Figura 2 - (A) Processo Digitale Semplice. (B) Processo Digitale Collettivo*

Nell'ambito del ModI, con processi digitali semplici si intendono i processi digitali che vedono coinvolta un singolo erogatore verso i cittadini e le imprese. Viceversa, con processi digitali collettivi si indicano i processi digitali che vedono coinvolte più organizzazioni.

Di seguito sono descritte delle buone pratiche per:

- supportare l'interoperabilità dei sistemi informatici;
- favorire il riutilizzo degli e-service.

Per dare seguito alla trasformazione digitale della Pubblica Amministrazione, nel ModI, la definizione dei processi digitali deve assicurare il rispetto dei principi del digital-by-default[1] e once-only[2].

[1] Il principio digital-by-default prevede che sia data priorità all'utilizzo dei servizi pubblici tramite canali digitali, preservando l'erogazione multicanale, ovvero coesistenza di canali fisici e digitali.
https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52016DC0179&from=EN

[2] Il principio del once-only prevede che cittadini ed imprese non debbano farsi carico di fornire la certificazione dei propri stati già conosciuti dalla Pubblica amministrazione, nel rispetto della norma sulla protezione dei dati personali.

La definizione di un processo digitale è avviata da una Pubblica Amministrazione ed è composta dalle fasi indicati di seguito.

**1) Individuazione delle esigenze**.

Fase in cui vengono determinate le esigenze, attraverso cui l'erogatore:

1. analizza la norma per determinare le funzioni di competenza e gli obiettivi da raggiungere. Sulla base di questi, evidenzia le caratteristiche del servizio da offrire, e individua i dati in ingresso e quelli restituiti.
2. determina se coinvolgere altre organizzazioni quali fonti autoritative dei dati necessari o soggetti preposti all'esecuzione di endoprocedimenti specifici.
3. determinare le attività necessarie ad implementare ed assicurare l'implementazione del servizio da offrire.

**2) Organizzativa**.

Fase in cui l'erogatore in base al punto 1.b coinvolge o meno altre organizzazioni:

- se non sono state individuate altre organizzazioni:
    1. l'erogatore realizza un processo digitale semplice in cui valuta la possibilità soddisfare le esigenze riusando e/o reingegnerizzando uno o più e-service già erogati, altrimenti implementa nuovi e-service progettandoli in modo da favorirne il riuso.
- se sono state individuate altre organizzazioni si realizza un processo digitale collettivo dove:
    1. l'erogatore individua le organizzazioni da coinvolgere in quanto fonti autoritative dei dati o soggetti preposti all'esecuzione di endoprocedimenti, evitando la duplicazione non necessaria dei dati;
    2. l'erogatore si confronta con esse per determinare ruoli e responsabilità;
    3. ogni organizzazione, per dare seguito alle responsabilità individuate valuta la possibilità soddisfare le esigenze riusando e/o reingegnerizzando uno o più e-service già erogati, altrimenti implementa nuovi e-service progettandoli in modo da favorirne il riuso.

**3) Semantica**.

Fase in cui le organizzazioni interessate, definiscono il modello dati da utilizzare (entità, proprietà e relativa metadatazione).

La definizione del modello dati è realizzata in coerenza con i vocabolari controllati e modelli di dati definiti a livello nazionale e internazionale.

**4) Tecnica**.

Fase in cui le organizzazioni interessate provvedono, ove necessario, a garantire la conformità al modello dati da utilizzare degli e-service individuati in fase di individuazione e/o implementazione delle API, per assicurare la disponibilità degli e-service.

**5) Governance dei servizi**.

Fase in cui ciascuna erogatore pubblica sul Catalogo le API implementate, in modo da incentivare l'utilizzo ed il riuso.

Stipula degli accordi di interoperabilità, dove sono formalizzati ruoli e responsabilità delle parti (erogatore e fruitore), per assicurare che le Pubbliche Amministrazioni interessate possano usufruire di un e-service fornito da un'altra PA attraverso una o più API.

Si rimanda ai seguenti paragrafi per dettagli.

### 1.5.1 Individuazione delle esigenze

Di seguito sono elencate una serie di raccomandazioni che le Pubbliche Amministrazioni POSSONO seguire per ottemperare a quanto richiesto.

| | |
|---|---|
| (RAC_ESIGENZE_001) | Individuare nella norma i soggetti, cittadini o imprese, e la relativa categoria di appartenenza. Ad esempio, i cittadini residenti sul territorio italiano, che possono richiedere un servizio. |
| (RAC_ESIGENZE_002) | Individuare dalla norma la risposta attesa dagli utilizzatori dei servizi informazioni di output). |
| (RAC_ESIGENZE_003) | Determinare le informazioni minime necessarie per dare seguito al procedimento amministrativo (informazioni di input). |
| (RAC_ESIGENZE_004) | Determinare se per l'implementazione del procedimento amministrativo realizzato dal servizio è richiesta l'esecuzione di endoprocedimenti da parte di altre Pubbliche Amministrazioni da coinvolgere. |
| (RAC_ESIGENZE_005) | DDeterminare tra le informazioni di input quelle già a disposizione di altre organizzazioni, quali fonti autoritative da coinvolgere, in attuazione del principio del once-only. |

### 1.5.2 Organizzativa

Al fine di realizzare e-service secondo il ModI sono di seguito elencate le regole che gli erogatori DEVONO attuare e le raccomandazioni che DOVREBBERO seguire.

Nello specifico del processo digitale, gli erogatori DEVONO:

| | |
|---|---|
| (REG_ORGANIZZ_001) | Assicurare l'erogazione di e-service in modo integrato, ponendo attenzione alla gestione di modifiche o aggiornamenti degli e-service, in quanto le stesse hanno impatto su differenti organizzazioni. |
| (REG_ORGANIZZ_002) | Definire per gli e-service e le relative API, i criteri per la gestione delle modifiche, al fine di permettere ai fruitori di assumere consapevolezza. |
| (REG_ORGANIZZ_003) | Organizzarsi per espletare i singoli servizi garantendo il livello di servizio (SLA) appropriato nel rispetto dei vincoli della norma. |
| (REG_ORGANIZZ_004) | Fornire un ambiente di test per l'esecuzione delle integrazioni da parte dei fruitori. |
| (REG_ORGANIZZ_005) | Individuare l'ufficio del Responsabile della trasformazione digitale (in seguit RTD) o in alternativa un ufficio preposto alla gestione dell'interoperabilità, quale punto di riferimento al fine di:<br>• (REG_ORGANIZZ_005.a) Efficientare l'erogazione dei servizi provvedendo alla reingegnerizzazione dei processi esistenti o definire e istituire nuovi processi in considerazione dell'utilizzo degli strumenti digitali.<br>• (REG_ORGANIZZ_005.b) Garantire il mantenimento degli SLA previsti negli Accordi di interoperabilità. |
| (REG_ORGANIZZ_006) | Pubblicare e mantenere gli Accordi di interoperabilità nella modalità individuata dal ModI direttamente o demandando l'azione ad un Ente capofila. |
| (REG_ORGANIZZ_007) | Individuare un punto di contatto tecnico per il supporto ai fruitori degli e-service (amministrazioni, cittadini, imprese). |
| (REG_ORGANIZZ_008) | Utilizzare un linguaggio formale per rappresentare il processo digitale. |

Nello specifico del processo digitale collettivo, oltre a quanto indicato in precedenza, gli erogatori e fruitori DEVONO:

| | |
|---|---|
| (REG_ORGANIZZ_009) | Individuare l'Amministrazione coordinatrice delle attività necessarie alla realizzazione del processo digitale collettivo. |
| (REG_ORGANIZZ_010) | Dare seguito, grazie agli RTD, alle necessarie azioni di condivisione degli obiettivi, definizione di calendari comuni e individuazione delle priorità, al fine di soddisfare le esigenze degli utenti finali ed erogare i servizi pubblici in modo integrato e organico. |
| (REG_ORGANIZZ_011) | Utilizzare un linguaggio formale condiviso per rappresentare il processo digitale collettivo. |

In generale le pubbliche amministrazioni DOVREBBERO:

| | |
|---|---|
| (RAC_ORGANIZZ_001) | Coinvolgere sin dalla progettazione cittadini e imprese, anche in forma associata, quando fruitori degli e-service di frontend per verificare l'usabilità del servizio. |
| (RAC_ORGANIZZ_002) | Coinvolgere sin dalla progettazione le imprese, anche in forma associata, se portatori di interessi degli e-service per verificare le caratteristiche dei servizi ed eseguire i test. |
| (RAC_ORGANIZZ_003) | Utilizzare gli standard Object Management Group (OMG), Business Process Model and Notation™ (BPMN™), Case Management Model and Notation™ (CMMN™) e Decision Model and Notation™ (DMN™) per la rappresentazione dei processi digitali, come attuazione delle G_OR008 e REG_OR010. |

### 1.5.3 Semantica

La comunicazione tra soggetti DEVE utilizzare modelli dati condivisi, in modo da razionalizzare e uniformare la rappresentazione dell'informazione quale presupposto per favorire l'interoperabilità tra soggetti differenti.

L'adozione di modelli dati condivisi permette ai soggetti che comunicano, di partire da una rappresentazione dei dati comune, evitando la proliferazione di modelli dati differenti impattano direttamente sulla modalità con cui i dati vengono resi persistenti. Ne consegue che l'adozione di modelli dati condivisi favorisce il riuso delle strutture dati, semplifica le integrazioni e assicura che gli elementi dei dati siano compresi nello stesso modo tra le parti tra loro comunicanti.

Il modello dati è un modello astratto che organizza gli elementi dei dati e standardizza il modo in cui si relazionano tra loro e con le proprietà delle entità del mondo reale[1].

La definizione dei modelli dati condivisa deriva le sue entità dalle ontologie che rappresentano la specifica formale ed esplicita di rappresentazione (concettualizzazione) condivisa di un dominio di conoscenza, definita sulla base di requisiti individuati.

I soggetti erogatori di e-service, nell'individuazione delle entità da condividere DEVONO:

| | |
|---|---|
| (REG_SEMANTICA_001) | Individuare i domini di interesse e in essi determinare le entità da rappresentare in termini di proprietà che li caratterizzano. |
| (REG_SEMANTICA_002) | Verificare la presenza delle entità per dominio tra quelli definiti a livello nazionale da AgID[3]. |

[1] Cf. https://en.wikipedia.org/wiki/Data_model
[3] Cf. https://github.com/italia/daf-ontologie-vocabolari-controllati

Si precisa che successivamente all'attuazione della REG_SEMANTICA_002 l'erogatore può trovarsi in uno dei seguenti casi:

1. tutte le entità e le relative proprietà trovano copertura;
2. almeno una delle entità non è compresa nelle rappresentazioni;
3. almeno una proprietà di un'entità presente non risulta rappresentata.

Nel caso 1), l'erogatore ha tutti gli elementi per rappresentare il proprio modello dati; viceversa, nei casi 2) e 3), la stessa amministrazione, in accordo con AgID, valutano l'opportunità di definire/aggiornare delle entità e/o proprietà a livello nazionale.

### 1.5.4 Tecnica

Gli Erogatori rendono disponibili gli e-service individuati nella fase organizzativa, assicurando:

- l'attuazione del modello dati definito nella fase semantica;
- la conformità ai pattern e/o profili di interoperabilità individuati dalle presenti Linee Guida.

L'obiettivo della fase tecnica è:

- definire le API quali descrizione di tutte le informazioni che caratterizzano un e-service per una specifica tecnologia;
- stipulare gli accordi di interoperabilità quali contratti tra due soggetti (PA, cittadino o impresa) in cui sono definite le regole per usufruire di un e-service (responsabilità, ruoli organizzativi).

Sono di seguito elencate le regole che erogatori e fruitori DEVONO attuare e le raccomandazioni che invece DOVREBBERO seguire.

Gli erogatori DEVONO:

| | |
|---|---|
| (REG_TECNICA_001) | Dare seguito all'analisi tecnica dei requisiti che, per ogni e-service, individui le caratteristiche funzionali o non funzionali per:<br>• (REG_TECNICA_001.a) Selezionare la porzione del modello dati individuato nella fase semantica.<br>• (REG_TECNICA_001.b) Individuare le tecnologie tra SOAP o REST.<br>• (REG_TECNICA_001.c) Individuare i pattern di interazione.<br>• (REG_TECNICA_001.d) Individuare i pattern di sicurezza. |
| (REG_TECNICA_002) | Implementare le API. |
| (REG_TECNICA_003) | Esporre ai fruitori una funzione utile a verificare lo stato di funzionamento delle API. |

Gli erogatori POSSONO:

| | |
|---|---|
| (RAC_TECNICA_001) | Erogare contemporaneamente gli e-service con API in tecnologia SOAP e REST. |

Gli erogatori DOVREBBERO:

| | |
|---|---|
| (RAC_TECNICA_002) | Adottare architetture capaci di adeguare l'offerta di e-service all'aumento dei fruitori, in modo da rispondere agli SLA concordati. |

La fase tecnica si conclude con l'attuazione di quanto disposto in «Governance dei servizi» relativamente alla pubblicazione degli e-service e della sottoscrizione degli accordi di interoperabilità.

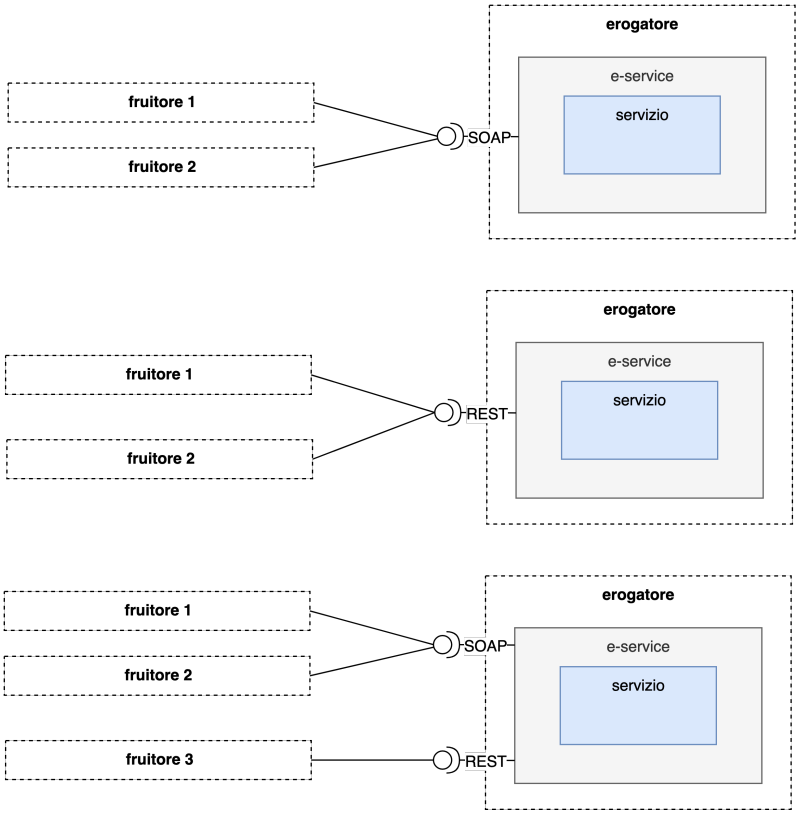


*Figura 3 - Relazione tra interfaccia di servizio - e-service – servizio*

## 1.6 Tecnologie per le API

Un aspetto che si vuole qui richiamare è la relazione tra l'interno e l'esterno del sistema informativo di una Pubblica Amministrazione, e come questo confine abbia impatti sugli e-service in termini di funzionalità e sicurezza.

Nel precedente modello di interoperabilità (il cosiddetto SPCoop del 2005) era stato definito il concetto di *dominio* di un'amministrazione, a indicare l'insieme delle risorse (dati e servizi) e delle politiche che un'amministrazione assicurava per dare seguito alla cooperazione con altre. Il confine tra i domini delle amministrazioni era istanziato fisicamente da uno specifico elemento architetturale, la Porta di Dominio (PdD).

Nel nuovo framework di interoperabilità, l'istanziazione della PdD come punto unico di accesso a un'interfaccia viene meno. Tuttavia, concettualmente il confine del dominio dell'amministrazione continua a esistere ed è importante considerarlo nella progettazione degli e-service. Gli e-service sono offerti da qualsiasi server applicativo, senza essere vincolati a essere raggiungibili attraverso un unico gateway.

Quindi, ogni server applicativo offre e-service, tuttavia è comunque significativo distinguere se gli stessi sono offerti per interoperare:

1. all'interno del dominio (da parte di client applicativi offerti dalla stessa amministrazione (ad es., un'applicazione Web o una mobile verso il back-end, integrazione tra banche dati nel back-end, ecc);
2. verso altre amministrazioni o altri soggetti con cui è stabilita una relazione di fiducia.

Le Pubbliche Amministrazioni, nella loro autonomia organizzativa possono adottare queste Linee Guida anche per i servizi sviluppati all'interno del proprio dominio. Adottare il principio API-first per i servizi interni permette di riutilizzare le componenti e di integrarle più facilmente con quelle delle altre organizzazioni in caso di necessità. Le regole tecniche inoltre permettono di applicare criteri di qualità uniformi per tutti i servizi ICT, consolidando le prassi di sicurezza, migliorando la qualità della spesa e mitigando i rischi anche interni legati alla protezione dei dati personali.

Una trattazione completa dei paradigmi per la progettazione e realizzazione delle API esula dagli scopi del presente documento. La breve discussione che viene presentata in questa sezione vuole ricordare a chi legge le differenti tecnologie per la realizzazione di API e il modello architetturale sottostante (RPC-like o resource-oriented).

Le API possono essere progettate secondo due paradigmi:

**Remote Procedure Call (RPC)-like.** In questo paradigma, un'interfaccia di servizio espone una serie di operazioni (metodi) che permettono l'invocazione delle operazioni offerte dall'interfaccia. Il significato dell'operazione è informalmente espresso dal nome dell'operazione, dal numero e dal tipo dei suoi parametri, dal tipo del valore di ritorno (il tutto viene detto tecnicamente firma o signature). Approcci formali prevedono che tale significato venga eventualmente anche descritto in opportuni documenti di accompagnamento e/o attraverso specifiche formali dell'interfaccia di servizio. Ogni operazione può quindi rappresentare, sia semplici operazioni di computazione, sia operazioni potenzialmente di lunga durata, sia operazioni di accesso a dati, ecc. Il paradigma è detto RPC-like in quanto le API ricordano le librerie di chiamata a procedura e quindi l'interfaccia, metaforicamente, è di fatto una libreria di funzioni.

**Resource-oriented.** In questo paradigma, l'interfaccia di servizio offre operazioni di creazione, lettura, aggiornamento e cancellazione di risorse. In inglese, Create, Read, Update, Delete, da cui l'acronimo da cui prende il nome il paradigma: CRUD. Una risorsa è un qualsiasi oggetto informativo che possiede uno stato. In questo paradigma, l'interfaccia di servizio è metaforicamente un accesso diretto a una base informativa, e le uniche operazioni possibili sono appunto le modifiche a tali risorse.

Parallelamente, esistono delle tecnologie con cui poter naturalmente realizzare API, che sono:

1. SOAP e il cosiddetto stack WS-*
2. lo stile architetturale REST, basato su HTTP.

**Un Web service SOAP** espone un insieme di metodi richiamabili da remoto da parte di un client. SOAP definisce una struttura dati per lo scambio di messaggi tra applicazioni, codificata in XML; di fatto SOAP utilizza HTTP come protocollo di trasporto, ma non è limitato né vincolato ad esso.

**Un Web service che sfrutta l'architettura REST** adotta il modello basato su *risorse* secondo le seguenti caratteristiche:

- individuazione delle risorse mediante il formalismo dei Uniform Resource Identifier (URI);
- operazioni sulle risorse effettuate sulla rappresentazione del loro stato;
- CRUD sulle risorse mediante HTTP utilizzando i metodi nella semantica prevista dal protocollo stesso.

L'approccio REST evidenzia le caratteristiche del Web come piattaforma leggera per l'elaborazione distribuita. Non è in prima istanza necessario aggiungere nulla a quanto è già esistente sul Web per consentire ad applicazioni remote di interagire.

Si noti che nell'applicazione pratica di REST si assiste al suo uso in modalità non del tutto canoniche. Ogni deviazione rispetto alle caratteristiche previste da REST porta alla realizzazione di architetture ibride tra il paradigma RESTful Web service e quello dei Web service RPC-like. In merito ai modelli ibridi che si possono presentare, esiste una classificazione, il cosiddetto Richardson Maturity Model 3 che prevede quattro livelli, da 0 a 3, in accordo al grado di aderenza ai dettami REST. In particolare, si possono presentare i casi seguenti:

**Livello 0** per servizi che semplicemente usano HTTP come protocollo di trasporto applicativo (tunnel HTTP). In questo caso il sistema non ha niente del modello REST.

**Livello 1** per i servizi che operano sulle risorse definite secondo la sintassi e la semantica previste per le URI, sulle quali si opera invocando delle operazioni (metodi) che agiscono su di esse.

**Livello 2** per i servizi che operano su risorse definite secondo la sintassi e la semantica previste per le URI, sulle quali si opera sulla rappresentazione del loro stato per mezzo del protocollo HTTP usando la semantica dei metodi (verbi) come previsti dal protocollo.

**Livello 3** come per il livello 2, con in aggiunta la possibile presenza di controlli ipermediali nella rappresentazione delle risorse.

Le Linee Guida accolgono l'implementazione di API REST classificabili al livello 1 del Richardson Maturity Model.

**Le Linee Guida accolgono indifferentemente SOAP e REST quali tecnologie per l'implementazione delle API. La scelta della tecnologia da utilizzare è lasciata alle Pubbliche Amministrazioni che costituiscono un dominio di interoperabilità sulla base delle specifiche esigenze.**

## 1.7 Pattern e profili di interoperabilità

Le diverse esigenze di interoperabilità delle PA e l'esperienza da loro maturata hanno evidenziano l'impossibilità di adottare un unico approccio risolutivo. Di contro, esigenze simili tra i vari domini determinano l'opportunità di definire un insieme di soluzioni tecnologiche comuni che permettano, anche attraverso la loro combinazione, di definire un quadro condiviso di regole applicate.

Le Linee Guida individuano i *pattern* e i *profili di interoperabilità* come modalità tecniche condivise secondo le quali un fruitore e un erogatore realizzano l'interoperabilità dei propri sistemi informatici, Nel dettaglio:

- i *pattern di interoperabilità* risolvono singoli requisiti che la PA deve soddisfare, e si dividono in:
  - pattern di *interazione* che descrivono il modello di comunicazione tra fruitore ed erogatore;
  - pattern di *sicurezza* che descrivono le modalità per assicurare specifiche caratteristiche di sicurezza della comunicazione;
- i *profili di interoperabilità* sono combinazioni dei pattern di interoperabilità per risolvere i casi d'uso che una PA deve soddisfare.

I pattern e i profili di interoperabilità che le PA DOVREBBERO utilizzare sono indicati nei Documenti Operativi: Pattern di integrazione, Pattern di sicurezza e Profili di interoperabilità.

Nel caso in cui le Pubbliche Amministrazioni rilevino l'impossibilità di dare seguito alle specifiche esigenze dei domini di interoperabilità che le vedano coinvolte, a causa della mancanza di pattern e/o profili di interoperabilità adeguati, o evidenzino l'opportunità di aggiornare i pattern e i profili di interoperabilità esistenti, DEVONO segnalare la circostanza ad AgID, definendo un esigenza e/o formulando una proposta.

L'AgID, stimolata dalle esigenze e dalle proposte della PA o su propria iniziativa, avvia un tavolo pubblico di concertazione per modificare o definire nuovi e/o modificare i pattern e/o i profili di interoperabilità.

I nuovi pattern e/o i profili di interoperabilità o la modifica di quelli esistenti sono formalmente introdotti nel ModI attraverso Circolari emanate dall'AgID per l'aggiornamento dei Documenti Operativi: Pattern di integrazione, Pattern di sicurezza e Profili di interoperabilità.

Il processo di mantenimento dei pattern prevede le seguenti fasi:

1. *[NUOVA ESIGENZA E PROPOSTA]* Le PA o AgID individuano nuove esigenze di interoperabilità e POSSONO proporre nuovi pattern come soluzione;

2. *[CONCERTAZIONE]* AgID apre il tavolo pubblico, in modo da acquisire eventuali proposte alternative per soddisfare le nuove esigenze evidenziate;

3. *[ADOZIONE]* AgID, valutando le proposte pervenute, individua i nuovi pattern che rispondono alle esigenze e aggiorna i Documenti Operativi.

Il processo di mantenimento dei pattern è realizzato da cicli successivi di applicazione delle fasi individuate. La pianificazione dei cicli di proposta assicura la raccolte delle esigenze dal 1 gennaio al 31 dicembre di ogni anno.

- (i)° ciclo [1 gennaio anno corrente - 31 ottobre anno corrente ]
  1. [NUOVA ESIGENZA E PROPOSTA] da 1° gennaio anno corrente - 30 giugno anno corrente
  2. [CONCERTAZIONE] dal 1 maggio anno corrente - 31 agosto anno corrente
  3. [ADOZIONE] dal 1° settembre anno corrente - 31 ottobre anno corrente
- (i+1)° ciclo [1 luglio anno corrente - 30 aprile anno successivo]
  1. [NUOVA ESIGENZA E PROPOSTA] da 1° luglio anno corrente - 31 dicembre anno corrente
  2. [CONCERTAZIONE] dal 1 novembre anno corrente - 31 febbraio anno successivo
  3. [ADOZIONE] dal 1° marzo anno successivo - 30 aprile anno successivo

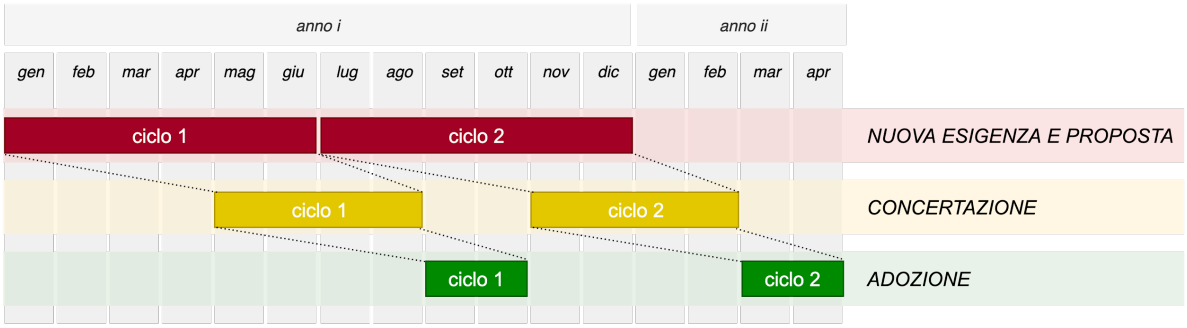


*Figura 4 - Rappresentazione del processo di gestione dei pattern*

Si rimanda ai seguenti paragragi per dettagli.

### 1.7.1 Nuove esigenze e proposta

Un erogatore o fruitore - di seguito richiedente, partendo da un caso d'uso determina un profilo applicativo, costituito dai *pattern di interoperabilità* e *profili di interoperabilità*, di sicurezza disponibili nel ModI.

Nel caso in cui il richiedente non riesca a definire un profilo applicativo per la mancanza di uno o più *pattern di interoperabilità* o *profili di interoperabilità*, DEVE dichiarare la necessità di introdurre un nuovo *pattern di interoperabilità* e *profili di interoperabilità* avviando una «nuova esigenza» e inviando una comunicazione attraverso gli strumenti messi a disposizione da AgID.

Un richiedente, secondo le proprie disponibilità e necessità, PUÒ definire un *pattern di interoperabilità* o *profili di interoperabilità* quale proposta che risolva la «nuova esigenza» e la trasmetterla ad AgiD come prima ipotesi risolutiva.

AgID, entro 15 giorni dall'invio della richiesta di una «nuova esigenza» valuta se sussistono le condizioni per avviare la fase di concertazione. In caso contrario DEVE comunicare al richiedente le motivazioni del rifiuto.

Il richiedente, trascorsi i 15 giorni indicati in precedenza, PUÒ preventivamente utilizzare il pattern proposto. A conclusione della fase di concertazione e adozione, se AgID non rileva problemi in merito all'utilizzo del *pattern di interoperabilità* o *profili di interoperabilità* proposto, il richiedente NON DEVE uniformare quanto implementato.

### 1.7.2 Concertazione ed Adozione

AgID avvia la fase di concertazione rendendo pubblica l'esigenza espressa dal richiedente e, se presente, anche il primo *pattern di interoperabilità* o *profili di interoperabilità* proposto.

Nella fase di concertazione viene messa a disposizione una piattaforma dove è possibile presentare proposte di *pattern di interoperabilità* o *profili di interoperabilità* che rispondano all'esigenza dichiarata dal richiedente e/o evidenziare carenze nei *pattern di interoperabilità* o *profili di interoperabilità* fino a quel momento proposti.

Nella fase di adozione, AgiD, considerando le eventuali proposte pervenute nella fase di concertazione, adotta i *pattern di interoperabilità* o *profili di interoperabilità* che faranno parte integrante del ModI attraverso le Circolari

emanate dall'AgID per l'aggiornamento dei Documenti Operativi: Pattern di integrazione, Pattern di sicurezza e Profili di interoperabilità.

## 1.8 Governance dei servizi

Il ModI identifica negli e-service gli elementi atomici per dare seguito allo scambio di dati e informazioni tra le Pubbliche Amministrazioni, le imprese e i cittadini.

Nel ModI, il Catalogo delle API è la componente che gestisce il ciclo di vita della pubblicazione, consultazione e fruizione degli e-service, assicurando la specifica delle API che essi implementano.

Il Catalogo delle API implementa quanto previsto dal CAD all'art.73, comma 3-ter, lettera c).

Gli e-service registrati nel Catalogo, per normalizzare l'integrazione tra i sistemi informatici, utilizzano le tecnologie, i pattern e i profili individuati nelle Linee Guida.

Relativamente a un'e-service, indichiamo con:

- *erogatore* la PA che lo rende disponibile;
- *fruitore* i soggetti (PA, imprese e cittadini) che lo utilizzano;
- *ente capofila* le PA responsabili, su delega di altre PA, delle attività di gestione sul Catalogo.

Il Catalogo realizza un punto centrale, unico per la pubblicazione, di consultazione e fruizione degli e-service, permettendo:

- alle PA erogatrici o alle PA ente capofila di pubblicare e gestire gli e-service;
- ai soggetti fruitori (PA, imprese e cittadini) di consultare gli e-service disponibili;
- alle PA erogatrici e ai soggetti fruitori di registrare, ove necessario, gli elementi tecnici convenuti per utilizzare gli e-service.

Nell'ambito di relazioni tra differenti soggetti (erogatori e fruitori) determinata dall'utilizzo di e-service è necessario concordare l'efficienza di questi ultimi. La misura dell'efficienza avviene attraverso l'individuazione dei seguenti elementi:

- *Service-Level Indicator* (di seguito SLI), ovvero metriche atte a misurare l'efficienza dei servizi individuati dall'erogatore;
- *Service-Level Objective* (di seguito SLO), ovvero gli obiettivi degli SLI per i servizi definiti dall'erogatore;
- *Service-Level Agreement* (di seguito SLA), ovvero accordi sul livello di servizio frutto della contrattazione tra erogatore e fruitore.

Un e-service è rappresentato nel Catalogo dalle seguenti entità:

- *i descrittori dell'e-service*, che definiscono le informazioni relative alla natura dell'e-service e DEVONO essere pubblicati dalle PA erogatrici.
- *le API*, che definiscono le modalità tecniche per usufruire dell'e-service e DEVONO essere pubblicate e gestite dalle PA erogatrici.
- *gli accordi di interoperabilità*, che permettono alle PA erogatrici e ai fruitori di dichiarare la costituzione di una relazione di utilizzo, comprensiva degli SLA condivisi quali insieme delle responsabilità e degli obblighi nell'utilizzo delle API, e DEVONO essere confermati dalla PA erogatrici e dai fruitori.

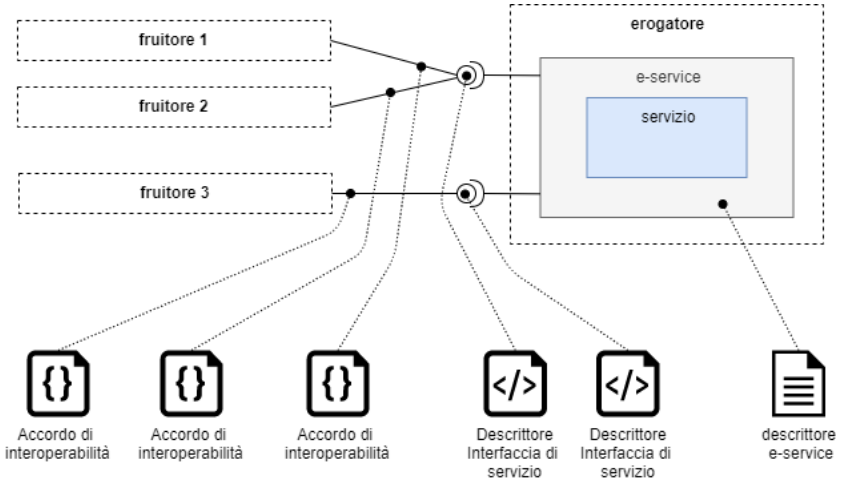


*Figura 5 - Entità gestite dal Catalogo*

Si precisa che gli erogatori, nell'ambito della definizione dei un'e-service, individuano le API da loro implementate e, allo stesso tempo, le API che i fruitori devono utilizzare per usufruire dell'e-service.

Il Catalogo supporta una governance distribuita degli e-service, assicurando i seguenti obiettivi:

- favorire l'uso degli e-service attraverso la loro pubblicazione;
- agevolare l'accentramento della gestione del ciclo di vita delle API;
- mitigare la creazione di API ridondanti;
- regolamentare le relazioni tra erogatore e fruitore, ove necessario, dichiarando i relativi SLA.

Il Catalogo, unico e centralizzato, è reso disponibile alle PA, alle imprese e ai cittadini dalla piattaforma realizzata da AgID a tale scopo.

CAPITOLO 2

# Documento operativo - Pattern di interazione

## 2.1 Introduzione

Il pattern di interazione definisce le modalità secondo le quali un fruitore ed un erogatore possono interagire. L'interazione avviene definendo le API.

I pattern di interazione quali riferimenti concettuali coniugano, fatto salvo per l'accesso CRUD, i message exchange pattern (MEP) per le tecnologie SOAP e REST proposte dal ModI.

Di seguito l'attenzione è rivolta agli aspetti funzionali rimandando per gli aspetti di sicurezza all'apposito Documento Operativo - Pattern di sicurezza.

Il Documento operativo descrive i pattern di interazione individuati da AgID che gli erogatori DEVONO utilizzare per soddisfare le necessità individuate dai requisiti funzionali e non funzionali delle specifiche interazioni con i propri fruitori.

Data la variabilità nel tempo delle esigenze delle amministrazioni e delle tecnologie abilitanti, nonché considerata la natura incrementale del ModI, l'elenco dei pattern di interazione non è da intendersi esaustivo. Nel caso in cui un'amministrazione abbia esigenze non ricoperte nei seguenti pattern di interazione DEVE informare l'Agenzia per l'Italia Digitale, nei modi indicati nel capitolo 7 «Pattern e profili di interoperabilità delle Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni».

## 2.2 Ambito di applicazione

Il presente Documento operativo, allegato delle Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni, riporta i pattern di interazione definiti nel ModI.

### 2.2.1 Soggetti destinatari

Il Documento Operativo è destinato ai soggetti di cui all'articolo 2, comma 2 del CAD, così come indicato dall'articolo 75 dello stesso. I destinatari la attuano nella realizzazione dei propri sistemi informatici che fruiscono o erogano dati e/o servizi digitali di/ad altri soggetti.

Il Documento Operativo è rivolto ai soggetti privati che devono interoperare con la Pubblica Amministrazione per erogare o fruire di dati e servizi tramite sistemi informatici.

## 2.3 Riferimenti e sigle

### 2.3.1 Note di lettura del documento

Conformemente alle norme ISO/IEC Directives, Part 3 per la stesura dei documenti tecnici le presenti Linee Guida utilizzano le parole chiave «DEVE», «DEVONO», «NON DEVE», «NON DEVONO», «DOVREBBE», «NON DOVREBBE», «PUÒ» e «OPZIONALE», la cui interpretazione è descritta di seguito.

- **DEVE** o **DEVONO**, indicano un requisito obbligatorio per rispettare le Linee Guida;
- **NON DEVE** o **NON DEVONO**, indicano un assoluto divieto delle specifiche;
- **DOVREBBE** o **NON DOVREBBE**, indicano che le implicazioni devono essere comprese e attentamente pesate prima di scegliere approcci alternativi;
- **PUÒ** o **POSSONO** o l'aggettivo **OPZIONALE**, indica che il lettore può scegliere di applicare o meno senza alcun tipo di implicazione o restrizione la specifica.

### 2.3.2 Riferimenti Normativi

Sono riportati di seguito gli atti normativi di riferimento del presente documento.

Tabella 2.1: Riferimenti Normativi

| | |
|---|---|
| [CAD] | decreto legislativo 7 marzo 2005, n. 82 recante «Codice dell'Amministrazione Digitale»; NOTA – Il D. Lgs. 82/2005 è noto anche con l'abbreviazione «CAD» |
| [EIF] | European Interoperability Framework (EIF) |
| [CE 1205/2008] | Regolamento (CE) n. 1205/2008 della Commissione del 3 dicembre 2008 recante attuazione della direttiva 2007/2/CE del Parlamento europeo e del Consiglio per quanto riguarda i metadati |
| [D.lgs. 196/2003] | Codice in materia di protezione dei dati personali |
| [UE 2016/679] | Regolamento (UE) n. 2016/679 del 27 aprile 2016 relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali (in breve GDPR) |
| [UE 910/2014] | Regolamento (UE) n. 910/2014 del 23 luglio 2014 in materia di identificazione elettronica e servizi fiduciari per le transazioni elettroniche nel mercato interno (in breve eIDAS) |
| [DUE 2019/1024] | Direttiva (UE) 2019/1024 del Parlamento Europeo e del Consiglio del 20 giugno 2019 relativa all'apertura dei dati e al riutilizzo dell'informazione del settore pubblico |

### 2.3.3 Standard di riferimento

Sono riportati di seguito gli standard tecnici indispensabili per l'applicazione del presente documento.

Tabella 2.2: Standard di riferimento

| | |
|---|---|
| [RFC7396] | JSON Merge Patch |
| [RFC7389] | Separazione di controllo e piano utente per Proxy Mobile IPv6 |
| [RFC5789] | Metodo PATCH per HTTP |

### 2.3.4 Termini e definizioni

Tabella 2.3: Termini e definizioni

| | |
|---|---|
| [AgID] | Agenzia per l'Italia Digitale |
| [CAD] | Codice Amministrazione Digitale, D.lgs. 7 marzo 2005, n. 82 |
| [PA] | Pubblica Amministrazione |
| [UML] | Linguaggio di modellazione unificato (Unified Modeling Language) |
| [RPC] | Remote procedure call |
| [SOAP] | Simple Object Access Protocol |
| [REST] | Representational State Transfer |

## 2.4 Principi generali

### 2.4.1 Interazione bloccante e non bloccante

Nell'*interazione bloccante* un fruitore effettua una chiamata al servente ed attende una risposta prima di continuare l'esecuzione. La chiamata codifica in modo opportuno la richiesta di servizio, anche attraverso il passaggio di dati (sia in input alla chiamata che in output nella risposta).

Nell'*interazione non bloccante*, invece, il fruitore invia un messaggio, ma non si blocca in attesa di alcuna risposta (se non una notifica di presa in carico). Il messaggio contiene in modo opportuno la richiesta ed eventuali dati di input. Talvolta il messaggio, proprio ad indicare il fatto che codifica la richiesta e le informazioni necessarie a soddisfarla, viene indicato come documento. La risposta da parte del servente, nei casi in cui ci sia, può apparire significativamente più tardi, ove «significativamente» va interpretato rispetto al tempo di computazione proprio dell'interazione. Anche la risposta del servente viene inviata tramite un messaggio.

Con abuso di nomenclatura, la comunicazione bloccante talvolta viene detta sincrona, ad indicare che client e servente si sono sincronizzati (attesa di uno da parte dell'altro); quella non bloccante viene detta asincrona, proprio a significare l'asincronicità che vi è tra l'invio di un messaggio e la risposta al messaggio stesso.

*Alonso, G., Casati, F., Kuno, H., Machiraju, V. (2004). Web Services. Concepts, Architectures and Applications. Springer*

### 2.4.2 Remote Procedure Call

Una Remote Procedure Call (chiamata a procedura remota, RPC) consiste nell'attivazione, da parte di un programma, di una procedura o subroutine attivata su un elaboratore potenzialmente diverso da quello sul quale il programma viene eseguito. Quindi l'RPC consente a un programma di eseguire subroutine «a distanza» su elaboratori remoti, accessibili attraverso una rete. Essenziale al concetto di RPC è l'idea di trasparenza: la chiamata di procedura remota deve essere infatti eseguita in modo il più possibile analogo a quello della chiamata di procedura locale; i dettagli della comunicazione su rete devono essere «nascosti» (resi trasparenti) all'utilizzatore del meccanismo. I problemi di comunicazione possono essere gestiti anch'essi in modo trasparente, oppure generare errori o eccezioni. Se il linguaggio è orientato agli oggetti, l'invocazione della procedura remote è in realtà l'invocazione di un metodo su un oggetto remoto, e si parla di Remote Method Invocation - RMI.

RPC/RMI è il meccanismo base con cui realizzare una interazione bloccante.

### 2.4.3 Idempotenza

Il concetto di idempotenza in matematica è una proprietà delle funzioni per la quale applicando molteplici volte una funzione data, il risultato ottenuto è uguale a quello derivante dall'applicazione della funzione un'unica volta (es. gli operatori di intersezione o unione). Applicato alle interfacce di servizio, questo concetto indica il fatto che una operazione, se eseguita più volte non comporta un risultato diverso sul sistema erogatore. Il caso classico è quello in cui si ha una operazione di creazione. Nel caso di errore di rete, l'operazione potrebbe essere eseguita senza che il fruitore riceva un messaggio di risposta. In questo caso il fruitore può ritentare la stessa operazione,

ma il risultato in questo caso non deve essere la creazione di una seconda risorsa. L'erogatore dell'interfaccia di servizio deve invece riconoscere la duplicazione della richiesta ed evitare comportamenti indesiderati. Questo comportamento è solitamente ottenuto tramite l'utilizzo di CorrelationID oppure tramite il confronto dati basato su dati che fungono da chiave.

## 2.5 Pattern bloccanti

Lo sviluppo di una interfaccia bloccante di tipo RPC-like si richiede nei casi in cui:

- L'esecuzione del metodo è poco onerosa computazionalmente e può essere portata immediatamente a termine dall'erogatore.
- Il contesto rende complessa l'implementazione delle modalità non bloccanti. Ad esempio, non è possibile per il fruitore esporre una propria interfaccia di servizio ed il fruitore non può farsi carico di mantenere il contesto necessario ad effettuare attesa attiva.

*Figura 1 - Interazione bloccante RPC*

In questo pattern si ha una risposta da parte dell'erogatore contestuale alla richiesta del fruitore. La figura mostra lo schema di questa interazione.

Nel seguito, per gli esempi proposti si fa riferimento ad un'amministrazione denominata **ente.example** che offre un'interfaccia di servizio secondo le due diverse tecnologie REST o SOAP. Inoltre, per quanto riguarda i pattern relativi a chiamata a procedura remota (bloccante e non bloccante), si farà riferimento ad un metodo **M** che accetta come parametri:

- **a**, un oggetto contenente a sua volta un array **a1** di interi ed una stringa **a2**;
- **b**, una stringa;

e restituisce una stringa **c** come output.

Le implementazioni degli esempi sono corredate dalla specifica dell'interfaccia e da uno scambio di messaggi esemplificativo.

Di seguito le indicazioni per le tecnologie accolte dal ModI.

L'AgID assicura l'aggiornamento degli stessi per soddisfare le esigenze espresse dalle PA.

### 2.5.1 [BLOCK_REST] Blocking REST

Nel caso di implementazione tramite tecnologia REST, DEVONO essere seguite almeno le seguenti indicazioni:

- La specifica dell'interfaccia DEVE dichiarare tutti i codici di stato HTTP previsti dall'interfaccia, con il relativo schema della risposta, oltre che ad eventuali header HTTP restituiti;
- La specifica dell'interfaccia DEVE dichiarare lo schema della richiesta insieme ad eventuali header HTTP richiesti;
- Al passo (1) il fruitore DEVE utilizzare come verbo HTTP per l'esecuzione della chiamata a procedura il verbo **HTTP method POST** [9] su un URL contenente gli ID interessati ed il nome del metodo;
- Al passo (2) l'erogatore DEVE utilizzare HTTP status 2xx a meno che non si verifichino errori.

#### 2.5.1.1 Regole di processamento

Al termine del processamento della richiesta, l'erogatore DEVE fare uso dei codici di stato HTTP rispettando la semantica. In particolare, al ricevimento della richiesta da parte del fruitore, l'erogatore:

- DEVE verificare la validità sintattica e semantica dei dati in ingresso;

[9] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

- DEVE, in caso di dati errati, restituire **HTTP status 400 Bad Request**[10] fornendo nel body di risposta dettagli circa l'errore;
- DOVREBBE, in caso di rappresentazione semanticamente non corretta, ritornare **HTTP status 422 Unprocessable Entity**[11];
- DOVREBBE, se qualcuno degli ID nel path o nel body non esiste, restituire **HTTP status 404 Not Found**[12], indicando nel body di risposta quale degli ID è mancante;
- PUÒ, se ipotizza che la richiesta sia malevola, ritornare HTTP status 400 Bad Request o **HTTP status 404 Not Found**[13]
- DEVE, in caso di errori non dipendenti dalla richiesta, restituire HTTP status 5xx rispettando la semantica degli stessi;
- DEVE, in caso di successo, restituire HTTP status 2xx inviando il risultato dell'operazione nel payload body.

NB: I messaggi di errore devono essere utili al client ma NON DEVONO rivelare dettagli tecnici e/o informazioni riservate.

#### 2.5.1.2 Esempio

Specifica Servizio

https://api.ente.example/rest/nome-api/v1/RESTblocking.yaml

```yaml
openapi: 3.0.1
info:
 title: RESTblocking
 version: "1.0"
 description: |-
   Questo file descrive semplicemente i metodi di un'API
   e non indica tutte le informazioni di metadatazione che
   normalmente vanno inserite.
 license:
   name: Apache 2.0 License
   url: http://www.apache.org/licenses/LICENSE-2.0.html
paths:
 /resources/{id_resource}/M:
   post:
     description: Esegue M
     operationId: M
     parameters:
     - name: id_resource
       in: path
       required: true
       schema:
         type: integer
         format: int32
     requestBody:
       content:
         application/json:
           schema:
             $ref: '#/components/schemas/MType'
     responses:
       200:
         description: Esecuzione di M avvenuta con successo
         content:
```

(continues on next page)

[10] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400
[11] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/422
[12] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[13] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404

(continua dalla pagina precedente)

```
            application/json:
              schema:
                $ref: '#/components/schemas/MResponseType'
        400:
          description: Richiesta non valida
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorMessage'
        404:
          description: Identificativo non trovato
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorMessage'
        default:
          description: |-
            Errore inatteso. Non ritornare informazioni
            sulla logica interna e/o non pertinenti all'interfaccia.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorMessage'

components:
 schemas:
   MType:
     type: object
     properties:
       a:
         $ref: '#/components/schemas/AComplexType'
       b:
         type: string
   MResponseType:
     type: object
     properties:
       c:
         type: string
   AComplexType:
     type: object
     properties:
       a1s:
         type: array
         items:
           type: integer
           format: int32
       a2:
         type: string
   ErrorMessage:
      type: object
      properties:
        detail:
          description: |
            A human readable explanation specific to this occurrence of the
            problem.
          type: string
        instance:
          description: |
            An absolute URI that identifies the specific occurrence of the problem.
            It may or may not yield further information if dereferenced.
          format: uri
```

(continues on next page)

(continua dalla pagina precedente)

```
        type: string
      status:
        description: |
          The HTTP status code generated by the origin server for this occurrence
          of the problem.
        exclusiveMaximum: true
        format: int32
        maximum: 600
        minimum: 100
        type: integer
      title:
        description: |
          A short, summary of the problem type. Written in english and readable
          for engineers (usually not suited for non technical stakeholders and
          not localized); example: Service Unavailable
        type: string
      type:
        default: about:blank
        description: |
          An absolute URI that identifies the problem type.  When dereferenced,
          it SHOULD provide human-readable documentation for the problem type
          (e.g., using HTML).
        format: uri
        type: string
```

Di seguito un esempio di chiamata al metodo **M**.

Endpoint

https://api.ente.example/rest/nome-api/v1/resources/1234/M

1. Request

```
POST /rest/nome-api/v1/resources/1234/M HTTP/1.1
Host: api.ente.example
Content-Type: application/json

{
  "a": {
    "a1s": [ 1, 2 ],
    "a2": "RGFuJ3MgVG9vbHMgYXJlIGNvb2wh"
  },
  "b": "Stringa di esempio"
}
```

2. Response **HTTP status 200 OK**[14]

```
HTTP/1.1 200 OK
Content-Type: application/json

{"c": "risultato"}
```

2. Response **HTTP status 500 Internal Server Error**[15]

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/problem+json

{
    "type": "https://apidoc.ente.example/probs/operation-too-long",
    "status": 500,
```

(continues on next page)

[14] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200
[15] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500

(continua dalla pagina precedente)

```
    "title": "L'operazione è durata troppo.",
    "detail": "Il sistema non e' riuscito a completare in tempo l'operazione␣
↪prevista."
}
```

2. Response **HTTP status 400 Bad Request**[16]

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
    "type": "https://apidoc.ente.example/probs/invalid-a",
    "status": 400,
    "title": "L'attributo \`b\` ha un valore non valido.",
    "detail": "L'attributo \`b\` dev'essere una stringa di lunghezza   inferiore a␣
↪32 caratteri."
}
```

2. Response **HTTP status 404 Not Found**[17]

```
HTTP/1.1 404 Not Found
Content-Type: application/problem+json

{
    "status": 404,
    "title": "Risorsa non trovata."
}
```

### 2.5.2 [BLOCK_SOAP] Blocking SOAP

Se il pattern viene implementato con tecnologia SOAP, a differenza del caso REST, il metodo invocato non è specificato nell'endpoint chiamato, poiché viene identificato all'interno del body. Inoltre tutti gli ID coinvolti DEVONO essere riportati all'interno del body. DEVE essere rispettata le seguente regola:

- la specifica dell'interfaccia dell'erogatore DEVE dichiarare tutti i metodi esposti con relativi schemi dei messaggi di richiesta e di ritorno. Inoltre le interfacce devono specificare eventuali header SOAP richiesti.

#### 2.5.2.1 Regole di processamento

Nel caso di errore il WS-I Basic Profile Version 2.0 richiede l'utilizzo del meccanismo della SOAP fault per descrivere i dettagli dell'errore. Al ricevimento della richiesta da parte del fruitore, l'erogatore:

- DEVE verificare la validità sintattica dei dati in ingresso. In caso di dati errati DEVE restituire **HTTP status 500 Internal Server Error**[18] fornendo dettagli circa l'errore utilizzando il meccanismo della SOAP fault;
- Se l'erogatore ipotizza che la richiesta sia malevola PUÒ ritornare **HTTP status 400 Bad Request**[19] o **HTTP status 404 Not Found**[20]
- In caso di errori non dipendenti dal fruitore, DEVE restituire i codici HTTP 5XX rispettando la semantica degli stessi o restituire il codice **HTTP status 500 Internal Server Error**[21] indicando il motivo dell'errore nella SOAP fault;

[16] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400
[17] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[18] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500
[19] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400
[20] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[21] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500

- In caso di successo restituire **HTTP status 200 OK**[22], riempiendo il body di risposta con il risultato dell'operazione.

#### 2.5.2.2 Esempio

Specifica Servizio

https://api.ente.example/soap/nome-api/v1?wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://ente.example/nome-api"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/" name=
↪"SOAPBlockingImplService" targetNamespace="http://ente.example/nome-api">
  <wsdl:types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://ente.example/nome-api" attributeFormDefault="unqualified"␣
↪elementFormDefault="unqualified" targetNamespace="http://ente.example/nome-api">
      <xs:element name="MRequest" type="tns:MRequest"/>
      <xs:element name="MRequestResponse" type="tns:MRequestResponse"/>
      <xs:complexType name="MRequest">
        <xs:sequence>
          <xs:element minOccurs="0" name="M" type="tns:mType"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="mType">
        <xs:sequence>
          <xs:element minOccurs="0" name="oId" type="xs:int"/>
          <xs:element minOccurs="0" name="a" type="tns:aComplexType"/>
          <xs:element minOccurs="0" name="b" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="aComplexType">
        <xs:sequence>
          <xs:element minOccurs="0" name="a1s" type="tns:a1ComplexType"/>
          <xs:element minOccurs="0" name="a2" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="a1ComplexType">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="a1" nillable="true
↪" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="MRequestResponse">
        <xs:sequence>
          <xs:element minOccurs="0" name="return" type="tns:mResponseType"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="mResponseType">
        <xs:sequence>
          <xs:element minOccurs="0" name="c" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="errorMessageFault">
        <xs:sequence>
          <xs:element minOccurs="0" name="customFaultCode" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="ErrorMessageFault" nillable="true" type=
↪"tns:errorMessageFault"/>
```

(continues on next page)

[22] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

(continua dalla pagina precedente)

```
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="MRequest">
    <wsdl:part element="tns:MRequest" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="MRequestResponse">
    <wsdl:part element="tns:MRequestResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="ErrorMessageException">
    <wsdl:part element="tns:ErrorMessageFault" name="ErrorMessageException"/>
  </wsdl:message>
  <wsdl:portType name="SOAPBlockingImpl">
    <wsdl:operation name="MRequest">
      <wsdl:input message="tns:MRequest" name="MRequest"/>
      <wsdl:output message="tns:MRequestResponse" name="MRequestResponse"/>
      <wsdl:fault message="tns:ErrorMessageException" name="ErrorMessageException"/
↪>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SOAPBlockingImplServiceSoapBinding" type=
↪"tns:SOAPBlockingImpl">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http
↪"/>
    <wsdl:operation name="MRequest">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="MRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="MRequestResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ErrorMessageException">
        <soap:fault name="ErrorMessageException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SOAPBlockingImplService">
    <wsdl:port binding="tns:SOAPBlockingImplServiceSoapBinding" name=
↪"SOAPBlockingImplPort">
      <soap:address location="https://api.ente.example/soap/nome-api/v1"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A seguire un esempio di chiamata al metodo **M**.

Endpoint

https://api.ente.example/soap/nome-api/v1

Method M

1. Request Body

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Header>
        <!--Autenticazione-->
    </soap:Header>
    <soap:Body>
        <m:MRequest>
            <M>
                <oId>1234</oId>
```

(continues on next page)

(continua dalla pagina precedente)

```
                <a>
                    <a1s>
                        <a1>1</a1>
                        ...
                        <a1>2</a1>
                    </a1s>
                    <a2>RGFuJ3MgVG9vbHMgYXJlIGNvb2wh</a2>
                </a>
                <b>Stringa di esempio</b>
            </M>
        </m:MRequest>
    </soap:Body>
</soap:Envelope>
```

2. Response Body (HTTP status code 200 OK)

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">

    <soap:Body>
        <m:MRequestResponse>
            <return>
                <m:c>OK</m:c>
            </return>
        </m:MRequestResponse>
    </soap:Body>

</soap:Envelope>
```

2. Response Body (HTTP status code 500 Internal Server Error)

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Body>
        <soap:Fault>
            <soap:Code>
                <soap:Value>soap:Receiver</soap:Value>
            </soap:Code>
            <soap:Reason>
                <soap:Text xml:lang="en">Error</soap:Text>
            </soap:Reason>
            <soap:Detail>
                <m:ErrorMessageFault>
                    <customFaultCode>1234</customFaultCode>
                </m:ErrorMessageFault>
            </soap:Detail>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

## 2.6 Pattern non bloccanti

I pattern di interazione non bloccanti di tipo RPC-like sono quelli in cui un fruitore invia una richiesta e questa viene solo presa in carico immediatamente, mentre il suo soddisfacimento può avvenire in maniera differita. Gli approcci non bloccanti vengono utilizzati nei casi in cui i tempi per l'erogazione di una risposta da parte dell'erogatore sono lunghi perché

- la richiesta è onerosa in termini temporali;
- l'erogatore non può farsi immediatamente carico dell'erogazione del servizio.

Al fine di collegare le richieste con le risposte si farà uso, sia nelle implementazioni SOAP che in quelle REST, di meta-informazioni specifiche (quali il CorrelationID e l'endpoint per le callback). Queste sono estranee solitamente alla business logic del servizio, ma è necessario definirle a livello di API ai fini dell'interoperabilità. A tal fine verranno definiti header (HTTP nel caso REST ed envelope nel caso SOAP) utili a contenere queste informazioni. In alcuni casi, una API viene creata al fine di automatizzare o semplificare un servizio già offerto dalla pubblica amministrazione. In una moltitudine di casi questi servizi sono asincroni (non bloccanti) per natura, e consistono di richieste a cui vengono allegati degli identificativi (es. numeri di protocollo) che accompagnano la richiesta. In questi casi, il CorrelationID può essere sostituito da questi identificativi già previsti dal servizio.

Nel seguito, per gli esempi proposti si fa riferimento ad un'amministrazione denominata **ente.example** che offre un'interfaccia di servizio secondo le due diverse tecnologie REST o SOAP. Inoltre, per quanto riguarda i pattern relativi a chiamata a procedura remota (bloccante e non bloccante), si farà riferimento ad un metodo **M** che accetta come parametri:

- **a**, un oggetto contenente a sua volta un array **a1** di interi ed una stringa **a2**;
- **b**, una stringa;

e restituisce una stringa **c** come output.

Le implementazioni degli esempi sono corredate dalla specifica dell'interfaccia e da uno scambio di messaggi esemplificativo.

Di seguito le indicazioni per le tecnologie accolte dal ModI.

L'AgID assicura l'aggiornamento degli stessi per soddisfare le esigenze espresse dalle PA.

### 2.6.1 Pattern non bloccante RPC PUSH (basato su callback)

Questo caso particolare, denominato RPC PUSH, è utilizzabile nel caso in cui il fruitore abbia a sua volta la possibilità di esporre una interfaccia di servizio per la ricezione delle risposte.

*Figura 2 - Interazione non bloccante tramite callback*

In questo pattern (vedi figura), la richiesta del fruitore contiene un riferimento al servizio da chiamare al momento della risposta. Si suppone infatti che i fruitori espongano a loro volta delle interfacce con segnatura standard. Al momento del ricevimento della richiesta (passo 1), l'erogatore risponde (passo 2) riconoscendo l'avvenuta ricezione (acknowledgement o in breve ack) ed indica un CorrelationID (ID di correlazione), generato lato erogatore, che permetta al fruitore, una volta ricevuta la risposta, di accoppiarla alla richiesta originale. Quando il processamento è terminato infatti, l'erogatore (passo 3) chiama il fruitore (invertendo quindi i ruoli chiamato/chiamante) riportando l'esito ed indicando il CorrelationID. Il fruitore riconosce (sempre mediante un messaggio di ack) la ricezione della risposta (passo 4).

#### 2.6.1.1 [NONBLOCK_PUSH_REST] Not Blocking Push REST

Nel caso in cui il pattern venga implementato con tecnologia REST, DEVONO essere rispettate le seguenti indicazioni:

- Le specifiche delle interfacce del fruitore e dell'erogatore DEVONO dichiarare tutti i codici di stato HTTP previsti dall'interfaccia, con il relativo schema della risposta, oltre che ad eventuali header HTTP restituiti;
- Le specifiche delle interfacce del fruitore e dell'erogatore DEVONO dichiarare gli schemi delle richieste insieme ad eventuali header HTTP richiesti;
- La specifica dell'interfaccia dell'erogatore deve dichiarare tramite il formalismo specifico il formato delle callback; questa specifica deve essere rispettata dall'interfaccia esposta dal fruitore, e quindi nella rispettiva specifica;
- Al passo (1), il fruitore DEVE indicare l'endpoint della callback utilizzando l'header HTTP custom *X-ReplyTo* ed usando HTTP method POST ;
- Al passo (2), l'erogatore DEVE fornire insieme all'acknowledgement della richiesta nel body, il CorrelationID utilizzando l'header HTTP custom *X-Correlation-ID*; Il codice HTTP di stato DEVE essere HTTP status 202 Accepted a meno che non si verifichino errori;

- Al passo (3), l'erogatore DEVE utilizzare lo stesso CorrelationID fornito al passo (2) sempre utilizzando l'header HTTP custom *X-Correlation-ID*; Il verbo HTTP utilizzato deve essere POST;
- Al passo (4), il fruitore DEVE riconoscere tramite un messaggio di acknowledgement il ricevimento della risposta; Il codice HTTP di stato DEVE essere **HTTP status 200 OK**[23] a meno che non si verifichino errori.

### Regole di processamento

Al termine del processamento delle richieste, l'erogatore ed il fruitore DEVONO fare uso dei codici di stato HTTP rispettando la semantica. Fruitore ed erogatore:

- DEVONO verificare la validità sintattica dei dati in ingresso. In caso di dati errati DEVONO restituire HTTP status 500 Internal Server Error fornendo dettagli circa l'errore;
- In caso di dati errati DEVONO restituire HTTP status 400 Bad Request fornendo nel body di risposta dettagli circa l'errore;
- In caso di representation semanticamente non corretta DEVONO ritornare HTTP status 422 Unprocessable Entity;
- Se qualcuno degli ID nel path o nel body non esiste, DEVONO restituire **HTTP status 404 Not Found**[24], indicando nel body di risposta quale degli ID è mancante;
- Se si ipotizza che la richiesta sia malevola, PUÒ ritornare HTTP status 400 Bad Request o **HTTP status 404 Not Found**[25]
- In caso di errori non dipendenti dalla richiesta, DEVONO restituire HTTP status 5XX rispettando la semantica degli stessi;
- Al momento della ricezione della richiesta, l'erogatore DEVE restituire HTTP status 202 Accepted. In caso di ricezione corretta della risposta, il fruitore DEVE restituire **HTTP status 200 OK**[26], ritornando nel body di risposta un acknowledgement dell'avvenuta ricezione. In caso di errore di ricezione della risposta da parte del fruitore, è possibile utilizzare meccanismi specifici per la ritrasmissione della risposta o della richiesta.

NB: I messaggi di errore devono essere utili al client ma NON DEVONO rivelare dettagli tecnici e/o informazioni riservate.

### Esempio

Specifica Servizio Server

https://api.ente.example/rest/nome-api/v1/RESTCallbackServer.yaml

```
openapi: 3.0.1
info:
 title: RESTCallbackServer
 version: "1.0"
 description: |-
   Questo file descrive semplicemente i metodi di un'API
   e non indica tutte le informazioni di metadatazione che
   normalmente vanno inserite.
 license:
   name: Apache 2.0 License
   url: http://www.apache.org/licenses/LICENSE-2.0.html
paths:
 /resources/{id_resource}/M:
   post:
```

(continues on next page)

[23] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200
[24] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[25] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[26] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

(continua dalla pagina precedente)

```
    description: M
    operationId: PushMessage
    parameters:
    - name: X-ReplyTo
      in: header
      schema:
        type: string
    - name: id_resource
      in: path
      required: true
      schema:
        type: integer
        format: int32
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/MType'
    responses:
      202:
        description: Preso carico correttamente di M
        headers:
          X-Correlation-ID:
            required: true
            schema:
              type: string
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ACKMessage'
      400:
        description: Richiesta non valida
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorMessage'
      404:
        description: Identificativo non trovato
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorMessage'
      default:
        $ref: '#/components/responses/default'
    callbacks:
      completionCallback:
        '{$request.header#/X-ReplyTo}':
          post:
            requestBody:
              content:
                application/json:
                  schema:
                    $ref: '#/components/schemas/MResponseType'
            responses:
              200:
                description: Risposta correttamente ricevuta
                content:
                  application/json:
                    schema:
                      $ref: '#/components/schemas/ACKMessage'
```

(continues on next page)

(continua dalla pagina precedente)

```
              default:
                $ref: '#/components/responses/default'
components:
 responses:

   default:
     description: |-
       Errore inatteso. Non ritornare informazioni
       sulla logica interna e/o non pertinenti all'interfaccia.
     content:
       application/json:
         schema:
           $ref: '#/components/schemas/ErrorMessage'
 schemas:
   MType:
     type: object
     properties:
       a:
         $ref: '#/components/schemas/AComplexType'
       b:
         type: string
   ACKMessage:
     type: object
     properties:
       outcome:
         type: string
   MResponseType:
     type: object
     properties:
       c:
         type: string
   AComplexType:
     type: object
     properties:
       a1s:
         type: array
         items:
           type: integer
           format: int32
       a2:
         type: string
   ErrorMessage:
      type: object
      properties:
        detail:
          description: |
            A human readable explanation specific to this occurrence of the
            problem.
          type: string
        instance:
          description: |
            An absolute URI that identifies the specific occurrence of the problem.
            It may or may not yield further information if dereferenced.
          format: uri
          type: string
        status:
          description: |
            The HTTP status code generated by the origin server for this occurrence
            of the problem.
          exclusiveMaximum: true
          format: int32
```

(continues on next page)

(continua dalla pagina precedente)

```
          maximum: 600
          minimum: 100
          type: integer
        title:
          description: |
            A short, summary of the problem type. Written in english and readable
            for engineers (usually not suited for non technical stakeholders and
            not localized); example: Service Unavailable
          type: string
        type:
          default: about:blank
          description: |
            An absolute URI that identifies the problem type.  When dereferenced,
            it SHOULD provide human-readable documentation for the problem type
            (e.g., using HTML).
          format: uri
          type: string
```

Specifica Servizio Client

https://api.client.example/rest/nome-api/v1/RESTCallbackClient.yaml

```
openapi: 3.0.1
info:
 title: RESTCallbackClient
 version: "1.0"
 description: |-
   Questo file descrive semplicemente i metodi di un'API
   e non indica tutte le informazioni di metadatazione che
   normalmente vanno inserite.
 license:
   name: Apache 2.0 License
   url: http://www.apache.org/licenses/LICENSE-2.0.html
paths:
 /MResponse:
   post:
     description: M
     operationId: PushResponseMessage
     parameters:
     - name: X-Correlation-ID
       in: header
       schema:
         type: string
     requestBody:
       content:
         application/json:
           schema:
             $ref: '#/components/schemas/MResponseType'
     responses:
       200:
         description: Risposta correttamente ricevuta
         content:
           application/json:
             schema:
               $ref: '#/components/schemas/ACKMessage'
       400:
         description: Richiesta non valida
         content:
           application/json:
             schema:
               $ref: '#/components/schemas/ErrorMessage'
```

(continues on next page)

(continua dalla pagina precedente)

```
        404:
          description: Identificativo non trovato
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorMessage'
        default:
          description: |-
            Errore inatteso. Non ritornare informazioni
            sulla logica interna e/o non pertinenti all'interfaccia.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorMessage'

components:
 schemas:
   ACKMessage:
     type: object
     properties:
       outcome:
         type: string
   MResponseType:
     type: object
     properties:
       c:
         type: string
   ErrorMessage:
      type: object
      properties:
        detail:
          description: |
            A human readable explanation specific to this occurrence of the
            problem.
          type: string
        instance:
          description: |
            An absolute URI that identifies the specific occurrence of the problem.
            It may or may not yield further information if dereferenced.
          format: uri
          type: string
        status:
          description: |
            The HTTP status code generated by the origin server for this occurrence
            of the problem.
          exclusiveMaximum: true
          format: int32
          maximum: 600
          minimum: 100
          type: integer
        title:
          description: |
            A short, summary of the problem type. Written in english and readable
            for engineers (usually not suited for non technical stakeholders and
            not localized); example: Service Unavailable
          type: string
        type:
          default: about:blank
          description: |
            An absolute URI that identifies the problem type.  When dereferenced,
            it SHOULD provide human-readable documentation for the problem type
```

(continues on next page)

(continua dalla pagina precedente)

```
            (e.g., using HTML).
          format: uri
          type: string
```

Di seguito il fruitore effettua una richiesta al metodo **M**; l'erogatore conferma la presa in carico della richiesta ritornando **HTTP status 202 Accepted**[27].

Endpoint: https://api.ente.example/rest/nome-api/v1/resources/1234/M

1. Request

```
POST /rest/nome-api/v1/resources/1234/M HTTP/1.1
Host: api.ente.example
Content-Type: application/json
X-ReplyTo: https://api.client.example/rest/v1/nomeinterfacciaclient/Mresponse

{
  "a": {
    "a1": [ 1, "..", 2 ],
    "a2": "RGFuJ3MgVG9vbHMgYXJlIGNvb2wh"
  },
  "b": "Stringa di esempio"
}
```

2. Response

```
HTTP/1.1 202 Accepted
Content-Type: application/json
X-Correlation-ID: 69a445fb-6a9f-44fe-b1c3-59c0f7fb568d

{"result": "ACK"}
```

Quindi l'erogatore notifica al fruitore l'avvenuto processamento delle informazioni tramite una **HTTP method POST** [28] all'URL concordato. Il fruitore conferma con **HTTP status 200 OK**[29] l'avvenuta ricezione della notifica.

Endpoint

https://api.client.example/rest/v1/nomeinterfacciaclient/Mresponse

3. Request

```
POST /rest/v1/nomeinterfacciaclient/Mresponse HTTP/1.1
Host: api.client.example
Content-Type: application/json
X-Correlation-ID: 69a445fb-6a9f-44fe-b1c3-59c0f7fb568d

{"c": "OK"}
```

4. Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": "ACK"}
```

[27] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/202
[28] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST
[29] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

#### 2.6.1.2 [NONBLOCK_PUSH_SOAP] Not Blocking Push SOAP

Nel caso di implementazione mediante tecnologia SOAP, l'endpoint di callback ed il CorrelationID, vengono inseriti all'interno dell'header SOAP come campi custom. Erogatore e fruitore DEVONO inoltre seguire le seguenti regole:

- Le specifiche delle interfacce del fruitore e dell'erogatore DEVONO dichiarare tutti i metodi esposti con relativi schemi dei messaggi di richiesta e di ritorno. Inoltre le interfacce devono specificare eventuali header SOAP richiesti;
- La specifica dell'interfaccia del fruitore DEVE rispettare quanto richiesto dall'erogatore; in particolare si richiede che l'erogatore fornisca un WSDL descrittivo del servizio di callback che il fruitore è tenuto ad implementare;
- Al passo (1), il fruitore DEVE indicare l'endpoint della callback utilizzando l'header SOAP custom X-ReplyTo;
- Al passo (2), l'erogatore DEVE fornire insieme all'acknowledgement della richiesta nel body, il CorrelationID utilizzando l'header SOAP custom X-Correlation-ID;
- Al passo (3), l'erogatore DEVE utilizzare lo stesso CorrelationID fornito al passo (2) sempre utilizzando l'header SOAP custom X-Correlation-ID;
- Al passo (4), il fruitore DEVE riconoscere tramite un messaggio di acknowledgement il ricevimento della risposta.

##### Regole di processamento

Nel caso di errore il WS-I Basic Profile Version 2.0 richiede l'utilizzo del meccanismo della SOAP fault per descrivere i dettagli dell'errore. In particolare, al ricevimento della richiesta, fruitore ed erogatore:

- DEVONO verificare la validità sintattica dei dati in ingresso. In caso di dati errati DEVONO restituire HTTP status 500 Internal Server Error fornendo dettagli circa l'errore, utilizzando il meccanismo della SOAP fault;
- Nel caso in cui qualcuno degli ID nel path o nel body non esista, DEVONO restituire **HTTP status 500 Internal Server Error**[30], indicando nel body di risposta quale degli ID è mancante;
- Se ipotizzano che la richiesta sia malevola POSSONO ritornare HTTP status 400 Bad Request o **HTTP status 404 Not Found**[31]
- In caso di errori non dipendenti dal fruitore, DEVE restituire i codici HTTP 5XX rispettando la semantica degli stessi o restituire il codice HTTP status 500 indicando il motivo dell'errore nella SOAP fault;
- Al momento della ricezione della richiesta, DEVONO restituire un codice 2XX, nel dettaglio:
  - **HTTP status 200 OK**[32] in caso di presenza della payload SOAP, riempiendo il body di risposta con il risultato relativo alla richiesta.
  - **HTTP status 200 OK**[33] o HTTP status 202 Accepted in caso di assenza della payload SOAP
- Nel caso di errore al momento di ricezione della risposta da parte del richiedente (fruitore o erogatore), è possibile definire meccanismi specifici per la ri-trasmettere le richieste.

##### Esempio

Specifica Servizio Erogatore

https://api.ente.example/soap/nome-api/v1?wsdl

[30] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500
[31] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[32] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200
[33] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

```
<?xml version="1.0"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://ente.example/nome-api"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
    name="SOAPCallbackServerService"
    targetNamespace="http://ente.example/nome-api">
    <wsdl:types>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://ente.example/nome-api" attributeFormDefault=
↪"unqualified" elementFormDefault="unqualified" targetNamespace="http://ente.
↪example/nome-api">

            <xs:element name="MRequest" type="tns:MRequest"/>
            <xs:element name="MRequestResponse" type="tns:MRequestResponse"/>

            <xs:element name="ErrorMessageFault" nillable="true" type=
↪"tns:errorMessageFault"/>
            <xs:element name="X-ReplyTo" nillable="true" type="xs:string"/>
            <xs:element name="X-Correlation-ID" nillable="true" type="xs:string"/>

            <xs:complexType name="MRequest">
                <xs:sequence>
                    <xs:element minOccurs="0" name="M" type="tns:mType"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="mType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="o_id" type="xs:int"/>
                    <xs:element minOccurs="0" name="a" type="tns:aComplexType"/>
                    <xs:element minOccurs="0" name="b" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="aComplexType">
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="a1s"␣
↪nillable="true" type="xs:string"/>
                    <xs:element minOccurs="0" name="a2" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="MRequestResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="tns:ackMessage"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="ackMessage">
                <xs:sequence>
                    <xs:element minOccurs="0" name="outcome" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="errorMessageFault">
                <xs:sequence>
                    <xs:element minOccurs="0" name="customFaultCode" type=
↪"xs:string"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
```

(continues on next page)

(continua dalla pagina precedente)

```
    <wsdl:message name="MRequest">
        <wsdl:part element="tns:MRequest" name="parameters"/>
        <wsdl:part element="tns:X-ReplyTo" name="X-ReplyTo"/>
    </wsdl:message>

    <wsdl:message name="MRequestResponse">
        <wsdl:part element="tns:MRequestResponse" name="result"/>
        <wsdl:part element="tns:X-Correlation-ID" name="X-Correlation-ID"/>
    </wsdl:message>

    <wsdl:message name="ErrorMessageException">
        <wsdl:part element="tns:ErrorMessageFault" name="ErrorMessageException"/>
    </wsdl:message>

    <wsdl:portType name="SOAPCallback">
        <wsdl:operation name="MRequest">
            <wsdl:input message="tns:MRequest" name="MRequest"/>
            <wsdl:output message="tns:MRequestResponse" name="MRequestResponse"/>
            <wsdl:fault message="tns:ErrorMessageException" name=
↪"ErrorMessageException"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="SOAPCallbackServiceSoapBinding" type="tns:SOAPCallback">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
↪http"/>
        <wsdl:operation name="MRequest">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name="MRequest">
                <soap:header message="tns:MRequest" part="X-ReplyTo" use="literal"/
↪>
                <soap:body parts="parameters" use="literal"/>
            </wsdl:input>
            <wsdl:output name="MRequestResponse">
                <soap:header message="tns:MRequestResponse" part="X-Correlation-ID
↪" use="literal"/>
                <soap:body parts="result" use="literal"/>
            </wsdl:output>
            <wsdl:fault name="ErrorMessageException">
                <soap:fault name="ErrorMessageException" use="literal"/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="SOAPCallbackService">
        <wsdl:port name="SOAPCallbackPort" binding=
↪"tns:SOAPCallbackServiceSoapBinding">
            <soap:address location="https://api.ente.example/soap/nome-api/v1"/>
        </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

Specifica Servizio Fruitore (callback)

https://api.client.example/soap/nome-api/v1?wsdl

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://ente.example/nome-api"
```

(continues on next page)

(continua dalla pagina precedente)

```
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
    name="SOAPCallbackClientInterfaceService"
    targetNamespace="http://ente.example/nome-api">
    <wsdl:types>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://ente.example/nome-api" attributeFormDefault=
↪"unqualified" elementFormDefault="unqualified" targetNamespace="http://ente.
↪example/nome-api">

            <xs:element name="MRequestResponse" type="tns:MRequestResponse"/>

            <xs:element name="MRequestResponseResponse" type=
↪"tns:MRequestResponseResponse"/>

            <xs:element name="X-Correlation-ID" nillable="true" type="xs:string"/>

            <xs:complexType name="MRequestResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="tns:mResponseType
↪"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="mResponseType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="c" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="MRequestResponseResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="tns:ackMessage"/>
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="ackMessage">
                <xs:sequence>
                    <xs:element minOccurs="0" name="outcome" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>

    <wsdl:message name="MRequestResponse">
        <wsdl:part element="tns:MRequestResponse" name="parameters"/>
        <wsdl:part element="tns:X-Correlation-ID" name="X-Correlation-ID"/>
    </wsdl:message>

    <wsdl:message name="MRequestResponseResponse">
        <wsdl:part element="tns:MRequestResponseResponse" name="result"/>
    </wsdl:message>

    <wsdl:portType name="SOAPCallbackClient">
        <wsdl:operation name="MRequestResponse">
            <wsdl:input message="tns:MRequestResponse" name="MRequestResponse"/>
            <wsdl:output message="tns:MRequestResponseResponse" name=
↪"MRequestResponseResponse"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="SOAPCallbackClientServiceSoapBinding" type=
↪"tns:SOAPCallbackClient">
```

(continues on next page)

(continua dalla pagina precedente)

```
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
↪http"/>
        <wsdl:operation name="MRequestResponse">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name="MRequestResponse">
                <soap:header message="tns:MRequestResponse" part="X-Correlation-ID
↪" use="literal"/>
                <soap:body parts="parameters" use="literal"/>
            </wsdl:input>
            <wsdl:output name="MRequestResponseResponse">
                <soap:body parts="result" use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="SOAPCallbackClientService">
        <wsdl:port binding="tns:SOAPCallbackClientServiceSoapBinding" name=
↪"SOAPCallbackClientPort">
            <soap:address location="https://api.client.example/soap/nome-api/v1"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Segue un esempio di chiamata al metodo **M** in cui l'erogatore conferma di essersi preso carico della richiesta.

Endpoint

https://api.ente.example/soap/nome-api/v1

Method M

1. Request Body

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Header>
        <m:X-ReplyTo>https://api.client.example/soap/nome-api/v1</m:X-ReplyTo>
    </soap:Header>
    <soap:Body>
        <m:MRequest>
            <M>
                <o_id>1234</o_id>
                <a>
                    <a1s>1</a1s>
                    <a2>prova</a2>
                </a>
                <b>prova</b>
            </M>
        </m:MRequest>
    </soap:Body>
</soap:Envelope>
```

2. Response Body

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">>
    <soap:Header>
        <m:X-Correlation-ID>b8268033-de67-4fa0-bf06-caebbfa5117a</m:X-Correlation-
↪ID>
    </soap:Header>
    <soap:Body>
        <m:MRequestResponse>
            <return>
```

(continues on next page)

(continua dalla pagina precedente)

```
                <outcome>ACCEPTED</outcome>
            </return>
        </m:MRequestResponse>
    </soap:Body>
</soap:Envelope>
```

Di seguito un esempio di richiesta da parte dell'erogatore verso la callback del fruitore.

Endpoint

https://api.client.example/soap/nomeinterfacciaclient/v1Method

MRequestResponse

3. Request Body

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Header>
        <m:X-Correlation-ID>b8268033-de67-4fa0-bf06-caebbfa5117a</m:X-Correlation-
↪ID>
    </soap:Header>
    <soap:Body>
        <m:MRequestResponse>
            <return>
                <c>OK</c>
            </return>
        </m:MRequestResponse>
    </soap:Body>
</soap:Envelope>
```

4. Response Body

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Body>
        <m:MRequestResponseResponse>
            <return>
                <outcome>OK</outcome>
            </return>
        </m:MRequestResponseResponse>
    </soap:Body>
</soap:Envelope>
```

### 2.6.2 Pattern non bloccanti RPC PULL (busy waiting)

Questo caso particolare, denominato RPC PULL, è utilizzabile nel caso in cui il fruitore non abbia a sua volta la possibilità di esporre una interfaccia di servizio per la ricezione delle risposte. L'erogatore fornisce un URL per verificare lo stato di processamento di una richiesta e, alla fine dell'elaborazione della stessa, il risultato.

Questo scenario prevede due possibili workflow, uno per REST ed uno per SOAP riportati nelle seguenti figure.

*Figura 3 - Interazione non bloccante tramite busy waiting REST*

Il fruitore invia una richiesta (passo 1.) e riceve immediatamente un acknowledge (passo 2.) insieme ad:

- un URL dove verificare lo stato del processamento (REST);
- oppure un CorrelationID (SOAP).

D'ora in poi il fruitore, periodicamente, verifica (passo 3.) lo stato dell'operazione utilizzando:

- l'URL indicato (REST)

- oppure il CorrelationID (SOAP)

fin quando il risultato dell'operazione sarà pronto (passo 4b.).

Gli intervalli di polling possono essere definiti tra le parti.

Quando la risposta è pronta il fruitore può accedere (passi 5. e 6.) al risultato del processamento

*Figura 4 - Interazione non bloccante tramite busy waiting SOAP*

#### 2.6.2.1 [NONBLOCK_PULL_REST] Not Blocking Pull REST

Quando il profilo viene implementato con tecnologia REST, DEVONO essere rispettate le seguenti regole:

- La specifica dell'interfaccia dell'erogatore DEVE dichiarare tutti i codici di stato HTTP restituiti con relativo schema della risposta, oltre che ad eventuali header HTTP restituiti;
- La specifica dell'interfaccia DEVE dichiarare gli schemi delle richieste insieme ad eventuali header HTTP richiesti;
- Al passo (1), il fruitore DEVE utilizzare il verbo HTTP POST;
- Al passo (2), l'erogatore DEVE fornire insieme all'acknowledgement della richiesta, un URL per interrogare lo stato di processamento utilizzando **HTTP header Location** [34], il codice HTTP di stato DEVE essere **HTTP status 202 Accepted**[35] a meno che non si verifichino errori;
- Al passo (3), il fruitore DEVE utilizzare il percorso di cui al passo (2) per richiedere lo stato della risorsa; il verbo HTTP utilizzato deve essere GET;
- Al passo (4) l'erogatore indica, sulla base dello stato del processamento, che l'operazione: * 4a. non è completata (**HTTP status 200 OK**[36]) * 4b. che la risorsa finale è pronta (**HTTP status 303 See Other**[37]) all'URL indicato nell' **HTTP header Location** [38];
- Al passo (5), il fruitore DEVE recuperare la risorsa con una richiesta **HTTP method GET** [39] all'URL indicato in (4b.);
- Al passo (6) l'erogatore risponde con la rappresentazione della risorsa, il codice HTTP restituito è **HTTP status 200 OK**[40].

#### Regole di processamento

Al termine del processamento delle richieste, l'erogatore deve fare uso dei codici di stato HTTP rispettando la semantica. In particolare, al ricevimento della richiesta da parte del fruitore, l'erogatore:

- DEVE verificare la validità sintattica e semantica dei dati in ingresso
- DEVE, in caso di dati errati, restituire **HTTP status 400 Bad Request**[41] fornendo nel body di risposta dettagli circa l'errore;
- DOVREBBE, in caso di representation semanticamente non corretta, ritornare **HTTP status 422 Unprocessable Entity**[42];
- DEVE, se qualcuno degli ID nel path o nel body non esiste, restituire **HTTP status 404 Not Found**[43], indicando nel body di risposta quale degli ID è mancante;

[34] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location
[35] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/202
[36] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200
[37] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/303
[38] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location
[39] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET
[40] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200
[41] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400
[42] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/422
[43] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404

- PUÒ, se ipotizza che la richiesta sia malevola, ritornare HTTP status 400 Bad Request o **HTTP status 404 Not Found**[44];
- DEVE, in caso di errori non dipendenti dalla richiesta, restituire HTTP status 5xx rispettando la semantica degli stessi;
- DEVE, ricevuta la richiesta, restituire **HTTP status 202 Accepted**[45].
- In caso di ricezione corretta della risposta, il fruitore DEVE restituire **HTTP status 200 OK**[46], riempiendo il body di risposta con il risultato dell'operazione.
- In caso di errore al momento di ricezione della risposta da parte del fruitore, è possibile definire meccanismi specifici per la ritrasmissione della risposta o della richiesta.

NB: I messaggi di errore devono essere utili al client ma NON DEVONO rivelare dettagli tecnici e/o informazioni riservate.

### Esempio

Specifica Servizio Server

https://api.ente.example/rest/nome-api/v1/openapi.yaml

```
openapi: 3.0.1
info:
 title: RESTbusywaiting
 version: "1.0"
 description: |-
   Questo file descrive semplicemente i metodi di un'API
   e non indica tutte le informazioni di metadatazione che
   normalmente vanno inserite.
 license:
   name: Apache 2.0 License
   url: http://www.apache.org/licenses/LICENSE-2.0.html
paths:
  /tasks/queue:
    post:
      description: Crea in maniera asincrona un task o una risorsa.
      operationId: PushMessage
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MType'
      responses:
        202:
          description: Preso carico correttamente
          headers:
            Location:
              description: URL dove verificare lo stato
              required: true
              schema:
                type: string
                format: uri
        '400':
          $ref: '#/components/responses/400BadRequest'

        default:
          $ref: '#/components/responses/default'
```

(continues on next page)

[44] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[45] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/202
[46] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

(continua dalla pagina precedente)

```
  /tasks/queue/{id_task}/:
    get:
      description: Verifica lo stato del task o risorsa
      operationId: CheckStatus
      parameters:
      - $ref: '#/components/parameters/id_task'
      responses:
        200:
          description: |-
            Lo stato del task o della risorsa.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TaskStatus'
        '303':
          description: Task Completato
          content:
              application/json:
                schema:
                  $ref: '#/components/schemas/TaskStatus'
          headers:
              Location:
                description: URL dove prelevare il risultato
                required: true
                schema:
                  type: string
                  format: uri
        '400':
          $ref: '#/components/responses/400BadRequest'
        '404':
          $ref: '#/components/responses/404NotFound'
        default:
          $ref: '#/components/responses/default'
  /tasks/result/{id_task}/:
    get:
      description: Recupera il risultato del task o la risorsa
      operationId: RetriveResource
      parameters:
      - $ref: '#/components/parameters/id_task'
      responses:
        200:
          description:  Il risultato del task o la risorsa
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/MResponseType'
        '400':
          $ref: '#/components/responses/400BadRequest'
        '404':
          $ref: '#/components/responses/404NotFound'
        default:
          $ref: '#/components/responses/default'
components:
  parameters:
    id_task:
      name: id_task
      in: path
      required: true
      schema:
        type: string
  responses:
```

(continues on next page)

(continua dalla pagina precedente)

```
    400BadRequest:
      description: Richiesta non accoglibile
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorMessage'
    404NotFound:
      description: Identificativo non trovato
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorMessage'
    default:
      description: |-
        Errore inatteso. Questo viene ritornato nel caso ci sia
        un errore inatteso. Non vanno mai esposti i dati interni
        del server.
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorMessage'
  schemas:
    TaskStatus:
      type: object
      properties:
        status:
          type: string
          enum: [pending, completed]
          example: pending
        message:
          type: string
    MType:
      type: object
      properties:
        a:
          $ref: '#/components/schemas/AComplexType'
        b:
          type: string
    MResponseType:
      type: object
      properties:
        c:
          type: string

    AComplexType:
      type: object
      properties:
        a1s:
          type: array
          items:
            type: string
        a2:
          type: string
    ErrorMessage:
      type: object
      properties:
        detail:
          description: |
            A human readable explanation specific to this occurrence of the
            problem.
          type: string
```

(continues on next page)

(continua dalla pagina precedente)

```
        instance:
          description: |
            An absolute URI that identifies the specific occurrence of the problem.
            It may or may not yield further information if dereferenced.
          format: uri
          type: string
        status:
          description: |
            The HTTP status code generated by the origin server for this occurrence
            of the problem.
          exclusiveMaximum: true
          format: int32
          maximum: 600
          minimum: 100
          type: integer
        title:
          description: |
            A short, summary of the problem type. Written in english and readable
            for engineers (usually not suited for non technical stakeholders and
            not localized); example: Service Unavailable
          type: string
        type:
          default: about:blank
          description: |
            An absolute URI that identifies the problem type.  When dereferenced,
            it SHOULD provide human-readable documentation for the problem type
            (e.g., using HTML).
          format: uri
          type: string
```

Il fruitore richiede la risorsa **M**, e l'erogatore risponde dichiarando di aver preso in carico la richiesta.

Endpoint https://api.ente.example/rest/nome-api/v1/resources/1234/M

1. Request:

```
POST /rest/nome-api/v1/resources/1234/M HTTP/1.1
Host: api.ente.example
Content-Type: application/json

{
  "a": {
     "a1": [1, "...", 2],
     "a2": "Stringa di esempio"
  },
  "b": "Stringa di esempio"
}
```

2. Response

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Location: /rest/nome-api/v1/resources/1234/M/8131edc0

{
  "status": "accepted",
  "message": "Preso carico della richiesta",
  "id": "8131edc0"
}
```

Quindi il fruitore verifica lo stato dell'esecuzione di **M** (3). L'erogatore ritorna un payload diverso a seconda dei casi: - processamento ancora in atto (4a) - e di processamento avvenuto (4b).

Endpoint

https://api.ente.example/rest/nome-api/v1/resources/1234/M/8131edc0

3. Request

```
GET /rest/nome-api/v1/resources/1234/M/8131edc0 HTTP/1.1
Host: api.ente.example
```

4a. Response (**HTTP status 200 OK**[47])

```
HTTP/1.1 200 OK
Host: api.ente.example
Content-Type: application/json

{
  "status": "processing",
  "message": "Richiesta in fase di processamento"
}
```

4b. Response (**HTTP status 303 See Other**[48])

Poiché al termine del processamento il client viene rediretto verso la risorsa finale, il payload deve contenere solamente le informazioni utili al redirect. Questo anche perché alcuni client potrebbero seguire direttamente l'**HTTP header Location** [49] ignorando il payload **HTTP status 303 See Other**[50] e ritornando invece quello indicato nel punto 5.

```
HTTP/1.1 303 See Other
Content-Type: application/json
Content-Location: /rest/nome-api/v1/resources/1234/M/8131edc0
Location: /rest/nome-api/v1/resources/1234/M/8131edc0/result

{
  "status": "done",
  "message": "Processamento completo",
  "href": "https://api.ente.example/rest/nome-api/v1/resources/1234/M/8131edc0/
↪result"
}
```

Di seguito un esempio di chiamata con cui il fruitore richiede l'esito della sua richiesta.

Endpoint

https://api.ente.example/rest/nome-api/v1/resources/1234/M/8131edc0/result

5. Request:

```
GET /rest/nome-api/v1/resources/1234/M/8131edc0/result HTTP/1.1
Host: api.ente.example
```

6. Response:

```
HTTP/1.1 200 OK
Host: api.ente.example
Content-Type: application/json

{"c": "OK"}
```

[47] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200
[48] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/303
[49] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location
[50] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/303

#### 2.6.2.2 [NONBLOCK_PULL_SOAP] Not Blocking Pull SOAP

Nel caso in cui il profilo venga implementato con tecnologia SOAP, DEVONO essere rispettate le seguenti regole:

- L'interfaccia di servizio dell'erogatore fornisce tre metodi differenti al fine di inoltrare una richiesta, controllare lo stato ed ottenerne il risultato;
- La specifica dell'interfaccia dell'erogatore DEVE indicare l'header SOAP `X-Correlation-ID`;
- Al passo (2), l'erogatore DEVE fornire insieme all'acknowledgement della richiesta nel body, un CorrelationID riportato nel header custom SOAP `X-Correlation-ID`;
- Al passo (3), il fruitore DEVE utilizzare il CorrelationID ottenuto al passo (2) per richiedere lo stato di processamento di una specifica richiesta;
- Al passo (4) l'erogatore, quando il processamento non si è ancora concluso fornisce informazioni circa lo stato della lavorazione della richiesta, quando invece il processamento si è concluso risponde indicando in maniera esplicita il completamento;
- Al passo (5), il fruitore utilizza il CorrelationID di cui al passo (2) al fine di richiedere il risultato della richiesta;
- Al passo (6), l'erogatore fornisce il risultato del processamento.

##### Regole di processamento

Nel caso di errore il WS-I Basic Profile Version 2.0 richiede l'utilizzo del meccanismo della SOAP fault per descrivere i dettagli dell'errore. Al ricevimento della richiesta da parte del fruitore, l'erogatore:

- DEVE verificare la validità sintattica dei dati in ingresso. In caso di dati errati DEVE restituire **HTTP status 500 Internal Server Error**[51] fornendo dettagli circa l'errore utilizzando il meccanismo della SOAP fault;
- Se l'erogatore ipotizza che la richiesta sia malevola PUÒ ritornare **HTTP status 400 Bad Request**[52] o **HTTP status 404 Not Found**[53];
- In caso di errori non dipendenti dal fruitore, DEVE restituire i codici HTTP 5XX rispettando la semantica degli stessi o restituire il codice **HTTP status 500 Internal Server Error**[54] indicando il motivo dell'errore nella SOAP fault;
- In caso di successo restituire **HTTP status 200 OK**[55], riempiendo il body di risposta con il risultato dell'operazione.

##### Esempio

Specifica Servizio Erogatore

https://api.ente.example/soap/nome-api/v1?wsdl

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://ente.example/nome-api"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
    name="SOAPPullService"
    targetNamespace="http://ente.example/nome-api">
    <wsdl:types>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

(continues on next page)

[51] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500
[52] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400
[53] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[54] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500
[55] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

(continua dalla pagina precedente)

```
            xmlns:tns="http://ente.example/nome-api" attributeFormDefault=
↪"unqualified" elementFormDefault="unqualified" targetNamespace="http://ente.
↪example/nome-api">

            <xs:element name="MProcessingStatus" type="tns:MProcessingStatus" />
            <xs:element name="MProcessingStatusResponse" type=
↪"tns:MProcessingStatusResponse" />

            <xs:element name="MRequest" type="tns:MRequest" />
            <xs:element name="MRequestResponse" type="tns:MRequestResponse" />

            <xs:element name="MResponse" type="tns:MResponse" />
            <xs:element name="MResponseResponse" type="tns:MResponseResponse" />

            <xs:element name="ErrorMessageFault" nillable="true" type=
↪"tns:errorMessageFault" />
            <xs:element name="X-Correlation-ID" nillable="true" type="xs:string" />

            <xs:complexType name="MProcessingStatus"/>
            <xs:complexType name="MProcessingStatusResponse">
                <xs:sequence>
                    <xs:element name="return" type="tns:processingStatus" />
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="MRequest">
                <xs:sequence>
                    <xs:element name="M" type="tns:mType" />
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="MRequestResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type=
↪"tns:processingStatus" />
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="MResponse"/>
            <xs:complexType name="MResponseResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="tns:mResponseType
↪" />
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="mType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="o_id" type="xs:int" />
                    <xs:element minOccurs="0" name="a" type="tns:aComplexType" />
                    <xs:element minOccurs="0" name="b" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="aComplexType">
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="a1s"␣
↪nillable="true" type="xs:string" />
                    <xs:element minOccurs="0" name="a2" type="xs:string" />
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="processingStatus">
```

(continues on next page)

(continua dalla pagina precedente)

```
                <xs:sequence>
                    <xs:element name="status" type="xs:string" />
                    <xs:element name="message" type="xs:string" />
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="mResponseType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="c" type="xs:string" />
                </xs:sequence>
            </xs:complexType>

            <xs:complexType name="errorMessageFault">
                <xs:sequence>
                    <xs:element name="customFaultCode" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>

    <wsdl:message name="MProcessingStatus">
        <wsdl:part element="tns:MProcessingStatus" name="parameters"/>
        <wsdl:part element="tns:X-Correlation-ID" name="X-Correlation-ID"/>
    </wsdl:message>

    <wsdl:message name="MProcessingStatusResponse">
        <wsdl:part element="tns:MProcessingStatusResponse" name="parameters"/>
    </wsdl:message>

    <wsdl:message name="MRequest">
        <wsdl:part element="tns:MRequest" name="parameters"/>
    </wsdl:message>

    <wsdl:message name="MRequestResponse">
        <wsdl:part element="tns:MRequestResponse" name="result"/>
        <wsdl:part element="tns:X-Correlation-ID" name="X-Correlation-ID"/>
    </wsdl:message>

    <wsdl:message name="MResponse">
        <wsdl:part element="tns:MResponse" name="parameters"/>
        <wsdl:part element="tns:X-Correlation-ID" name="X-Correlation-ID"/>
    </wsdl:message>

    <wsdl:message name="MResponseResponse">
        <wsdl:part element="tns:MResponseResponse" name="parameters"/>
    </wsdl:message>

    <wsdl:message name="ErrorMessageException">
        <wsdl:part element="tns:ErrorMessageFault" name="ErrorMessageException"/>
    </wsdl:message>

    <wsdl:portType name="SOAPPull">
        <wsdl:operation name="MRequest">
            <wsdl:input message="tns:MRequest" name="MRequest"/>
            <wsdl:output message="tns:MRequestResponse" name="MRequestResponse"/>
            <wsdl:fault message="tns:ErrorMessageException" name=
↪"ErrorMessageException"/>
        </wsdl:operation>

        <wsdl:operation name="MProcessingStatus">
            <wsdl:input message="tns:MProcessingStatus" name="MProcessingStatus"/>
```

(continues on next page)

(continua dalla pagina precedente)

```
            <wsdl:output message="tns:MProcessingStatusResponse" name=
↪"MProcessingStatusResponse"/>
            <wsdl:fault message="tns:ErrorMessageException" name=
↪"ErrorMessageException"/>
        </wsdl:operation>

        <wsdl:operation name="MResponse">
            <wsdl:input message="tns:MResponse" name="MResponse"/>
            <wsdl:output message="tns:MResponseResponse" name="MResponseResponse"/>
            <wsdl:fault message="tns:ErrorMessageException" name=
↪"ErrorMessageException"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="SOAPPullServiceSoapBinding" type="tns:SOAPPull">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
↪http" />

        <wsdl:operation name="MRequest">
            <soap:operation soapAction="" style="document" />

            <wsdl:input name="MRequest">
                <soap:body use="literal" />
            </wsdl:input>

            <wsdl:output name="MRequestResponse">
                <soap:header message="tns:MRequestResponse" part="X-Correlation-ID
↪" use="literal"/>
                <soap:body parts="result" use="literal" />
            </wsdl:output>

            <wsdl:fault name="ErrorMessageException">
                <soap:fault name="ErrorMessageException" use="literal" />
            </wsdl:fault>

        </wsdl:operation>

        <wsdl:operation name="MProcessingStatus">
            <soap:operation soapAction="" style="document" />

            <wsdl:input name="MProcessingStatus">
                <soap:header message="tns:MProcessingStatus" part="X-Correlation-ID
↪" use="literal"/>
                <soap:body parts="parameters" use="literal" />
            </wsdl:input>

            <wsdl:output name="MProcessingStatusResponse">
                <soap:body use="literal" />
            </wsdl:output>

            <wsdl:fault name="ErrorMessageException">
                <soap:fault name="ErrorMessageException" use="literal" />
            </wsdl:fault>

        </wsdl:operation>

        <wsdl:operation name="MResponse">
            <soap:operation soapAction="" style="document" />

            <wsdl:input name="MResponse">
                <soap:header message="tns:MResponse" part="X-Correlation-ID" use=
↪"literal"/>
```

(continues on next page)

(continua dalla pagina precedente)

```
            <soap:body parts="parameters" use="literal" />
        </wsdl:input>

        <wsdl:output name="MResponseResponse">
            <soap:body use="literal" />
        </wsdl:output>

        <wsdl:fault name="ErrorMessageException">
            <soap:fault name="ErrorMessageException" use="literal" />
        </wsdl:fault>
    </wsdl:operation>
</wsdl:binding>

<wsdl:service name="SOAPPullService">
    <wsdl:port binding="tns:SOAPPullServiceSoapBinding" name="SOAPPullPort">
        <soap:address location="https://api.ente.example/soap/nome-api/v1" />
    </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Di seguito un esempio di chiamata ad **M** in cui l'erogatore risponde di avere preso in carico la richiesta.

Endpoint

https://api.ente.example/soap/nome-api/v1

Method MRequest

1. Request Body

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Body>
        <m:MRequest>
            <M>
                <o_id>1234</o_id>
                <a>
                    <a1s>1</a1s>
                    <a2>prova</a2>
                </a>
                <b>prova</b>
            </M>
        </m:MRequest>
    </soap:Body>
</soap:Envelope>
```

2. Response Body (HTTP status code 200 OK)

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Header>
        <m:X-Correlation-ID>c8e191a8-f34f-41ed-82ea-68e096466707</m:X-Correlation-
↪ID>
    </soap:Header>
    <soap:Body>
        <m:MRequestResponse>
            <return>
                <status>accepted</status>
                <message>Preso carico della richiesta</message>
            </return>
        </m:MRequestResponse>
```

(continues on next page)

(continua dalla pagina precedente)

```
    </soap:Body>
</soap:Envelope>
```

Di seguito un esempio di chiamata con cui il fruitore verifica l'esecuzione di M nei casi di processamento ancora in atto e di processamento avvenuto.

Endpoint

https://api.ente.example/soap/nome-api/v1

Method MProcessingStatus

3. Request Body status

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Header>
        <m:X-Correlation-ID>c8e191a8-f34f-41ed-82ea-68e096466707</m:X-Correlation-
↪ID>
    </soap:Header>
    <soap:Body>
        <m:MProcessingStatus/>
    </soap:Body>
</soap:Envelope>
```

4. Response Body (HTTP status code 200 OK) status in attesa

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Body>
        <m:MProcessingStatusResponse>
            <return>
                <status>processing</status>
                <message>Richiesta in fase di processamento</message>
            </return>
        </m:MProcessingStatusResponse>
    </soap:Body>
</soap:Envelope>
```

4. Response Body (HTTP status code 200 OK) status completata

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Body>
        <m:MProcessingStatusResponse>
            <return>
                <status>done</status>
                <message>Richiesta completata</message>
            </return>
        </m:MProcessingStatusResponse>
    </soap:Body>
</soap:Envelope>
```

Di seguito un esempio di chiamata con cui il fruitore richiede l'esito della sua richiesta.

Endpoint

https://api.ente.example/soap/nome-api/v1

Method MResponse

5. Request Body result

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Header>
        <m:X-Correlation-ID>c8e191a8-f34f-41ed-82ea-68e096466707</m:X-Correlation-
↪ID>
    </soap:Header>
    <soap:Body>
        <m:MResponse/>
    </soap:Body>
</soap:Envelope>
```

6. Response Body (HTTP status code 200 OK) result

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:m="http://ente.example/nome-api">
    <soap:Body>
        <m:MResponseResponse>
            <return>
                <c>OK</c>
            </return>
        </m:MResponseResponse>
    </soap:Body>
</soap:Envelope>
```

## 2.7 Accesso CRUD a risorse

In un'architettura orientata alle risorse le API vengono utilizzate, più che per eseguire compiti complessi, per eseguire operazioni di tipo CRUD - Create, Read, Update, Delete - su risorse del dominio di interesse. Ad esempio, una prenotazione è una risorsa che può essere creata (quando viene fissata), letta, modificata ed eliminata.

In questo scenario si assume che le API siano utilizzate per la gestione da parte del fruitore delle risorse messe a disposizione dall'erogatore. L'insieme di operazioni CRUD offerte dall'erogatore dipende dalla natura della risorse e dalla relazione costruita con i fruitori: sono possibili relazioni in cui l'erogatore rende disponibile ai fruitori la sola operazione di lettura (Read).

*Figura 5 - Interazione CRUD*

Il sequence diagram è, come si nota in figura, equivalente a quello di una RPC bloccante. In questo caso la richiesta DEVE contenere:

- Una operazione da effettuare sulla risorsa da scegliere tra Create, Read, Update e Delete;
- Un percorso che indica un identificativo di risorsa (es. una specifica prenotazione) oppure di una collezione di risorse (es. un insieme di prenotazioni);
- Nel caso di una operazione di Create o Update, un pacchetto dati indicante come inizializzare o modificare (anche parzialmente) la risorsa in questione.

Per quanto sia possibile sviluppare funzionalità CRUD anche sviluppando una interfaccia di servizio SOAP, il ModI prevede lo sviluppo di una interfaccia REST.

### 2.7.1 [CRUD_REST] CRUD REST

#### 2.7.1.1 Regole di processamento

L'approccio RESTful trova la sua applicazione naturale in operazioni CRUD - Create, Read, Update, Delete su risorse ed a tal fine sfrutta i metodi standard dell'HTTP per indicare tali operazioni. In particolare la seguente mappatura viene utilizzata:

| CRUD | Metodo HTTP | Esito (stato HTTP) Se applicato a intera collezione (es. `/clienti`) | Esito (stato HTTP) Se applicato a risorsa specifica (es. `/clienti/{id}`) |
|---|---|---|---|
| Create | POST | **HTTP status 201 Created**[56], l'header «Location» nella risposta può contenere un link a /clienti/{id} che indica l'ID creato. | 404 (Not Found), 409 (Conflict) se la risorsa è già esistente. |
| Read | GET | 200 (OK), lista dei clienti. Implementare meccanismi di paginazione, ordinamento e filtraggio per navigare grandi liste. | 200 (OK), 404 (Not Found), se l'ID non è stato trovato o è non valido. |
| Update: Replace/ Create | PUT | 405 (Method Not Allowed), a meno che non si voglia sostituire ogni risorsa nella collezione. | 200 (OK) o 204 (No Content). 404 (Not Found), se l'ID non è stato trovato o è non valido. 201 (Created), nel caso di UPSERT. |
| Update: Modify | PATCH | 405 (Method Not Allowed), a meno che non si voglia applicare una modifica ad ogni risorsa nella collezione. | 200 (OK) o 204 (No Content). 404 (Not Found), se l'ID non è stato trovato o è non valido. |
| Delete | DELETE | 405 (Method Not Allowed), a meno che non si voglia permettere di eliminare l'intera collezione. | 200 (OK). 404 (Not Found), se l'ID non è stato trovato o è non valido. |

Si noti l'uso distinto di **HTTP method PUT** [57] e **HTTP method PATCH** [58] per la sostituzione e l'applicazione di modifiche ad una risorsa rispettivamente. È possibile utilizzare anche altri **HTTP method a** [59] patto che si rispettino i dettami dell'approccio RESTful.

In alcuni casi l'" **HTTP method PUT** [60] può essere utilizzato con funzionalità di UPSERT (Update o Insert). In particolare, questo è necessario nei casi in cui sia il fruitore a definire gli identificativi del sistema erogatore.

Per usare il **HTTP method PATCH** [61] bisogna usare alcuni accorgimenti, perché questo metodo non è definito nelle nuove specifiche di HTTP/1.1 del 2014 ma nel precedente **RFC 5789**[62].

NON SI DOVREBBE associare un significato di patch a dei media-type che non lo prevedano (eg. `application/json` o `application/xml`) ma utilizzare dei media-type adeguati[1].

È possibile ad esempio usare `application/merge-patch+json` definito in **RFC 7396**[63] facendo attenzione:

- che **HTTP method PATCH** [64] rifiuti richieste con media-type non adeguato con **HTTP status 415 Unsupported Media Type**[65];
- che il media-type di patching sia compatibile con gli schemi utilizzati;
- di verificare le considerazioni di sicurezza presenti in **RFC 7396#section-5**[66] e **RFC 5789#section-5**[67].

#### 2.7.1.2 Esempio

Per illustrare l'approccio RESTful al CRUD, faremo l'esempio di un API per gestire le prenotazioni di un appuntamento presso un ufficio municipale. L'erogatore verifica la compatibilità con la disponibilità nello specifico orario ed accetta o nega la creazione o l'eventuale variazione. Come da specifica seguente i metodi implementati

[56] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/201
[57] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT
[58] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH
[59] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/a
[60] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT
[61] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH
[62] https://tools.ietf.org/html/rfc5789.html
[1] Cf. https://www.rfc-editor.org/errata/eid3169
[63] https://tools.ietf.org/html/rfc7396.html
[64] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH
[65] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/415
[66] https://tools.ietf.org/html/rfc7396.html#section-5
[67] https://tools.ietf.org/html/rfc5789.html#section-5

sono **HTTP method POST** [68] (creazione), **HTTP method DELETE** [69] (eliminazione), **HTTP method PATCH** [70] (modifica) e **HTTP method GET** [71] (lettura).

Specifica Servizio Server

https://api.ente.example/rest/appuntamenti/v1/openapi.yaml

```yaml
openapi: 3.0.1
info:
  title: RESTCRUD
  version: "1.0"
  description: |-
    Questo file descrive semplicemente i metodi di un'API
    e non indica tutte le informazioni di metadatazione che
    normalmente vanno inserite.
  license:
    name: Apache 2.0 License
    url: http://www.apache.org/licenses/LICENSE-2.0.html
paths:
  /municipio/{id_municipio}/ufficio/{id_ufficio}/prenotazioni:
    get:
      description: Mostra prenotazioni
      operationId: listReservations
      parameters:
      - $ref: '#/components/parameters/limit'
      - $ref: '#/components/parameters/cursor'
      - $ref: '#/components/parameters/path_id_municipio'
      - $ref: '#/components/parameters/path_id_ufficio'
      responses:
        '200':
          description: Una lista di prenotazioni.
          content:
            application/json:
              schema:
                properties:
                  prenotazioni:
                    type: array
                    items:
                      $ref: '#/components/schemas/Prenotazione'
                  count:
                    type: integer
                    format: int32
                  next:
                    type: string
        '400':
          $ref: '#/components/responses/400BadRequest'
        '404':
          $ref: '#/components/responses/404NotFound'
        default:
          $ref: '#/components/responses/default'

    post:
      description: Aggiungi una prenotazione
      operationId: AddReservation_1
      parameters:
      - $ref: '#/components/parameters/path_id_municipio'
      - $ref: '#/components/parameters/path_id_ufficio'
      requestBody:
        content:
```

(continues on next page)

[68] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST
[69] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE
[70] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH
[71] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

(continua dalla pagina precedente)

```
        application/json:
          schema:
            $ref: '#/components/schemas/Prenotazione'
    responses:
      '201':
        description: Prenotazione Creata.
        headers:
          Location:
            description: ID della prenotazione creata
            schema:
              type: string
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Prenotazione'
      '400':
        $ref: '#/components/responses/400BadRequest'
      '404':
        $ref: '#/components/responses/404NotFound'
      default:
        $ref: '#/components/responses/default'

/municipio/{id_municipio}/ufficio/{id_ufficio}/prenotazioni/{id_prenotazione}:
  get:
    description: LeggiPrenotazione
    operationId: GetReservation_1
    parameters:
    - $ref: '#/components/parameters/path_id_municipio'
    - $ref: '#/components/parameters/path_id_ufficio'
    - name: id_prenotazione
      in: path
      required: true
      schema:
        type: integer
        format: int32
    responses:
      '200':
        description: Prenotazione estratta correttamente
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Prenotazione'
      '400':
        $ref: '#/components/responses/400BadRequest'
      '404':
        $ref: '#/components/responses/404NotFound'
      default:
        $ref: '#/components/responses/default'
  delete:
    description: EliminaPrenotazione
    operationId: DeleteReservation
    parameters:
    - $ref: '#/components/parameters/path_id_municipio'
    - $ref: '#/components/parameters/path_id_ufficio'
    - name: id_prenotazione
      in: path
      required: true
      schema:
        type: integer
        format: int32
    responses:
```

(continues on next page)

(continua dalla pagina precedente)

```
        '200':
          description: Prenotazione eliminata correttamente
        '404':
          $ref: '#/components/responses/404NotFound'
        default:
          $ref: '#/components/responses/default'

    patch:
      description: ModificaPrenotazione
      operationId: PatchReservation
      parameters:
      - $ref: '#/components/parameters/path_id_municipio'
      - $ref: '#/components/parameters/path_id_ufficio'
      - name: id_prenotazione
        in: path
        required: true
        schema:
          type: integer
          format: int32
      requestBody:
        content:
          application/merge-patch+json:
            schema:
              $ref: '#/components/schemas/PatchPrenotazione'
      responses:
        '200':
          description: Prenotazione modificata correttamente
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Prenotazione'
        '400':
          $ref: '#/components/responses/400BadRequest'
        '404':
          $ref: '#/components/responses/404NotFound'
        default:
          $ref: '#/components/responses/default'
components:
  parameters:
    path_id_municipio:
      name: id_municipio
      in: path
      required: true
      schema:
        type: integer
        format: int32
    path_id_ufficio:
      name: id_ufficio
      in: path
      required: true
      schema:
        type: integer
        format: int32
    limit:
        description: How many items to return at one time (max 100)
        in: query
        name: limit
        schema:
          format: int32
          type: integer
    cursor:
```

(continues on next page)

(continua dalla pagina precedente)

```
        description: An opaque identifier that points to the next item in the␣
↪collection.
        example: 01BX9NSMKVXXS5PSP2FATZM123
        in: query
        name: cursor
        schema:
          type: string
  responses:
    400BadRequest:
      description: Richiesta non accoglibile
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorMessage'
    404NotFound:
      description: Identificativo non trovato
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorMessage'
    default:
      description: |-
        Errore inatteso. Questo viene ritornato nel caso ci sia
        un errore inatteso. Non vanno mai esposti i dati interni
        del server.
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorMessage'
  schemas:
    TaxCode:
        description: Il codice fiscale.
        example: RSSMRA75L01H501A
        externalDocs:
          url: https://w3id.org/italia/onto/CPV/taxCode
        pattern: /^(?:(?:[B-DF-HJ-NP-TV-Z]|[AEIOU])[AEIOU][AEIOUX]|[B-DF-HJ-NP-TV-
↪Z]{2}[A-Z]){2}[\dLMNP-V]{2}(?:[A-EHLMPR-T](?:[04LQ][1-9MNP-V]|[1256LMRS][\dLMNP-
↪V])|[DHPS][37PT][0L]|[ACELMRT][37PT][01LM])(?:[A-MZ][1-9MNP-V][\dLMNP-V]{2}|[A-
↪M][0L](?:[1-9MNP-V][\dLMNP-V]|[0L][1-9MNP-V]))[A-Z]$/i
        type: string
    Prenotazione:
      type: object
      properties:
        nome:
          type: string
        cognome:
          type: string
        codice_fiscale:
          $ref: '#/components/schemas/TaxCode'
        dettagli:
          $ref: '#/components/schemas/PatchPrenotazione'
    PatchPrenotazione:
      type: object
      properties:
        data:
          type: string
          format: date-time
        motivazione:
          type: string
    ErrorMessage:
      type: object
```

(continues on next page)

(continua dalla pagina precedente)

```
    properties:
      detail:
        description: |
          A human readable explanation specific to this occurrence of the
          problem.
        type: string
      instance:
        description: |
          An absolute URI that identifies the specific occurrence of the problem.
          It may or may not yield further information if dereferenced.
        format: uri
        type: string
      status:
        description: |
          The HTTP status code generated by the origin server for this occurrence
          of the problem.
        exclusiveMaximum: true
        format: int32
        maximum: 600
        minimum: 100
        type: integer
      title:
        description: |
          A short, summary of the problem type. Written in english and readable
          for engineers (usually not suited for non technical stakeholders and
          not localized); example: Service Unavailable
        type: string
      type:
        default: about:blank
        description: |
          An absolute URI that identifies the problem type.  When dereferenced,
          it SHOULD provide human-readable documentation for the problem type
          (e.g., using HTML).
        format: uri
        type: string
```

Di seguito un esempio di chiamata per creare una prenotazione.

1. Request

```
POST /rest/appuntamenti/v1/municipio/{id_municipio}/ufficio/{id_ufficio}/
↪prenotazioni HTTP/1.1
Host: api.ente.example
Content-Type: application/json

{
    "nome_proprio": "Mario",
    "cognome": "Rossi",
    "codice_fiscale": "MRORSS77T05E472I",
    "dettagli": {
        "data": "2018-12-03T14:29:12.137Z",
        "motivazione": "string"
    }
}
```

2. Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: https://api.ente.example/rest/appuntamenti/v1/municipio/{id_municipio}/
↪ufficio/{id_ufficio}/prenotazioni/12323254
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    "id": 12323254,
    "nome_proprio": "Mario",
    "cognome": "Rossi",
    "codice_fiscale": "MRORSS77T05E472I",
    "dettagli": {
        "data": "2018-12-03T14:29:12.137Z",
        "motivazione": "string"
    }
}
```

Di seguito, un esempio in cui il fruitore richiede l'estrazione di una specifica prenotazione. Si noti l'utilizzo dell'URL restituito nell' **HTTP header Location** [72] al passo precedente.

1. Request

```
GET /rest/appuntamenti/v1/municipio/{id_municipio}/ufficio/{id_ufficio}/
↪prenotazioni/12323254 HTTP/1.1
Host: api.ente.example
```

2. Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 12323254,
    "nome_proprio": "Mario",
    "cognome": "Rossi",
    "codice_fiscale": "MRORSS77T05E472I",
    "dettagli": {
        "data": "2018-12-03T14:29:12.137Z",
        "motivazione": "string"
    }
}
```

Di seguito una richiesta di modifica dei dettagli di una prenotazione.

1. Request

```
PATCH /rest/appuntamenti/v1/municipio/{id_municipio}/ufficio/{id_ufficio}/
↪prenotazioni/12323254 HTTP/1.1
Host: api.ente.example
Content-Type: application/merge-patch+json

{
    "dettagli": {
        "data": "2018-12-03T14:29:12.137Z",
        "motivazione": "nuova motivazione"
    }
}
```

2. Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "nome_proprio": "Mario",
```

(continues on next page)

[72] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location

(continua dalla pagina precedente)

```
 "cognome": "Rossi",
 "codice_fiscale": "MRORSS77T05E472I",
 "dettagli": {
     "data": "2018-12-03T14:29:12.137Z",
     "motivazione": "nuova motivazione"
 }
}
```

Di seguito una richiesta di modifica dei dettagli di una prenotazione con media-type application/json, che non avendo una semantica di patching definita, dev'essere rifiutato seguendo le indicazioni presenti in **RFC 5789#section-2.2**[73]. La response ritorna il media-type suggerito dalla specifica tramite **HTTP header Accept-Patch** [74]

1. Request

```
PATCH /rest/appuntamenti/v1/municipio/{id_municipio}/ufficio/{id_ufficio}/
↪prenotazioni/12323254 HTTP/1.1
Host: api.ente.example
Content-Type: application/json

    {
      "dettagli": {
            "data": "2018-12-03T14:29:12.137Z",
            "motivazione": "nuova motivazione"
      }
    }
```

2. Response

```
HTTP/1.1 415 Unsupported Media Type
Accept-Patch: application/merge-patch+json
```

Di seguito un esempio di cancellazione di una specifica prenotazione.

1. Request

```
DELETE /rest/appuntamenti/v1/municipio/{id_municipio}/ufficio/{id_ufficio}/
↪prenotazioni/12323254 HTTP/1.1
Host: api.ente.example
```

2. Response

```
HTTP/1.1 200 OK
```

[73] https://tools.ietf.org/html/rfc5789.html#section-2.2
[74] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Patch

CAPITOLO 3

# Documento operativo - Pattern sicurezza

## 3.1 Introduzione

I pattern di sicurezza definiscono le modalità per assicurare che le interazioni tra fruitore ed erogatore siano realizzate nel rispetto delle specifiche esigenze di sicurezza determinate dalla natura delle transazioni realizzate e dalle prescrizioni normative che riguardano le stesse.

I pattern di sicurezza si applicano ai pattern di interazione indicati nel Documento Operativo - Pattern di interazione, e sono scelti dall'erogatore in funzione alle specifiche esigenze applicative in relazione alla natura dei fruitori.

Il Documento operativo descrive i pattern di sicurezza individuati da AgID che gli erogatori DEVONO utilizzare per soddisfare le necessità individuate dai requisiti funzionali e non funzionali delle specifiche interazioni con i propri fruitori.

Data la variabilità nel tempo delle esigenze delle amministrazioni e delle tecnologie abilitanti, nonché considerata la natura incrementale del ModI, l'elenco dei pattern di sicurezza non è da intendersi esaustivo. Nel caso in cui un'amministrazione abbia esigenze non ricoperte nei seguenti pattern di sicurezza DEVE informare AgID, nei modi indicati nel capitolo 7 «Pattern e profili di interoperabilità» delle Linee di indirizzo sull'interoperabilità tecnica delle Pubbliche Amministrazioni. Le tecnologie e standard per assicurare la sicurezza dell'interoperabilità tramite API utilizzabili nel ModI, tra cui OAuth 2.0, sono individuate nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

I pattern di sicurezza individuati coprono gli aspetti di comunicazione «sicura» tra i domini delle singole parti. Le parti mantengono la loro autonomia negli aspetti organizzativi e di sicurezza interni al proprio dominio.

I pattern di sicurezza

- definiscono a livello di specifica tecnologica uno «strumento condiviso» utile a favorire l'interoperabilità tra erogatori e fruitori.
- forniscono un comune linguaggio per fruitori ed erogatori utile a trattare le necessità e le caratteristiche delle interfacce di servizio.
- offrono agli sviluppatori le modalità tecniche supportate da standard tecnologici documentati, revisionati e testati per esporre i servizi digitali.

I pattern di sicurezza affrontano il tema della sicurezza su due livelli differenti:

- Canale: definisce le modalità di trasporto dei messaggi tra i confini dei domini delle entità coinvolte.

- Messaggio: definisce le modalità di comunicazione dei messaggi tra componenti interne dei domini delle entità coinvolte.

Ogni pattern di sicurezza è strutturato come segue:

- Descrizione: rappresentazione in linguaggio naturale del profilo con relativi precondizioni e obiettivi.
- Regole di processamento: elenco dei passi da eseguire per implementare il profilo.
- Tracciato: ove presente, fornisce un esempio dei messaggi prodotti nell'interazione.

Gli erogatori, ove necessario in accordo con i fruitori, a seguito dell'analisi dei requisiti realizzata, per individuare le proprie esigenze funzionali e non funzionali, DOVREBBERO:

- individuare tra i pattern di interazione (vedi Documento operativo - Pattern di interazione) quelli che soddisfano le proprie esigenze;
- individuare tra i pattern di sicurezza quelli che soddisfano le proprie esigenze;
- implementare le interfacce di servizio attraverso la combinazione dei pattern di interazione e di pattern di sicurezza.

L'individuazione dei pattern di sicurezza DEVE ricoprire solamente i requisiti necessari.

Il *Trust* è uno dei mezzi più importanti per gestire le problematiche di sicurezza nello scambio di informazione in rete per consentire l'interoperabilità tra i sistemi. Esso si basa sul reciproco riconoscimento delle entità interagenti e sulla fiducia nei rispettivi comportamenti.

Nel presente Documento operativo, per *direct trust* si intende la relazione di fiducia tra fruitore ed erogatore, stabilita in modalità diretta, attraverso accordi che si basano sulla condivisione del reciproco modus operandi.

Si rimanda alle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale), in merito agli algoritmi utilizzabili per la corretta implementazione dei pattern di sicurezza.

## 3.2 Ambito di applicazione

Il presente Documento operativo, allegato delle Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni, definisce i pattern di sicurezza definiti nel ModI

### 3.2.1 Soggetti destinatari

Il Documento Operativo è destinato ai soggetti di cui all'articolo 2, comma 2 del CAD, così come indicato dall'articolo 75 dello stesso. I destinatari la attuano nella realizzazione dei propri sistemi informatici che fruiscono o erogano dati e/o servizi digitali ad altri soggetti.

Il Documento Operativo è rivolto ai soggetti privati che devono interoperare con la Pubblica Amministrazione per erogare o fruire di dati e servizi tramite sistemi informatici.

## 3.3 Riferimenti e sigle

### 3.3.1 Note di lettura del documento

Conformemente alle norme ISO/IEC Directives, Part 3 per la stesura dei documenti tecnici le presenti Linee Guida utilizzano le parole chiave «DEVE», «DEVONO», «NON DEVE», «NON DEVONO», «DOVREBBE», «NON DOVREBBE», «PUÒ» e «OPZIONALE», la cui interpretazione è descritta di seguito.

- **DEVE** o **DEVONO**, indicano un requisito obbligatorio per rispettare le Linee Guida;
- **NON DEVE** o **NON DEVONO**, indicano un assoluto divieto delle specifiche;

- **DOVREBBE** o **NON DOVREBBE**, indicano che le implicazioni devono essere comprese e attentamente pesate prima di scegliere approcci alternativi;
- **PUÒ** o **POSSONO** o l'aggettivo **OPZIONALE**, indica che il lettore può scegliere di applicare o meno senza alcun tipo di implicazione o restrizione la specifica.

### 3.3.2 Standard di riferimento

Sono riportati di seguito gli standard tecnici indispensabili per l'applicazione del presente documento.

Tabella 3.1: Riferimenti Normativi

| | |
|---|---|
| [ISO 19115] | UNI EN ISO 19115:2005, Informazioni geografiche – Metadati |
| [RFC3230] | Instance Digests in http |
| [RFC3744] | Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol |
| [RFC5246] | The Transport Layer Security (TLS) Protocol Version 1.2 |
| [RFC7231] | Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content |
| [RFC7233] | Hypertext Transfer Protocol (HTTP/1.1): Range Requests |
| [RFC7515] | JSON Web Signature (JWS) |
| [RFC7519] | JSON Web Token (JWT) |
| [RFC8725] | JSON Web Token Best Current Practice |

### 3.3.3 Termini e definizioni

Tabella 3.2: Termini e definizioni

| | |
|---|---|
| [AgID] | Agenzia per l'Italia Digitale |
| [CAD] | Codice Amministrazione Digitale, D.lgs. 7 marzo 2005, n. 82 |
| [PA] | Pubblica Amministrazione |
| [UML] | Linguaggio di modellazione unificato (Unified Modeling Language) |
| [RPC] | Remote procedure call |
| [SOAP] | Simple Object Access Protocol |
| [REST] | Representational State Transfer |

## 3.4 Sicurezza di canale e/o identificazione delle organizzazioni

Di seguito le indicazioni per le tecnologie accolte dal ModI.

L'AgID assicura l'aggiornamento degli stessi per soddisfare le esigenze espresse dalle PA.

### 3.4.1 [ID_AUTH_CHANNEL_01] Direct Trust Transport-Level Security

Comunicazione tra fruitore ed erogatore che assicuri, a livello di canale:

- confidenzialità;
- integrità;
- identificazione dell'erogatore, quale organizzazione;
- difesa dalle minacce derivanti dagli attacchi: Replay Attack e Spoofing.

#### 3.4.1.1 Descrizione

Il presente profilo assume l'esistenza di un trust tra fruitore ed erogatore, che permette il riconoscimento del certificato X.509, o la CA emittente dell'erogatore, così come previsto dal protocollo Transport Layer Security.

La sequenza dei messaggi di richiesta/risposta avviene dopo aver instaurato il canale di trasmissione sicuro.

*Figura 1 - Sicurezza di canale e/o Autenticazione dell'erogatore*

#### 3.4.1.2 Regole di processamento

Il canale sicuro tra erogatore e fruitore viene instaurato utilizzando il protocollo TLS, secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

**A: Richiesta**

1. Il fruitore costruisce un messaggio di richiesta;
2. Il fruitore spedisce sul canale sicuro stabilito il messaggio di richiesta all'interfaccia di servizio dell'erogatore.

**B: Risposta**

3. L'erogatore elabora il messaggio e restituisce il risultato.

Come indicato in **RFC 5246**[75] l'impiego del protocollo TLS garantisce a livello di canale:

- l'autenticazione dell'erogatore identificato mediante il certificato X.509;
- la confidenzialità dei dati scambiati;
- l'integrità dei dati scambiati.

L'impiego del protocollo TLS, mitiga il rischio di:

- Replay Attack;
- Spoofing.

### 3.4.2 [ID_AUTH_CHANNEL_02] Direct Trust mutual Transport-Level Security

Comunicazione tra fruitore ed erogatore che assicuri a livello di canale:

- confidenzialità;
- integrità;
- identificazione dell'erogatore e del fruitore, quale organizzazioni;
- difesa dalle minacce derivanti dagli attacchi: Replay Attack e Spoofing.

#### 3.4.2.1 Descrizione

Il presente profilo assume l'esistenza di un trust tra fruitore (client) ed erogatore (server), che permette il riconoscimento da entrambe le parti dei certificati X.509, o le CA emittenti, così come previsto dal protocollo Transport Layer Security.

La sequenza dei messaggi di richiesta/risposta avviene dopo aver instaurato il canale di trasmissione sicuro.

*Figura 2 - Sicurezza di canale e/o Autenticazione delle organizzazioni*

[75] https://tools.ietf.org/html/rfc5246.html

#### 3.4.2.2 Regole di processamento

Il canale sicuro tra erogatore e fruitore viene instaurato in mutua autenticazione utilizzando il protocollo TLS, secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

**A: Richiesta**

1. Il fruitore costruisce un messaggio di richiesta.
2. Il fruitore spedisce utilizzando il canale sicuro stabilito con il messaggio di richiesta all'interfaccia di servizio dell'erogatore.

**B: Risposta**

3. L'erogatore elabora il messaggio e restituisce il risultato.

Come indicato in **RFC 5246**[76] l'impiego del protocollo TLS garantisce a livello di canale:

- l'autenticazione di erogatore e fruitore identificati mediante certificati X.509;
- la confidenzialità dei dati scambiati;
- l'integrità dei dati scambiati.

L'impiego del protocollo TLS, mitiga il rischio di:

- Replay Attack;
- Spoofing.

## 3.5 Accesso del soggetto fruitore

Di seguito le indicazioni per le tecnologie accolte dal ModI.

L'AgID assicura l'aggiornamento degli stessi per soddisfare le esigenze espresse dalle PA.

### 3.5.1 [ID_AUTH_SOAP_01] Direct Trust con certificato X.509 su SOAP

Comunicazione tra fruitore ed erogatore che assicuri a livello di messaggio:

- accesso del soggetto fruitore, quale organizzazione o unità organizzativa fruitrice, o entrambe le parti.

#### 3.5.1.1 Descrizione

Il presente profilo specializza lo standard OASIS Web Services Security X.509 Certificate Token Profile Versione 1.1.1.

Si assume l'esistenza di un trust tra fruitore ed erogatore, che permette il riconoscimento da parte dell'erogatore del certificato X.509, o la CA emittente.

Il meccanismo con cui è stabilito il trust, inclusa la modalità di scambio dei certificati X.509, non condiziona il presente profilo.

Il fruitore inoltra un messaggio all'interfaccia di servizio dell'erogatore includendo o referenziando il certificato X.509 e una porzione significativa del messaggio firmata.

L'erogatore, ricevuto il messaggio, verifica il certificato X.509 e valida la porzione firmata del messaggio. Se la verifica e la validazione sono superate, l'erogatore elabora la richiesta e produce la relativa risposta.

*Figura 3 - Accesso del Fruitore*

[76] https://tools.ietf.org/html/rfc5246.html

#### 3.5.1.2 Regole di processamento

**A: Richiesta**

1. Il fruitore costruisce un messaggio SOAP per il servizio.
2. Il fruitore aggiunge al messaggio l'header WS-Addressing e l'elemento <wsu:Timestamp> composto dagli elementi <wsu:Created> e <wsu:Expires>
3. Il fruitore calcola la firma per gli elementi significativi del messaggio, in particolare <wsu:Timestamp> e <wsa:To> del blocco WS-Addressing. Il digest è firmato usando la chiave privata associata al certificato X.509 del fruitore. L'elemento <Signature> è posizionato nell'header <Security> del messaggio.
4. Il fruitore referenzia il certificato X.509 usando in maniera alternativa, nell'header <Security>, i seguenti elementi previsti nella specifica ws-security:
   (a) <wsse:BinarySecurityToken>
   (b) <wsse:KeyIdentifier>
   (c) <wsse:SecurityTokenReference>
5. Il fruitore spedisce il messaggio all'interfaccia di servizio dell'erogatore.

**B: Risposta**

6. L'erogatore verifica il contenuto dell'elemento <wsu:Timestamp> nell'header del messaggio al fine di verificare la validità temporale del messaggio.
7. L'erogatore verifica la corrispondenza tra se stesso e quanto definito nell'elemento <wsa:To> del blocco WS-Addressing.
8. L'erogatore recupera il certificato X.509 referenziato nell'header <Security>.
9. L'erogatore verifica il certificato secondo i criteri del trust.
10. L'erogatore valida l'elemento <Signature> nell'header <Security>.
11. L'erogatore garantisce l'accesso al fruitore.
12. Se le azioni da 6 a 11 hanno avuto esito positivo, il messaggio viene elaborato e viene restituito il risultato del servizio richiamato.

Note:

- In merito agli algoritmi da utilizzare nell'elemento <Signature> rispettivamente <DigestMethod>, <SignatureMethod> e <CanonicalizationMethod> si fa riferimento agli algoritmi indicati nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).
- Un meccanismo simile può essere utilizzato specularmente per l'erogatore.

#### 3.5.1.3 Esempio

Di seguito è riportato un tracciato del messaggio inoltrato dal fruitore all'interfaccia di servizio dell'erogatore relativo ad un servizio di echo.

I namespace utilizzati nel tracciato sono riportati di seguito:

```
soap="http://www.w3.org/2003/05/soap-envelope"
wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.
↪0.xsd"
wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.
↪0.xsd"
ds="http://www.w3.org/2000/09/xmldsig#"
ec="http://www.w3.org/2001/10/xml-exc-c14n#"
```

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
↪wss-wssecurity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/
↪01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://
↪docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"␣
↪wsu:Id="X509-39011475-65d5-446e-ba38-be84220fd720">
↪MIICqDCCAZCgAwIBAgIEXLSSUTANBgkqhkiG9w0BAQsFADAW...</wsse:BinarySecurityToken>
      <wsu:Timestamp wsu:Id="TS-819df7b7-379d-48f7-8d9c-28c5b5d252f0">
        <wsu:Created>2019-04-15T14:53:34.649Z</wsu:Created>
        <wsu:Expires>2019-04-15T14:58:34.649Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-6e09e972-
↪cbe6-43fc-a10c-38e6dce56dbe">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
↪c14n#">
            <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-
↪c14n#" PrefixList="soap"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more
↪#rsa-sha256"/>
          <ds:Reference URI="#TS-819df7b7-379d-48f7-8d9c-28c5b5d252f0">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-
↪exc-c14n#" PrefixList="soap    wsse"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <ds:DigestValue>K/3Fq1fYjG5PXv8UlKBuT4XBCWudGR5w2M10wPcZ/Yo=**</
↪ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-96f9b013-17e5-489d-8068-52c3f1345c75">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-
↪exc-c14n#" PrefixList="soap"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <ds:DigestValue>eH3Vlc3l19NbBawDOuFDN11BfmbgGAnl6Z4LpJVM3UM=**</
↪ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>jAtZqkfRcFJW+jx9YDv+r2Q8V4IWEWLAZckZlWsmo...</
↪ds:SignatureValue>
        <ds:KeyInfo Id="KI-32484d1e-867e-4465-a96f-52a8668d5a0c">
          <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/
↪2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" wsu:Id="STR-3cf69cce-c56f-461a-905d-dfc20ab0742c">
            <wsse:Reference URI="#X509-39011475-65d5-446e-ba38-be84220fd720"␣
↪ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
↪profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
```

(continues on next page)

(continua dalla pagina precedente)

```
    </wsse:Security>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://profile.security.
↪modi.agid.org/HelloWorld/sayHi</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:55e6bc57-2286-
↪4b7d-82a9-fdbcf57721b1</MessageID>
    <To xmlns="http://www.w3.org/2005/08/addressing"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" wsu:Id="id-96f9b013-17e5-489d-8068-52c3f1345c75">
↪https://api.ente.example/soap/echo/v1</To>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
      <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
    </ReplyTo>
  </soap:Header>
  <soap:Body>
    <ns2:sayHi xmlns:ns2="http://profile.security.modi.agid.org/">
      <arg0>OK</arg0>
    </ns2:sayHi>
  </soap:Body>
</soap:Envelope>
```

Il tracciato rispecchia le seguenti scelte implementative esemplificative:

- riferimento al security token (BinarySecurityToken)
- algoritmi di canonizzazione (CanonicalizationMethod)
- algoritmi di firma (SignatureMethod)
- algoritmo per il digest (DigestMethod)

Le parti, in base alle proprie esigenze, individuano gli specifici algoritmi secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

### 3.5.2 [ID_AUTH_SOAP_02] Direct Trust con certificato X.509 su SOAP con con unicità del token/messaggio

Il seguente profilo estende il profilo ID_AUTH_SOAP_01. Comunicazione tra fruitore ed erogatore che assicuri a livello di messaggio:

- accesso del soggetto fruitore, quale organizzazione o unità organizzativa fruitore, o entrambe le parti;
- difesa dalle minacce derivanti dagli attacchi: Replay Attack.

#### 3.5.2.1 Descrizione

Il presente profilo specializza lo standard OASIS Web Services Security X.509 Certificate Token Profile Versione 1.1.1].

Si assume l'esistenza di un trust tra fruitore ed erogatore, che permette il riconoscimento da parte dell'erogatore del certificato X.509, o la CA emittente.

Il meccanismo con cui è stabilito il trust, inclusa la modalità di scambio dei certificati X.509, non condiziona il presente profilo.

Il fruitore inoltra un messaggio all'interfaccia di servizio dell'erogatore includendo o referenziando il certificato X.509 e assicurando la firma dei claim del messaggio.

L'erogatore, ricevuto il messaggio, verifica il certificato X.509, valida la firma dei claim ed garantisce l'accesso al fruitore. Se la verifica e la validazione sono superate, l'erogatore elabora la richiesta e produce la relativa risposta.

*Figura 4 - Accesso del Fruitore*

#### 3.5.2.2 Regole di processamento

**A: Richiesta**

1. Il fruitore costruisce un messaggio SOAP per il servizio.
2. Il fruitore aggiunge al messaggio l'header WS-Addressing e l'elemento <wsu:Timestamp> composto dagli elementi <wsu:Created> e <wsu:Expires>
3. Il fruitore calcola la firma per gli elementi significativi del messaggio, in particolare <wsa:To> e <wsa:MessageID> del blocco WS-Addressing e <wsu:Timestamp>. Il digest è firmato usando la chiave privata associata al certificato X.509 del fruitore. L'elemento <Signature> è posizionato nell'header <Security> del messaggio.
4. Il fruitore referenzia il certificato X.509 usando in maniera alternativa, nell'header <Security>, i seguenti elementi previsti nella specifica ws-security:
   (a) <wsse:BinarySecurityToken>
   (b) <wsse:KeyIdentifier>
   (c) <wsse:SecurityTokenReference>
5. Il fruitore spedisce il messaggio all'interfaccia di servizio dell'erogatore.

**B: Risposta**

1. L'erogatore verifica il contenuto dell'elemento <wsu:Timestamp> nell'header del messaggio al fine di verificare la validità temporale del messaggio.
2. L'erogatore verifica la corrispondenza tra se stesso e quanto definito nell'elemento <wsa:To> del blocco WS-Addressing.
3. L'erogatore verifica l'univocità del <wsa:MessageID> del blocco WS-Addressing
4. L'erogatore recupera il certificato X.509 referenziato nell'header <Security>.
5. L'erogatore verifica il certificato secondo i criteri del trust.
6. L'erogatore valida l'elemento <Signature> nell'header <Security>.
7. L'erogatore garantisce l'accesso al fruitore.
8. Se le azioni da 6 a 12 hanno avuto esito positivo, il messaggio viene elaborato e viene restituito il risultato del servizio richiamato.

Note:

- In merito agli algoritmi da utilizzare nell'elemento <Signature> rispettivamente <DigestMethod>, <SignatureMethod> e <CanonicalizationMethod> si fa riferimento agli algoritmi indicati nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).
- Un meccanismo simile può essere utilizzato specularmente per l'erogatore.

#### 3.5.2.3 Esempio

Di seguito è riportato un tracciato del messaggio inoltrato dal fruitore all'interfaccia di servizio dell'erogatore relativo ad un servizio di echo.

I namespace utilizzati nel tracciato sono riportati di seguito:

```
soap="http://schemas.xmlsoap.org/soap/envelope/"
wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.
↪0.xsd"
wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.
↪0.xsd"
```

(continues on next page)

(continua dalla pagina precedente)

```
ds="http://www.w3.org/2000/09/xmldsig#"
ec="http://www.w3.org/2001/10/xml-exc-c14n#"
```

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
↪wss-wssecurity-secext-1.0.xsd"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/
↪01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://
↪docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"␣
↪wsu:Id="X509-bf881daf-371a-4d18-9502-d9f92af9a949">MIICqDCCAZ
                                                    ␣
↪CgAwIBAgIEXLSSUTANBgkqhkiG9w0BAQsFADAWMRQwEgYDVQQDDAttb2RpU2VjUHJvZjA
                                                    ␣
↪eFw0xOTA0MTUxNDE2NDlaFw0yNDA0MTUxNDE2NDlaMBYxFDASBgNVBAMMC21vZGlTZWNQ
                                                    ␣
↪cm9mMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvBJNKBiLS+ZcmwGUku5l2
                                                    ␣
↪FKeHogeSZejjOOrO2Ag6DGPXo1MtHt2XwgUXmgT+v0IjhZp5XH2XbwSWw2EMWSG3Zz0CJ
                ILqWGPg0M/LxaIZAxSdxJpVNWg/
↪profO+xKz0B6QHK+I0yecHg7TtI4es9AuDyR4pKslp
                cXyMEqJQ7m5N8v2e4WldeHF2SRN/
↪ereEOuewEi15c7akh4TdkGdiwOSif2AXIugHRgdpH
                jH86iJxFu24IJmBA7C5tytz7mfKollGhI9+2d0902ayVshCV4/
↪pmnX0pDiGayV1C6SDPT
                bapXKXJrp1+fBHaUkDY+W/
↪2Q9sC4o8pttmcpHeMRxFDkwIDAQABMA0GCSqGSIb3DQEBCw
                UAA4IBAQALwKbIm8S2BpYpHaqMwJLeWBPCaDeT7J+KDj39Ac3YxDb8E/
↪hGM+Hn1mq2ssY
                                                    ␣
↪qu5JTvuAQ9o8v3UpcIct15RPgOKYfBzxnH1h2vCavpiFCFTc6UoQgPBZGyyNOOKNOxEnX
                                                    ␣
↪tW7ff1gl2GRLIWXlXDf1fdX7VJVBqWfBvivhIbsDa5LRBCrNsXORx2azUb5QBgMm2UZJx
                YA3+dFRgYmLSY/
↪RyRLf0o03lwCRhAyrU7ya9IMYgrxgjEos2fHB2IGJJ1Wh+gTQWMP+wJ
                ymlC0qyjTHx5pyZOzJGtH5HnaVU7EgtxdBRC9dTlWVpNgmD8nS6Yr/
↪am5cZJZrkIHRyfx
                qkA2W
      </wsse:BinarySecurityToken>
      <wsu:Timestamp wsu:Id="TS-09f1357c-beb4-4804-9410-76c5a06e2e48">
        <wsu:Created>2019-04-15T15:02:15.515Z</wsu:Created>
        <wsu:Expires>2019-04-15T15:07:15.515Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-4d949c5b-
↪968b-4fd5-be67-4cd1d1a41ce3">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
↪c14n#">
            <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-
↪c14n#" PrefixList="soap"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more
↪#rsa-sha256"/>
          <ds:Reference URI="#TS-09f1357c-beb4-4804-9410-76c5a06e2e48">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-
↪exc-c14n#" PrefixList="soap wsse"/>
```

(continues on next page)

(continua dalla pagina precedente)

```
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
          <ds:DigestValue>HPYjNXdxIuJIWk1EArE+8PIgyWt5nAD+upwcjOSDB20=</
↪ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#id-27c23bc8-0c4f-4d98-b046-6e590ea9661b">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-
↪exc-c14n#" PrefixList="soap"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
          <ds:DigestValue>MJzRD4ZRMsFOxskbnfNV9BnDTCLxuLSnmZ8I4IjaxHw=</
↪ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#id-fb4c1fa0-e804-4169-b70e-5b55c5f9d912">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-
↪exc-c14n#" PrefixList="soap"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
          <ds:DigestValue>MIi+ovLTqYu1HqxUtmUnuhVdMmNKOpOX8vn/fKjvQFU=</
↪ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>SBYs6aikHbfsHHV04ifV/ljVTysxNLRTPU6gsOGJamWG
                  YLMPqOETjBf+NFJhPDVdolQSSHw0SD7uA/RlYkE9amRH1K+hoaUIa/
↪PEhPgC1io/LqZdi
                  3rt+b8uRlk+CXcUKOObgf/N960F/sM6s0ArKQxg/Yx6pqWamXBXo0PH/
↪1FvHSgwdA62s0

↪+Sli96qY0EnJPoyKIrqzskiscLXI1jCe8sesyA+xtJ0qBdFKAn2af48sVStPFv4gizC8+
                  bsCRpQ36ihUIlI8DInJ13EgoKV9/
↪rC4PheExO7HvSNTpBFdQt+Wr9wAb3oHq4urRBdugA 6mX2xaJ8/XyZVajivvuVTw==
      </ds:SignatureValue>
      <ds:KeyInfo Id="KI-dab2ce54-b000-439a-bcc2-9b8249626a1c">
        <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/
↪2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" wsu:Id="STR-068909fe-1a64-4cf1-bd5a-355a20b0495f">
          <wsse:Reference URI="#X509-bf881daf-371a-4d18-9502-d9f92af9a949"␣
↪Valuewss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
↪profile-1.0#X509v3"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
  </wsse:Security>
  <Action xmlns="http://www.w3.org/2005/08/addressing">http://profile.security.
↪modi.agid.org/HelloWorld/sayHi
  </Action>
  <MessageID xmlns="http://www.w3.org/2005/08/addressing"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" wsu:Id="id-fb4c1fa0-e804-4169-b70e-5b55c5f9d912">
↪urn:uuid:46d
        a4ec1-f962-4f24-8524-48bb74b505d7
  </MessageID>
  <To xmlns="http://www.w3.org/2005/08/addressing"
```

(continues on next page)

(continua dalla pagina precedente)

```
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" wsu:Id="id-27c23bc8-0c4f-4d98-b046-6e590ea9661b">
↪http://local
            host:8080/security-profile/echo
    </To>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
      <Address>http://www.w3.org/2005/08/addressing/anonymous
      </Address>
    </ReplyTo>
  </soap:Header>
  <soap:Body>
    <ns2:sayHi xmlns:ns2="http://profile.security.modi.agid.org/">
      <arg0>OK</arg0>
    </ns2:sayHi>
  </soap:Body>
</soap:Envelope>
```

Il tracciato rispecchia le seguenti scelte implementative esemplificative:

- riferimento al security token (BinarySecurityToken)
- algoritmi di canonizzazione (CanonicalizationMethod)
- algoritmi di firma (SignatureMethod).
- algoritmo per il digest (DigestMethod)

Le parti, in base alle proprie esigenze, individuano gli specifici algoritmi secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

### 3.5.3 [ID_AUTH_REST_01] Direct Trust con certificato X.509 su REST

Comunicazione tra fruitore ed erogatore che assicuri a livello di messaggio:

- accesso del soggetto fruitore, quale organizzazione o unità organizzativa fruitore, o entrambe le parti.

#### 3.5.3.1 Descrizione

Il presente profilo declina l'utilizzo di:

- JSON Web Token (JWT) definita dall'**RFC 7519**[77]
- JSON Web Signature (JWS) definita dall'**RFC 7515**[78]

Si assume l'esistenza di un trust tra fruitore ed erogatore, che permette il riconoscimento da parte dell'erogatore del certificato X.509, o la CA emittente.

Il meccanismo con cui è stabilito il trust, inclusa la modalità di scambio dei certificati X.509, non condiziona il presente profilo.

Il fruitore inoltra un messaggio all'erogatore includendo o referenziando il certificato X.509 e una porzione significativa del messaggio firmata.

L'erogatore, ricevuto il messaggio, verifica il certificato X.509 e valida la porzione firmata del messaggio, inclusa la corrispondenza del destinatario e l'intervallo di validità della firma. Se la verifica e la validazione sono superate, l'erogatore elabora la richiesta e produce la relativa risposta.

*Figura 5 - Accesso del Fruitore*

[77] https://tools.ietf.org/html/rfc7519.html
[78] https://tools.ietf.org/html/rfc7515.html

#### 3.5.3.2 Regole di processamento

La creazione ed il processamento dei JWT DEVE rispettare le buone prassi di sicurezza indicate in **RFC 8725**[79].

**A: Richiesta**

1. Il fruitore predispone il payload del messaggio (ad esempio un oggetto JSON)
2. Il fruitore, o il soggetto individuato dal trust definito tra erogatore e fruitore, costruisce il JWT popolando:
   (a) il JOSE Header con almeno i parameter:
      i. `alg` con l'algoritmo di firma, vedi **RFC 8725**[80]
      ii. `typ` uguale a JWT
      iii. una o più delle seguenti opzioni per referenziare il certificato X.509:

- `x5u` (X.509 URL)
- `x5c` (X.509 Certificate Chain)
- `x5t#S256` (X.509 Certificate SHA-256 Thumbprint)

   2. il payload del JWT coi claim rappresentativi degli elementi chiave del messaggio, **contenente almeno**:
      (a) i riferimenti temporali di emissione e scadenza: `iat`, `exp`. Se il flusso richiede di verificare l'istante di prima validità del token, si può usare il claim `nbf`.
      (b) il riferimento dell'erogatore in `aud`
3. il fruitore, o il soggetto individuato dal trust definito tra erogatore e fruitore, firma il token adottando la JWS Compact Serialization
4. il fruitore posiziona il JWT nell' **HTTP header Authorization** [81]
5. Il fruitore spedisce il messaggio all'erogatore

**B: Risposta**

6. L'erogatore decodifica il JWT presente in **HTTP header Authorization** [82] secondo le indicazioni contenute in **RFC 7515#section-5.2**[83], le buone prassi indicate in **RFC 8725**[84] e valida i claim contenuti nel JOSE Header, in particolare verifica:
7. il contenuto dei claim `iat` ed `exp`;
8. la corrispondenza tra se stesso e il claim `aud`;
9. L'erogatore recupera il certificato X.509 referenziato nel JOSE Header facendo attenzione alle indicazioni contenute in **RFC 8725#section-3.10**[85]
10. L'erogatore verifica il certificato secondo i criteri del trust
11. L'erogatore valida la firma verificando l'elemento Signature del JWT
12. L'erogatore garantisce l'accesso al fruitore
13. Se le azioni da 6 a 10 hanno avuto esito positivo, il messaggio viene elaborato e viene restituito il risultato del servizio richiamato

Note:

- Gli algoritmi da utilizzare in `alg` sono indicati nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

[79] https://tools.ietf.org/html/rfc8725.html
[80] https://tools.ietf.org/html/rfc8725.html
[81] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
[82] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
[83] https://tools.ietf.org/html/rfc7515.html#section-5.2
[84] https://tools.ietf.org/html/rfc8725.html
[85] https://tools.ietf.org/html/rfc8725.html#section-3.10

- Un meccanismo simile può essere utilizzato specularmente per l'erogatore.
- Per prevenire il rischio di user-enumeration, i messaggi di errore di autenticazione non DEVONO fornire informazioni sull'esistenza o meno dell'utenza.

#### 3.5.3.3 Esempio

Di seguito è riportato un tracciato del messaggio inoltrato dal fruitore all'erogatore.

Esempio porzione messaggio HTTP.

```
GET https://api.erogatore.example/rest/service/v1/hello/echo/Ciao HTTP/1.1
Accept: application/json
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5c.vz8...
```

Esempio porzione JWT

```
# header

{
"alg": "ES256",
"typ": "JWT",
"x5c": [
"MIICyzCCAbOgAwIBAgIEC..."
]
}

# *payload*
{
"iat": 1554382877,
"nbf": 1554382877,
"exp": 1554382879,
"iss": "https://api.fruitore.example",
"sub": "https://api.fruitore.example",
"aud": "https://api.erogatore.example/rest/service/v1/hello/echo"
}
```

Gli elementi presenti nel tracciato rispettano le seguenti scelte implementative e includono:

- l'intervallo temporale di validità, in modo che il JWT possa essere usato solo tra gli istanti `nbf` ed `exp`;
- indica l'istante `iat` di emissione del JWT. Se le parti possono accordarsi nel considerarlo come l'istante iniziale di validità del token, **RFC 7519**[86] non assegna a questo claim nessun ruolo specifico nella validazione, a differenza di `nbf`;
- il riferimento al firmatario del token nel claim aggiuntivo `iss`, che deve essere raccordato con il contenuto del certificato;
- il riferimento al fruitore nel claim aggiuntivo `sub`;
- il destinatario del JWT, che DEVE sempre essere validato;
- contenuto della certificate chain X.509 (`x5c`)
- algoritmi di firma e digest (`alg`).

Le parti, in base alle proprie esigenze, individuano gli specifici algoritmi secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

[86] https://tools.ietf.org/html/rfc7519.html

### 3.5.4 [ID_AUTH_REST_02] Direct Trust con certificato X.509 su REST con unicità del token/messaggio

Il seguente profilo estende il profilo ID_AUTH_REST_01. Comunicazione tra fruitore ed erogatore che assicuri a livello di messaggio:

- accesso del soggetto fruitore, quale organizzazione o unità organizzativa fruitore, o entrambe le parti
- la difesa dalle minacce derivanti dagli attacchi: Replay Attack quando il JWT o il messaggio NON DEVONO essere riprocessati.

#### 3.5.4.1 Descrizione

Il presente profilo declina l'utilizzo di:

- JSON Web Token (JWT) definita dall'**RFC 7519**[87]
- JSON Web Signature (JWS) definita dall'**RFC 7515**[88]

Si assume l'esistenza di un trust tra fruitore (client) ed erogatore (server), che permette il riconoscimento da parte dell'erogatore del certificato X.509, o la CA emittente.

Il meccanismo con cui è stabilito il trust, inclusa la modalità di scambio dei certificati X.509) non condiziona il presente profilo.

Il fruitore inoltra un messaggio all'interfaccia di servizio dell'erogatore includendo o referenziando il certificato X.509 e assicurando la firma dei claim del messaggio.

L'erogatore, ricevuto il messaggio, verifica il certificato X.509 e valida la porzione firmata del messaggio, inclusa la corrispondenza del destinatario e l'intervallo di validità della firma.

L'erogatore verifica inoltre l'univocità dell'identificativo ricevuto nel JWT.

Se la verifica e la validazione sono superate, l'erogatore elabora la richiesta e produce la relativa risposta.

*Figura 6 - Accesso del Fruitore*

#### 3.5.4.2 Regole di processamento

La creazione ed il processamento dei JWT DEVE rispettare le buone prassi di sicurezza indicate in **RFC 8725**[89].

**A: Richiesta**

1. Il fruitore predispone il payload del messaggio (ad esempio un oggetto JSON)
2. Il fruitore, o il soggetto individuato dal trust definito tra erogatore e fruitore, costruisce il JWT popolando:
   (a) il Jose Header con almeno i parameter:
      i. alg con l'algoritmo di firma, vedi **RFC 8725**[90]
      ii. typ uguale a JWT
      iii. una o più delle seguenti opzioni per referenziare il certificato X.509:

- x5u (X.509 URL)
- x5c (X.509 Certificate Chain)
- x5t#S256 (X.509 Certificate SHA-256 Thumbprint)

   2. il payload del JWT coi claim rappresentativi degli elementi chiave del messaggio, **contenente almeno**:

[87] https://tools.ietf.org/html/rfc7519.html
[88] https://tools.ietf.org/html/rfc7515.html
[89] https://tools.ietf.org/html/rfc8725.html
[90] https://tools.ietf.org/html/rfc8725.html

(a) i riferimenti temporali di emissione e scadenza: `iat`, `exp`. Se il flusso richiede di verificare l'istante di prima validità del token, si può usare il claim `nbf`.

(b) il riferimento dell'erogatore in `aud`;

(c) un identificativo univoco del token jti. Se utile alla logica applicativa l'identificativo può essere anche collegato al messaggio.

3. il fruitore, o il soggetto individuato dal trust definito tra erogatore e fruitore firma il token adottando la JWS Compact Serialization
4. il fruitore posiziona il JWT nell' **HTTP header Authorization** [91]
5. Il fruitore spedisce il messaggio all'erogatore.

**B: Risposta**

6. L'erogatore decodifica il JWT presente in **HTTP header Authorization** [92] secondo le indicazioni contenute in **RFC 7515#section-5.2**[93], le buone prassi indicate in **RFC 8725**[94] e valida i claim contenuti nel JOSE Header, in particolare verifica:
   (a) il contenuto dei claim iat ed exp;
   (b) la corrispondenza tra se stesso e il claim aud;
   (c) l'univocità del claim jti
3. L'erogatore recupera il certificato X.509 referenziato nel JOSE Header facendo attenzione alle indicazioni contenute in **RFC 8725#section-3.10**[95]
4. L'erogatore verifica il certificato secondo i criteri del trust
5. L'erogatore valida la firma verificando l'elemento Signature del JWT
6. L'erogatore garantisce l'accesso al fruitore
7. Se le azioni da 6 a 10 hanno avuto esito positivo, il messaggio viene elaborato e viene restituito il risultato del servizio richiamato.

Note:

- In merito agli algoritmi da utilizzare si fa riferimento alle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).
- Un meccanismo simile può essere utilizzato specularmente per l'erogatore.

#### 3.5.4.3 Esempio

Di seguito è riportato un tracciato del messaggio inoltrato dal fruitore all'interfaccia di servizio dell'erogatore.

Esempio porzione pacchetto HTTP.

```
GET https://api.erogatore.example/rest/service/v1/hello/echo/Ciao HTTP/1.1
Accept: application/json
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5c.vz8...
```

Esempio porzione JWT

[91] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
[92] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
[93] https://tools.ietf.org/html/rfc7515.html#section-5.2
[94] https://tools.ietf.org/html/rfc8725.html
[95] https://tools.ietf.org/html/rfc8725.html#section-3.10

```
# *header*
{
        "alg": "ES256",
        "typ": "JWT",
        "x5c": [
                "MIICyzCCAbOgAwIBAgIEC..."
                ]
}

# *payload*
{
        "aud": "https://api.erogatore.example/rest/service/v1/hello/echo"
        "iat": 1516239022,
        "nbf": 1516239022,
        "exp": 1516239024,
        "iss": "https://api.fruitore.example",
        "sub": "https://api.fruitore.example",
        "jti": "065259e8-8696-44d1-84c5-d3ce04c2f40d"
}
```

Gli elementi presenti nel tracciato rispettano le seguenti scelte implementative e includono:

- l'intervallo temporale di validità, in modo che il JWT possa essere usato solo tra gli istanti `nbf` ed `exp`;
- indica l'istante iat di emissione del JWT. Se le parti possono accordarsi nel considerarlo come l'istante iniziale di validità del token, **RFC 7519**[96] non assegna a questo claim nessun ruolo specifico nella validazione, a differenza di `nbf`;
- il riferimento al firmatario del token nel claim aggiuntivo `iss`, che deve essere raccordato con il contenuto del certificato;
- il riferimento al fruitore nel claim aggiuntivo `sub`;
- il destinatario del JWT, che DEVE sempre essere validato;
- contenuto della certificate chain X.509 (`x5c`)
- algoritmi di firma e digest (`alg`).

Le parti, in base alle proprie esigenze, individuano gli specifici algoritmi secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

## 3.6 Integrità

Di seguito le indicazioni per le tecnologie accolte dal ModI.

L'AgID assicura l'aggiornamento degli stessi per soddisfare le esigenze espresse dalle PA.

### 3.6.1 [INTEGRITY_SOAP_01] Integrità del payload del messaggio SOAP

Il presente profilo estende ID_AUTH_SOAP_01 o ID_AUTH_SOAP_02, aggiungendo alla comunicazione tra fruitore ed erogatore a livello di messaggio:

- integrità del payload del messaggio.

[96] https://tools.ietf.org/html/rfc7519.html

#### 3.6.1.1 Descrizione

Il presente profilo specializza lo standard OASIS Web Services Security X.509 Certificate Token Profile Versione 1.1.1.

Si assume l'esistenza di un trust tra fruitore ed erogatore, che permette il riconoscimento da parte dell'erogatore del certificato X.509, o la CA emittente.

Il meccanismo con cui è stabilito il trust non condiziona il presente profilo.

Il fruitore inoltra un messaggio all'interfaccia di servizio dell'erogatore includendo o referenziando il certificato X.509 e la firma del payload del messaggio.

L'erogatore, ricevuto il messaggio, verifica il certificato X.509 e valida l'integrità del payload del messaggio firmato. Se la verifica e la validazione sono superate, l'erogatore elabora la richiesta e produce la relativa risposta.

*Figura 7 - Integrità del payload del messaggio*

#### 3.6.1.2 Regole di processamento

**A: Richiesta**

1. Il fruitore costruisce un messaggio SOAP per il servizio.
2. Il fruitore calcola la firma del payload del messaggio usando l'XML Signature. Il digest è firmato usando la chiave privata associata al certificato X.509 del fruitore. L'elemento <Signature> è posizionato nell'header <Security> del messaggio.
3. Il fruitore referenzia il certificato X.509 usando in maniera alternativa, nell'header <Security>, i seguenti elementi previsti nella specifica ws-security:
   (a) <wsse:BinarySecurityToken>
   (b) <wsse:KeyIdentifier>
   (c) <wsse:SecurityTokenReference>
4. Il fruitore spedisce il messaggio all'interfaccia di servizio dell'erogatore.

**B: Risultato**

5. L'erogatore recupera il certificato X.509 referenziato nell'header <Security>.
6. L'erogatore verifica il certificato secondo i criteri del trust.
7. L'erogatore valida la firma verificando l'elemento <Signature> nell'header <Security>.
8. Se il certificato è valido anche per identificare il soggetto fruitore, l'erogatore autentica lo stesso
9. Se le azioni da 5 a 8 hanno avuto esito positivo, il messaggio viene elaborato e viene restituito il risultato del servizio richiamato

Note:

- Per quanto riguarda gli algoritmi da utilizzare nell'elemento <Signature> rispettivamente <DigestMethod> , <SignatureMethod> e <CanonicalizationMethod> si fa riferimento agli algoritmi indicati nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).
- Un meccanismo simile può essere utilizzato per garantire l'integrità del payload del messaggio risposta dell'erogatore al fruitore.

#### 3.6.1.3 Esempio

Di seguito è riportato un tracciato del messaggio inoltrato dal fruitore all'interfaccia di servizio dell'erogatore.

I namespace utilizzati nel tracciato sono riportati di seguito:

```
soap="http://schemas.xmlsoap.org/soap/envelope/"
wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.
↪0.xsd"
ds="http://www.w3.org/2000/09/xmldsig#"
ec="http://www.w3.org/2001/10/xml-exc-c14n#"
```

```
<?xml version="1.0"?>
<soap:Envelope>
  <soap:Header>
    <wsse:Security soap:mustUnderstand="1">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/
↪01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://
↪docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x5 09-token-profile-1.0#X509v3"␣
↪wsu:Id="X509-44680ddc-e35a-4374-bcbf-2b6dcba722d7">MIICyzCCAb␣
↪OgAwIBAgIECxY+9TAhkiG9w...</wsse:BinarySecurityToken>
      <ds:Signature Id="SIG-f58c789e-e3d3-4ec3-9ca7-d1e9a4a90f90">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
↪c14n#">
            <ec:InclusiveNamespaces PrefixList="soap"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more
↪#rsa-sha256"/>
          <ds:Reference URI="#bd-567d101-aed1-789e-81cb-5ae1c5dbef1a">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="soap"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <ds:DigestValue>0cJNCJ1W8Agu66fGTXlPRyy0EUNUQ9OViFlm8qf8Ysw=* *</
↪ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>AIrDa7ukDfFJD867goC+c7K3UampxpX/Nj/...</
↪ds:SignatureValue>
        <ds:KeyInfo Id="KI-cad9ee47-dec8-4340-8fa1-74805f7e26f8">
          <wsse:SecurityTokenReference wsu:Id="STR-e193f25f-9727-4197-b7aa-
↪25b01c9f2ba3">
            <wsse:Reference URI="#X509-44680ddc-e35a-4374-bcbf-2b6dcba722d7"␣
↪ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
↪profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
↪wssecurity-utility-1.0.xsd" wsu:id="bd-567d101-aed1-789e-81cb-5ae1c5dbef1a">
    <ns2:sayHi xmlns:ns2="http://example.profile.security.modi.agid.gov.it/">
      <arg0>Hello World!</arg0>
    </ns2:sayHi>
  </soap:Body>
</soap:Envelope>
```

Il codice rispecchia alcune scelte implementative esemplificative in merito:

- riferimento al security token (BinarySecurityToken)
- algoritmi di canonizzazione (CanonicalizationMethod)
- algoritmi di firma (SignatureMethod)
- algoritmo per il digest (DigestMethod)

Le parti, in base alle proprie esigenze, individuano gli specifici algoritmi secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

### 3.6.2 [INTEGRITY_REST_01] Integrità del payload messaggio REST

Il presente profilo estende ID_AUTH_REST_01 o ID_AUTH_REST_02, aggiungendo alla comunicazione tra fruitore ed erogatore a livello di messaggio:

- integrità del payload del messaggio

Si adottano le indicazione riportate in **RFC 7231**[97]. Considereremo sempre richieste e risposte complete, con i metodi standard definiti in RFC 7231#section-4.

Questo scenario non copre quindi Range Requests **RFC 7233**[98] o HTTP method PATCH che trasmette una rappresentazione parziale.

#### 3.6.2.1 Descrizione

Il presente profilo propone l'utilizzo di:

- semantica HTTP **RFC 7231**[99];
- Digest HTTP header **RFC 3230**[100] per l'integrità della rappresentazione della risorsa;
- JSON Web Token (JWT) definita dall' **RFC 7519**[101];
- JSON Web Signature (JWS) definita dall' **RFC 7515**[102].

Si assume l'esistenza di un trust tra fruitore ed erogatore, che permette il riconoscimento da parte dell'erogatore del certificato X.509, o la CA emittente.

Il meccanismo con cui è stabilito il trust non condiziona il presente profilo.

*Figura 8 - Integrità del payload del messaggio*

#### 3.6.2.2 Regole di processamento

La creazione ed il processamento dei JWT DEVE rispettare le buone prassi di sicurezza indicate in **RFC 8725**[103].

**A: Richiesta**

1. Il fruitore predispone il body del messaggio (ad esempio un oggetto JSON)
2. Il fruitore calcola il valore del Digest header dei representation data secondo le indicazioni in **RFC 3230**[104]
3. Il fruitore individua l'elenco degli HTTP Header da firmare, incluso Digest e se presenti **HTTP header Content-Type** [105] e HTTP header Content-Encoding
4. Il fruitore crea la struttura o la stringa da firmare in modo che includa gli http header da proteggere, i riferimenti temporali di validità della firma e degli estremi della comunicazione, ovvero:
   (a) il JOSE Header con almeno i parameter:
      i. alg con l'algoritmo di firma, vedi **RFC 8725**[106]

[97] https://tools.ietf.org/html/rfc7231.html
[98] https://tools.ietf.org/html/rfc7233.html
[99] https://tools.ietf.org/html/rfc7231.html
[100] https://tools.ietf.org/html/rfc3230.html
[101] https://tools.ietf.org/html/rfc7519.html
[102] https://tools.ietf.org/html/rfc7515.html
[103] https://tools.ietf.org/html/rfc8725.html
[104] https://tools.ietf.org/html/rfc3230.html
[105] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type
[106] https://tools.ietf.org/html/rfc8725.html

ii. typ uguale a JWT

iii. una o più delle seguenti opzioni per referenziare il certificato X.509:

- `x5u` (X.509 URL)
- `x5c` (X.509 Certificate Chain)
- `x5t#S256` (X.509 Certificate SHA-256 Thumbprint)

(b) i seguenti claim obbligatori:

i. i riferimenti temporali di emissione e scadenza: `iat` , `exp`. Se il flusso richiede di verificare l'istante di prima validità del token, si può usare il claim `nbf`.

ii. il riferimento dell'erogatore in `aud`;

(c) i seguenti claim, secondo la logica del servizio:

i. `sub`: oggetto (principal see **RFC 3744#section-2**[107]) dei claim contenuti nel jwt

ii. `iss`: identificativo del mittente

iii. `jti`: identificativo del JWT, per evitare replay attack

(d) il claim signed_headers con gli header http da proteggere ed i rispettivi valori, ovvero:

i. **HTTP header Digest** [108]

ii. **HTTP header Content-Type** [109]

iii. **HTTP header Content-Encoding** [110]

9. il fruitore firma il token adottando la JWS Compact Serialization
10. il fruitore posiziona il JWS nell'header Agid-JWT-Signature
11. Il fruitore spedisce il messaggio all'erogatore.

**B: Risultato**

8. L'erogatore decodifica il JWS presente in Agid-JWT-Signature header secondo le indicazioni contenute in **RFC 7515#section-5.2**[111], le buone prassi indicate in **RFC 8725**[112] e valida i claim contenuti nel Jose Header, in particolare verifica:

(a) il contenuto dei claim `iat` , `exp`;

(b) la corrispondenza tra se stesso e il claim `aud`;

(c) l'univocità del claim `jti` se presente.

5. L'erogatore recupera il certificato X.509 referenziato nel JOSE Header facendo attenzione alle indicazioni contenute in **RFC 8725#section-3.10**[113]
6. L'erogatore verifica il certificato secondo i criteri del trust
7. L'erogatore valida la firma verificando l'elemento Signature del JWS
8. L'erogatore verifica la corrispondenza tra i valori degli header passati nel messaggio e quelli presenti nel claim signed-header, Content-Type e Content-Encoding se presenti devono essere sempre firmati, come indicato nel punto A3
9. L'erogatore quindi verifica la corrispondenza tra Digest ed il payload body ricevuto
10. Se le azioni da 8 a 13 hanno avuto esito positivo, il messaggio viene elaborato e viene restituito il risultato del servizio richiamato.

[107] https://tools.ietf.org/html/rfc3744.html#section-2
[108] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Digest
[109] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type
[110] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Encoding
[111] https://tools.ietf.org/html/rfc7515.html#section-5.2
[112] https://tools.ietf.org/html/rfc8725.html
[113] https://tools.ietf.org/html/rfc8725.html#section-3.10

Note:

- Per gli algoritmi da utilizzare in alg e Digest si vedano le Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).
- Un meccanismo simile può essere utilizzato per garantire l'integrità della risposta da parte dell'erogatore al fruitore. In questo caso si ricorda che Digest fa riferimento al checksum del payload body della selected representation. Per una richiesta con **HTTP method HEAD** [114] il server DEVE ritornare il checksum dell'ipotetico payload body ritornato dalla corrispondente richiesta con **HTTP method GET** [115].

#### 3.6.2.3 Esempio

Di seguito è riportato un tracciato del messaggio inoltrato dal fruitore all'interfaccia di servizio dell'erogatore.

Richiesta HTTP con Digest e representation metadata

```
POST https://api.erogatore.example/rest/service/v1/hello/echo/ HTTP/1.1
Accept: application/json
Agid-JWT-Signature: eyJhbGciOiJSUzI1NiIsInR5c.vz8...
Digest: SHA-256=cFfTOCesrWTLVzxn8fmHl4AcrUs40Lv5D275FmAZ96E=
Content-Type: application/json

{"testo": "Ciao mondo"}
```

Porzione JWS con campi protetti dalla firma

```
# *header*
{
  "alg": "ES256",
  "typ": "JWT",
  "x5c": [
        "MIICyzCCAbOgAwIBAgIEC..."
  ]
}
# *payload*

{
  "aud": "https://api.erogatore.example/rest/service/v1/hello/echo"
  "iat": 1516239022,
  "nbf": 1516239022,
  "exp": 1516239024,
  "signed_headers": [
    {"digest": "SHA-256=cFfTOCesrWTLVzxn8fmHl4AcrUs40Lv5D275FmAZ96E="},
    {"content-type": "application/json"}
  ],
}
```

Il tracciato rispecchia alcune scelte implementative esemplificative in merito:

- include tutti gli elementi del JWS utilizzati in ID_AUTH_REST_02
- mette in minuscolo i nomi degli header firmati
- utilizza il claim custom signed_headers contenente una lista di json objects per supportare la firma di più header ed eventualmente verificare il loro ordinamento

Le parti, in base alle proprie esigenze, individuano gli specifici algoritmi secondo quanto indicato nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale).

[114] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD
[115] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

CAPITOLO 4

# Documento operativo - Profili di interoperabilità

## 4.1 Introduzione

I profili di interoperabilità individuano combinazioni dei pattern di interazione, indicati nel Documento Operativo - Pattern di interazione, e pattern di sicurezza, indicati Documento Operativo - Pattern di sicurezza, che risolvono una esigenza specifica della comunicazione tra fruitore ed erogatore.

I profili di interoperabilità sono scelti dall'erogatore in funzione alle specifiche esigenze applicative ed in relazione alla natura dei fruitori.

Data la variabilità nel tempo delle esigenze delle amministrazioni e delle tecnologie abilitanti, nonché considerata la natura incrementale del ModI, l'elenco dei profili non è da intendersi esaustivo. Nel caso in cui un'amministrazione abbia esigenze non ricoperte nei seguenti profili DEVE informare AgID, nei modi indicati nel capitolo 7 «Pattern e profili di interoperabilità» delle Linee di indirizzo sull'interoperabilità tecnica delle Pubbliche Amministrazioni. Le tecnologie e standard per assicurare la sicurezza dell'interoperabilità tramite API utilizzabili nel ModI, tra cui OAuth 2.0, sono individuate nelle Linee Guida sulla sicurezza, emanate dall'Agenzia per l'Italia Digitale ai sensi dell'articolo 71 del decreto legislativo 7 marzo 2005, n. 82 (Codice dell'Amministrazione Digitale)

## 4.2 Ambito di applicazione

Il presente Documento operativo è redatto quale documento operativo relativo alla Linee Guida sull'interoperabilità tecnica.

### 4.2.1 Soggetti destinatari

Il Documento Operativo è destinato ai soggetti di cui all'articolo 2, comma 2 del CAD, così come indicato dall'articolo 75 dello stesso. I destinatari la attuano nella realizzazione dei propri sistemi informatici che fruiscono o erogano dati e/o servizi digitali ad altri soggetti.

Il Documento Operativo è rivolto ai soggetti privati che devono interoperare con la Pubblica Amministrazione per erogare o fruire di dati e servizi tramite sistemi informatici

## 4.3 Riferimenti e sigle

### 4.3.1 Note di lettura del documento

Conformemente alle norme ISO/IEC Directives, Part 3 per la stesura dei documenti tecnici le presenti Linee Guida utilizzano le parole chiave «DEVE», «DEVONO», «NON DEVE», «NON DEVONO», «DOVREBBE», «NON DOVREBBE», «PUÒ» e «OPZIONALE», la cui interpretazione è descritta di seguito.

- **DEVE** o **DEVONO**, indicano un requisito obbligatorio per rispettare le Linee Guida;
- **NON DEVE** o **NON DEVONO**, indicano un assoluto divieto delle specifiche;
- **DOVREBBE** o **NON DOVREBBE**, indicano che le implicazioni devono essere comprese e attentamente pesate prima di scegliere approcci alternativi;
- **PUÒ** o **POSSONO** o l'aggettivo **OPZIONALE**, indica che il lettore può scegliere di applicare o meno senza alcun tipo di implicazione o restrizione la specifica.

### 4.3.2 Standard di riferimento

Sono riportati di seguito gli standard tecnici indispensabili per l'applicazione del presente documento.

Tabella 4.1: Riferimenti Normativi

| | |
|---|---|
| [X.509] | Standard dell'Unione Internazionale delle telecomunicazioni (ITU-T), che definisce definire il formato dei certificati a chiave pubblica e delle autorità di certificazione |

### 4.3.3 Termini e definizioni

Tabella 4.2: Termini e definizioni

| | |
|---|---|
| [AgID] | Agenzia per l'Italia Digitale |
| [CAD] | Codice Amministrazione Digitale, D.lgs. 7 marzo 2005, n. 82 |
| [PA] | Pubblica Amministrazione |
| [UML] | Linguaggio di modellazione unificato (Unified Modeling Language) |
| [RPC] | Remote procedure call |
| [SOAP] | Simple Object Access Protocol |
| [REST] | Representational State Transfer |

## 4.4 Profili di interoperabilità

Di seguito le indicazioni per le tecnologie accolte dal ModI.

L'AgID assicura l'aggiornamento degli stessi per soddisfare le esigenze espresse dalle PA.

### 4.4.1 [PROFILE_CONF_ID_AUTH_01] Profilo per confidenzialità ed autenticazione del fruitore

Dare seguito ad uno scambio tra fruitore ed erogatore che garantisca:

- la confidenzialità a livello di canale
- l'autenticazione del fruitore

Il fruitore potrebbe non coincidere con l'unità organizzativa fruitore, ma comunque appartenere alla stessa.

Questo profilo è indipendente dal pattern di interazione implementato ed utilizza i seguenti pattern di sicurezza:

- ID_AUTH_CHANNEL_01
- ID_AUTH_SOAP_01 o ID_AUTH_REST_01

Si assume l'esistenza di un trust tra fruitore ed erogatore che stabilisce:

- riconoscimento da parte dell'erogatore dei certificati X.509, o la CA emittente, relative al fruitore
- riconoscimento da parte del fruitore del certificato X.509, o la CA emittente, relative al soggetto erogatore

Il meccanismo con cui è stabilito il trust non condiziona quanto descritto di seguito.

#### 4.4.1.1 Flusso delle interazioni

**A: Richiesta**

Il messaggio di richiesta viene predisposto utilizzando il pattern [ID_AUTH_SOAP_01] nel caso di utilizzo di SOAP o [ID_AUTH_REST_01] nel caso di utilizzo di REST, per garantire:

- l'identità del fruitore.

Il fruitore invia il messaggio di richiesta all'interfaccia di servizio dell'erogatore.

Il messaggio viene trasmesso su un canale sicuro utilizzando il profilo ID_AUTH_CHANNEL_01 per garantire:

- la confidenzialità a livello di canale.

**B: Risposta**

L'erogatore da seguito a quanto previsto nel pattern ID_AUTH_SOAP_01 nel caso di utilizzo di SOAP o ID_AUTH_REST_01 nel caso di utilizzo di REST.

### 4.4.2 [PROFILE_NON_REPUDIATION_01] Profilo per la non ripudiabilità della trasmissione

Dare seguito ad uno scambio tra fruitore ed erogatore che garantisca la non ripudiabilità assicurando a livello di messaggio:

- integrità del messaggio
- autenticazione del fruitore, quale organizzazione o unità organizzativa fruitore quale mittente del contenuto
- conferma da parte dell'erogatore della ricezione del contenuto
- opponibilità ai terzi
- robustezza della trasmissione

Il presente profilo utilizza come modello di comunicazione il Pattern di interazione BLOCK_SOAP nel caso di utilizzo di SOAP o BLOCK_REST nel caso di utilizzo di REST. Questo profilo utilizza i seguenti pattern di sicurezza:

- ID_AUTH_CHANNEL_01 o in alternativa ID_AUTH_CHANNEL_02
- per SOAP: ID_AUTH_SOAP_02 e INTEGRITY_SOAP_01
- per REST: ID_AUTH_REST_02 e INTEGRITY_REST_01

Si assume l'esistenza di un trust tra fruitore ed erogatore che stabilisce:

- reciproco riconoscimento da parte dell'erogatore e del fruitore dei certificati X.509, o le CA emittenti.
- Il meccanismo con cui è stabilito il trust non condiziona quanto descritto nella sezione. Fruitore ed erogatore devono concordare:

  - un identificativo univoco del messaggio, necessario a garantire il riscontro di ritrasmissioni (vedi ID_AUTH_SOAP_02 e ID_AUTH_REST_02), e le relative modalità di scambio;
  - l'arco temporale di persistenza dei messaggi, che dipende dalle caratteristiche del contenuto dei dati scambiati e dal rispetto delle norme di legge.
  - il tempo di validità della transazione che intercorre tra:
    * l'istante di inoltro del fruitore
    * l'istante di ricezione dell'erogatore;
  - il tempo massimo di attesa del fruitore del messaggio di risposta per ritenere la comunicazione non avvenuta;
  - il numero massimo di tentativi di rinvio da parte del fruitore accettati dall'erogatore;
  - eventuale utilizzo di canali alternativi per superare o evidenziare problemi di comunicazione riscontrati.

Attraverso le tecnologie di criptazione sono garantite le seguenti proprietà:

- integrità e non ripudio del messaggio inviato dal fruitore
- integrità e non ripudio del messaggio di conferma da parte dell'erogatore
- autenticazione del fruitore
- autenticazione dell'erogatore
- validazione temporale che certifichi l'istante in cui il messaggio è stato trasmesso
- validazione temporale che certifichi l'istante in cui il messaggio è stato ricevuto.

#### 4.4.2.1 Flusso delle interazioni

*Figura 1 - Non ripudiabilità della trasmissione*

**A: Verifica numero tentativi di inoltro**

Il fruitore realizza una delle seguenti azioni:

*A.1 [Primo Invio]*

Il fruitore inizializza il numero di tentativi di inoltro ad 1 e prosegue a quanto indicato al passo B.

*A.2 [Invio Successivo con numero di tentativi inferiore al massimo pattuito]*

Il fruitore incrementa il numero di tentativi di inoltro e da seguito a quanto indicato al passo B.

*A.3 [Superamento numero di tentativi massimi pattuiti]*

Il fruitore utilizza i canali alternativi per superare o evidenziare problemi di comunicazione riscontrati non proseguendo con i passi successivi.

**B: Richiesta**

Il messaggio di richiesta viene costruito aggiungendo un identificativo univoco del messaggio (vedi [ID_AUTH_SOAP_02] o [ID_AUTH_REST_02]), l'istante di trasmissione

- SOAP: <wsu:Timestamp> della ws-security
- REST: claim iat contenuta nel payload del token JWT

Tutti gli elementi utili al non ripudio, inclusi quelli descritti in ID_AUTH_SOAP_02 o ID_AUTH_REST_02, vengono firmati utilizzando il profilo desiderato INTEGRITY_SOAP_01 o INTEGRITY_REST_01 per garantire:

- l'integrità del contenuto
- l'identità del mittente
- il momento di invio.

Il fruitore invia il messaggio di richiesta all'interfaccia di servizio dell'erogatore. Il messaggio viene trasmesso su un canale sicuro per garantire:

- la confidenzialità a livello di canale utilizzando i pattern ID_AUTH_CHANNEL_01 o in alternativa ID_AUTH_CHANNEL_02.

**C. Persistenza erogatore**

Per garantire la non ripudiabilità del messaggio ricevuto dal fruitore, così come previsto dai profili utilizzati:

- L'erogatore provvede all'autenticazione del fruitore;
- L'erogatore verifica l'integrità del messaggio firmato. Inoltre la presenza dell'istante di trasmissione nel messaggio ne garantisce validità a lungo termine.

Per assicurare l'opponibilità a terzi:

- L'erogatore rende persistente il messaggio firmato tracciando l'istante di ricezione.

La persistenza del messaggio:

- DEVE garantire la capacità di ricercare ed esportare le informazioni memorizzate;
- DEVE essere garantita per un periodo di tempo che dipende dagli accordi tra le parti.

L'erogatore realizza una delle seguenti azioni:

*C.1 [Prima Ricezione]*

L'erogatore inizializza il numero di tentativi di richieste ricevute ad 1 e prosegue al passo D.

*C.2 [Duplicato con numero di tentativi inferiore al massimo pattuito]*

L'erogatore rileva la presenza di un identificativo univoco del messaggio già ricevuto, a causa di una mancata ricezione del messaggio di conferma da parte del fruitore. Incrementa il numero di tentativi di richieste ricevute e prosegue al passo D.

*C.3 [Superamento numero massimo di tentativi pattuiti]*

L'erogatore rileva la presenza di un identificativo univoco del messaggio già ricevuto, a causa di una mancata ricezione del messaggio di conferma da parte del fruitore.

L'erogatore rileva di aver raggiunto il numero massimo di tentativi di richieste ricevute. L'erogatore utilizza i canali alternativi per superare o evidenziare problemi di comunicazione riscontrati non proseguendo con i passi successivi.

**D: Risposta**

L'erogatore costruisce un messaggio di conferma includendo un identificativo che permetta di associare univocamente al messaggio di richiesta (ad esempio il digest presente nel messaggio di richiesta) e l'istante di trasmissione.

Inoltre al messaggio di conferma viene aggiunto l'istante di trasmissione:

- SOAP: <wsu:Timestamp> della ws-security
- REST: claim iat contenuta nel payload del token JWT

Tutti gli elementi utili al non ripudio, inclusi quelli descritti in ID_AUTH_SOAP_02 o ID_AUTH_REST_02, vengono firmati utilizzando il profilo desiderato INTEGRITY_SOAP_01 o INTEGRITY_REST_01 per garantire:

- l'integrità del contenuto
- l'identità del mittente
- il momento di invio

**E: Persistenza Richiedente**

Per garantire la non ripudiabilità del messaggio inviato all'erogatore:

- Il fruitore provvede all'autenticazione dell'erogatore rispetto al messaggio di risposta.

- Il fruitore verifica l'integrità del messaggio di risposta firmato in cui la presenza del timestamp sul protocollo di messaggio ne garantisce validazione a lungo termine e il tempo di ricezione.

Per assicurare l'opponibilità a terzi:

- Il fruitore rende persistente il messaggio di risposta firmato.

La persistenza del messaggio:

- DEVE garantire la capacità di ricercare ed esportare le informazioni memorizzate;
- DEVE essere garantita per un periodo di tempo che dipende dagli accordi tra le parti.

Note:

Nel caso in cui il fruitore non riceve il messaggio di risposta entro i termini concordati tra le parti, si ritiene la comunicazione non conclusa, in quanto può presentarsi uno dei seguenti casi:

- il messaggio di richiesta non ha raggiunto l'erogatore
- il messaggio di richiesta ha raggiunto l'erogatore ma il fruitore non ha ricevuto il messaggio di risposta.

In queste situazioni il fruitore riesegue il passo A.

CAPITOLO 5

# Documento operativo - Raccomandazioni di implementazione

## 5.1 Introduzione

Il presente documento operativo riporta le indicazioni che gli erogatori considerano nell'implementazione delle API al fine di favorire l'interoperabilità con i fruitori.

Le raccomandazioni sono applicate dagli erogatori in funzione alle specifiche esigenze applicative ed in relazione alla natura dei fruitori.

## 5.2 Ambito di applicazione

Il presente Documento operativo è redatto quale documento operativo relativo alla Linee Guida sull'interoperabilità tecnica.

### 5.2.1 Soggetti destinatari

Il Documento Operativo è destinato ai soggetti di cui all'articolo 2, comma 2 del CAD, così come indicato dall'articolo 75 dello stesso. I destinatari la attuano nella realizzazione dei propri sistemi informatici che fruiscono o erogano dati e/o servizi digitali di/ad altri soggetti.

Per i servizi implementati dagli erogatori prima dell'emanazione del Documento Operativo, al fine di assicurare la convergenza al ModI, si richiede di:

- assicurare per i nuovi fruitori l'applicazione di modalità di fruizione conformi al Documento Operativo;
- prevedere, a valle di una valutazione di impatto che includa l'effetto sui fruitori, la dismissione delle modalità difformi al Documento Operativo.

Il Documento Operativo è rivolto ai soggetti privati che devono interoperare con la Pubblica Amministrazione per erogare o fruire di dati e servizi tramite sistemi informatici.

## 5.3 Riferimenti e sigle

### 5.3.1 Note di lettura del documento

Conformemente alle norme ISO/IEC Directives, Part 3 per la stesura dei documenti tecnici le presenti Linee Guida utilizzano le parole chiave «DEVE», «DEVONO», «NON DEVE», «NON DEVONO», «DOVREBBE», «NON DOVREBBE», «PUÒ» e «OPZIONALE», la cui interpretazione è descritta di seguito.

- **DEVE** o **DEVONO**, indicano un requisito obbligatorio per rispettare le Linee Guida;
- **NON DEVE** o **NON DEVONO**, indicano un assoluto divieto delle specifiche;
- **DOVREBBE** o **NON DOVREBBE**, indicano che le implicazioni devono essere comprese e attentamente pesate prima di scegliere approcci alternativi;
- **PUÒ** o **POSSONO** o l'aggettivo **OPZIONALE**, indica che il lettore può scegliere di applicare o meno senza alcun tipo di implicazione o restrizione la specifica.

### 5.3.2 Standard di riferimento

Sono riportati di seguito gli standard tecnici indispensabili per l'applicazione del presente documento.

Tabella 5.1: Riferimenti Normativi

| | |
|---|---|
| [IEEE 754] | Standard per il calcolo in virgola mobile |
| [ISO 639] | Standard che elenca i codici brevi per l'individuazione delle maggiori lingue |
| [ISO 3166] | Standard che definisce i codici per i nomi dei Paesi |
| [ISO 4217] | Standard internazionale per identificare le valute |
| [ISO 8601] | Standard per una rappresentazione numerica di date e orari |
| [ISO 60559] | Standard per la aritmetica a virgola mobile |
| [RFC3339] | Date and Time on the Internet |
| [RFC6838] | Media Type Specifications and Registration Procedures |
| [RFC8259] | JavaScript Object Notation (JSON) Data Interchange |
| [RFC7231] | Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content |
| [RFC7232] | Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests |
| [RFC7234] | Hypertext Transfer Protocol (HTTP/1.1): Caching |
| [RFC7807] | REST Problem Details |
| [RFC8288] | Web Linking |

### 5.3.3 Termini e definizioni

Tabella 5.2: Termini e definizioni

| | |
|---|---|
| [AgID] | Agenzia per l'Italia Digitale |
| [API] | Application programming interface |
| [CAD] | Codice Amministrazione Digitale, D.Lgs 7 marzo 2005, n. 82 |
| [HTTP] | Hypertext Transfer Protocol |
| [IDL] | Interface Description Language |
| [JSON] | JavaScript Object Notation |
| [ModI] | Modello di Interoperabilità |
| [MTOM] | Message Transmission Optimization Mechanism |
| [OPENAPI] | Specifica per gestire servizi web RESTful |
| [REST] | Representational State Transfer |
| [SOAP] | Simple Object Access Protocol |
| [SWA] | SOAP with Attachments |
| [URI] | Uniform Resource Identifier |
| [UTC] | Universal Time Coordinated |
| [UTF-8] | Unicode Transformation Format, 8 bit |
| [XOP] | XML-binary Optimized Packaging |
| [W3C] | World Wide Web Consortium |
| [WSDL] | Web Services Description Language |
| [WS-I] | Web service Basic Profile versione 2 |

## 5.4 Raccomandazioni tecniche generali

In quanto segue sono riportate le raccomandazione tecniche che gli erogatori considerano nell'implementazione delle proprie API.

### 5.4.1 Raccomandazioni globali

#### 5.4.1.1 [RAC_GEN_001] Descrizione delle API

Le API DEVONO essere rappresentate mediante un Interface Description Language standard (IDL). Nello specifico:

- per REST, OpenAPI 3.0 e successive;
- per SOAP, WSDL 1.1 e successive.

#### 5.4.1.2 [RAC_GEN_002] Endpoint delle API

Il numero di versione NON DEVE essere presente all'interno del nome della API.

Si DOVREBBE indicare il numero di versione e la tecnologia nell'endpoint delle API.

Esempio:

```
https://<dominioOrganizzativo>/[rest|soap]/<DominioApplicativo>/v<major>[.<minor>[.
↪<patch>]]/<NomeAPI>
```

dove:

- <dominioOrganizzativo> indica l'organizzazione che espone il servizio;
- [rest|soap] indica la tecnologia della API;
- <DominioApplicativo> indica il settore all'interno dell'organizzazione;

- v<major>[.<minor>[.<patch>]] indica il numero di versione in coerenza con Semantic Versioning 2.0.0[1];
- <NomeAPI> è il nome della specifica API.

Il documento di specifica dell'API DEVE indicare la versione, includendo `<major>.<minor>.<patch>`.

Listato 5.1: Un esempio di versioning in formato OAS3

```
openapi: 3.0.3
info:
  version: 1.3.4
...
```

#### 5.4.1.3 [RAC_GEN_003] Codifica di default

Si DOVREBBE utilizzare UTF-8 come codifica di default per i dati.

#### 5.4.1.4 [RAC_GEN_004] Non passare credenziali o dati riservati nell'URL

Eventuali dati riservati o credenziali e token di autenticazione NON DEVONO essere passati nei query parameters o comunque nell'URL.

### 5.4.2 Raccomandazioni sul formato dei dati

#### 5.4.2.1 [RAC_GEN_FORMAT_001] Utilizzare Content-Type semanticamente coerenti

Quando si ritornano dati binari, immagini o documenti (eg. pdf, png, ...) si DEVONO utilizzare i rispettivi Content-Type. Nel protocollo HTTP, l'**HTTP header Content-Type** [116] indica il media-type di una risorsa.

#### 5.4.2.2 [RAC_GEN_FORMAT_002] Evitare Content-Type personalizzati

Si DOVREBBE evitare l'uso di media-type personalizzati come da **RFC 6838#section-3.4**[117] (eg. application/x.custom.name+xml, application/x.custom.name+json) ed utilizzare nomi standard come:

- application/xml
- application/soap+xml
- application/json
- application/problem+json
- application/jose+json

L'elenco dei media-type è reperibile sul sito IANA[118].

#### 5.4.2.3 [RAC_GEN_FORMAT_003] Formati standard per Data ed Ora

Le date DEVONO essere conformi:

- alla sintassi «full-date» indicata in **RFC 3339**[119], ad esempio 2015-05-28 se si indica una data;

[1] Cfr. https://semver.org/
[116] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type
[117] https://tools.ietf.org/html/rfc6838.html#section-3.4
[118] https://www.iana.org/assignments/media-types/media-types.xhtml
[119] https://tools.ietf.org/html/rfc3339.html

- alla sintassi «date-time» indicata in **RFC 3339**[120], ad esempio 2015-05-28T14:07:17Z o nel formato UNIX Timestamp definito in «The Open Group Base Specifications Issue 7, Rationale: Base Definitions, section A.4 General Concepts» se si indica un momento esatto nel tempo.

**RFC 3339**[121] permette di indicare una timezone prefissando la data con la distanza da UTC:

- 2015-05-28T14:07:17+01:00
- 2015-05-28T14:07:17-05:00

Quando la data è specificata in UTC occorre utilizzare sempre il suffisso Z (Zulu time zone):

- 2015-05-28T14:07:17Z

Ove non dettagliato nelle specifiche, le date negli header HTTP DEVONO essere conformi:

- o al formato UNIX Timestamp;
- o alla sintassi HTTP-date definito in **RFC 7231**[122], eg. «Sun, 06 Nov 1994 08:49:37 GMT».

#### 5.4.2.4 [RAC_GEN_FORMAT_004] Tempi di durata e intervalli

I tempi di durata e gli intervalli DEVONO essere espresse in secondi o usare lo standard ISO 8601.

Di seguito alcuni esempi di durata in formato ISO 8601.

I tempi di durata sono prefissati da «P»; giorni e ore sono separati da «T».

Esempi:

- P1Y2M3D - 1 anno, 2 mesi e 3 giorni
- PT1H4M5S - 1 ora, 4 minuti e 5 secondi
- P1M - 1 mese
- PT1M - 1 minuto
- P1Y2M10DT2H30M - 1 anno, 2 mesi, 10 giorni 2 ore e 30 minuti

#### 5.4.2.5 [RAC_GEN_FORMAT_005] Lingue e monete

Si DEVONO utilizzare per le codifiche standard indicate nelle Linee Guida per la Valorizzazione del Patrimonio Informativo Nazionale[1], inclusi:

- ISO 3166-1-alpha2 country (due lettere)
- ISO 639-1 language code
- ISO 639-1 per le varianti dei linguaggi.
- ISO 4217 alpha-3 currency codes

Per le valute è possibile basarsi sullo schema Money - ripreso dal lavoro di standardizzazione del Berlin Group sotto l'egida dello European Standards e contenente i campi:

- amount (string)
- currency [ISO-4217]

Esempio:

[120] https://tools.ietf.org/html/rfc3339.html
[121] https://tools.ietf.org/html/rfc3339.html
[122] https://tools.ietf.org/html/rfc7231.html
[1] Cfr. https://docs.italia.it/italia/daf/lg-patrimonio-pubblico/it/bozza/index.html

```
{
        "tax_id": "imu-e472",
        "value": {
                "amount": "100.23",
                "currency": "EUR"
        }
}
```

```
<payment>
        <taxId>imu-e472</taxId>
        <value>
                <currency>EUR</currency>
                <amount>100.23</amount>
        </value>
</payment>
```

### 5.4.3 Raccomandazioni sulla progettazione e naming

#### 5.4.3.1 [RAC_GEN_NAME_001] Utilizzare i nomi delle proprietà secondo nomenclature standard

Le proprietà DEVONO utilizzare, ove possibile, la nomenclatura indicata nelle Linee Guida per la valorizzazione del Patrimonio[1] informativo pubblico e le relative ontologie[2].

#### 5.4.3.2 [RAC_GEN_NAME_002] Nomenclatura delle proprietà

Le proprietà DOVREBBERO avere una nomenclatura consistente.

Scegliere uno dei due stili di seguito e modificarlo in ASCII:

- snake_case
- camelCase

Non usare contemporaneamente snake_case e camelCase nella stessa API.

Ad esempio:

SI

```
{
  "givenName": "Mario",
  "familyName": "Rossi"
}
```

SI

```
{
  "given_name": "Mario",
  "family_name": "Rossi"
}
```

NO

```
{
  "given_name": "Mario",
  "familyName": "Rossi"
}
```

[1] Cfr. https://docs.italia.it/italia/daf/lg-patrimonio-pubblico/it/bozza/index.html
[2] Cfr. https://github.com/italia/daf-ontologie-vocabolari-controllati

SI

```
<givenName>Mario</givenName>
<familyName>Rossi</familyName>
```

SI

```
<given_name>Mario</given_name>
<family_name>Rossi</family_name>
```

NO

```
<given_name>Mario</given_name>
<familyName>Rossi</familyName>
```

#### 5.4.3.3 [RAC_GEN_NAME_003] Descrittività dei nomi utilizzati

I nomi utilizzati per servizi, path, operation o schemi DEVONO essere auto-descrittivi e fornire quanta più informazione utile riguardo al comportamento implementato, evitando però le ridondanze.

Si deve inoltre evitare l'utilizzo di acronimi quando questi non siano universalmente riconosciuti anche al di fuori del dominio applicativo.

Esempio in un'architettura orientata alle risorse:

> In un servizio per la gestione delle istanze dei cittadini, il nome dell'attributo
>
> `gestioneIstanzeCittadinoAbilitatoBoolean`
>
> può essere semplificato in
>
> `cittadinoAbilitato`
>
> se il servizio è limitato alla gestione delle istanze e l'output del campo è desumibile dal contesto.

### 5.4.4 Raccomandazioni sul logging

#### 5.4.4.1 [RAC_GEN_LOG_01] Informazioni di Logging

I log file DEVONO contenere:

- l'istante della comunicazione in formato UTC (**RFC 3339**[123]) e con i separatori Z e T in maiuscolo. La specifica è fondamentale per l'interoperabilità dei sistemi di logging e auditing, evitando problemi di transizione all'ora legale e la complessità nella gestione di fusi orari nell'ottica dell'interoperabilità con altre PA europee;
- l'URI che identifica l'erogatore e l'operazione richiesta;
- la tipologia di chiamata (ad esempio, metodo HTTP per i protocolli basati su HTTP);
- esito della chiamata (ad es., stato della response HTTP se disponibile, SOAP fault nel caso di web service SOAP);
- ove applicabile, l'Indirizzo IP del client;
- ove applicabile, identificativo del consumatore o altro soggetto operante la richiesta comunicato dal fruitore. È cura del fruitore procedere alla codifica e all'anonimizzazione, ove necessario;
- ove applicabile, un identificativo univoco della richiesta, utile a eventuali correlazioni.

[123] https://tools.ietf.org/html/rfc3339.html

### 5.4.5 Raccomandazioni sulla robustezza

Ai fini di garantire la responsività di una API è necessario impedire a singoli fruitori di esaurire la capacità di calcolo e di banda dell'erogatore. La tecnica comunemente utilizzata in questi casi è il rate limiting (anche noto come throttling). Il rate limit fornisce ad uno specifico fruitore un numero massimo di richieste soddisfacibili all'interno di uno specifico arco temporale (es. 1000 richieste al minuto). Un numero di richieste che superi il limite imposto provoca il rifiuto di ulteriori richieste da parte di uno specifico fruitore per un intervallo di tempo predeterminato.

Sulle politiche riguardanti il numero massimo di richieste e la relativa finestra temporale, e quelle riguardanti il tempo di attesa per nuove richieste (che può essere incrementato in caso di richieste reiterate, es. con una politica di aumento esponenziale) si lascia libertà agli implementatori previa un'analisi di carico massimo sopportabile dall'erogatore.

#### 5.4.5.1 [RAC_ROBUSTEZZA_001] Segnalare raggiunti limiti di utilizzo

Gli erogatori di interfacce di servizio REST DEVONO segnalare eventuali limiti raggiunti con **HTTP status 429 Too Many Requests**[124].

Le API restituiscono in ogni risposta i valori globali di throttling tramite i seguenti header[1]:

- X-RateLimit-Limit: limite massimo di richieste per un endpoint;
- X-RateLimit-Remaining: numero di richieste rimanenti fino al prossimo reset;
- X-RateLimit-Reset: numero di secondi che mancano al prossimo reset.

In caso di superamento delle quote, le API DEVONO restituire anche l'header:

- **HTTP header Retry-After** [125] : numero minimo di secondi dopo cui il client è invitato a riprovare[2].

Nel caso di SOAP non esistono regole guida standard per la gestione del rate limit e del throttling. Si POSSONO utilizzare gli stessi header e status code HTTP visti nel caso REST.

I fruitori DEVONO:

- rispettare gli header di throttling;
- rispettare l'header X-RateLimit-Reset quando restituisce il numero di secondi che mancano al prossimo reset, ed eventualmente gestire l'indicazione in timestamp unix;
- rispettare l'header **HTTP header Retry-After** [126] sia nella variante che espone il numero di secondi dopo cui riprovare, sia nella variante che espone la data in cui riprovare.

#### 5.4.5.2 [RAC_ROBUSTEZZA_002] Segnalare il sovraccarico del sistema o l'indisponibilità del servizio

Gli erogatori DEVONO definire ed esporre un piano di continuità operativa segnalando il sovraccarico del sistema o l'indisponibilità del servizio con **HTTP status 503 Service Unavailable**[127] Service Unavailable.

In caso di sovraccarico o indisponibilità, l'erogatore DEVE ritornare anche:

- **HTTP header Retry-After** [128] con il numero minimo di secondi dopo cui il client è invitato a riprovare.

I fruitori DEVONO:

[124] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/429

[1] È in corso il processo di standardizzazione dell'utilizzo degli header indicati https://datatracker.ietf.org/doc/draft-ietf-httpapi-ratelimit-headers/

[125] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Retry-After

[2] Cfr. **RFC 7231** prevede che l'header **HTTP header Retry-After** possa essere utilizzato sia in forma di data che di secondi

[126] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Retry-After

[127] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/503

[128] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Retry-After

- rispettare **HTTP header Retry-After** [129] sia nella variante che espone il numero di secondi dopo cui riprovare, sia nella variante che espone la data in cui riprovare.

Per REST si DEVE prevedere nella descrizione delle API l'indicazione degli header relativi al rate limiting. L'utilizzo degli header HTTP in SOAP è fuori dagli obiettivi di WSDL come Interface Definition Language.

Esempio di specifica API REST per applicazione del throttling e segnalazione del sovraccarico o dell'indisponibilità:

```
openapi: 3.0.1
info:
   title: RESTrobustezza
   license:
      name: Apache 2.0 License
      url: http://www.apache.org/licenses/LICENSE-2.0.html
   version: "1.0"
paths:
   /resources/{id_resource}/M:
      post:
         description: M
         operationId: PushMessage_1
         parameters:
         - name: id_resource
           in: path
           required: true
           schema:
            type: integer
            format: int32
         requestBody:
            content:
               application/json:
                 schema:
                    $ref: '#/components/schemas/MType'
         responses:
            200:
               description: Esecuzione di M avvenuta con successo
               headers: &ratelimit_headers
                  X-RateLimit-Limit:
                     $ref: '#/components/headers/X-RateLimit-Limit'
                  X-RateLimit-Remaining:
                     $ref: '#/components/headers/X-RateLimit-Remaining'
                  X-RateLimit-Reset:
                     $ref: '#/components/headers/X-RateLimit-Reset'
               content:
                  application/json:
                     schema:
                        $ref: '#/components/schemas/MResponseType'
            404:
               description: Identificativo non trovato
               headers:
                  <<: *ratelimit_headers
               content:
                  application/json:
                     schema:
                        $ref: '#/components/schemas/ErrorMessage'
            400:
               description: Richiesta malformata
               headers:
                  <<: *ratelimit_headers
               content:
                  application/json:
```

(continues on next page)

[129] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Retry-After

(continua dalla pagina precedente)

```
                     schema:
                        $ref: '#/components/schemas/ErrorMessage'
            429:
               description: Limite di richieste raggiunto
               headers:
                  Retry-After:
                     description: Limite massimo richieste
                     schema:
                        type: string
               content:
                  application/json:
                     schema:
                        $ref: '#/components/schemas/ErrorMessage'
            500:
               description: Errore interno avvenuto
               content:
                  application/json:
                     schema:
                        $ref: '#/components/schemas/ErrorMessage'
components:
   headers:
      Retry-After:
         description: |-
            Retry contacting the endpoint \*at least\* after seconds.
            See https://tools.ietf.org/html/rfc7231#section-7.1.3
         schema:
            format: int32
            type: integer
      X-RateLimit-Limit:
         description: The number of allowed requests in the current period
         schema:
            format: int32
            type: integer
      X-RateLimit-Remaining:
         description: The number of remaining requests in the current period
         schema:
            format: int32
            type: integer
      X-RateLimit-Reset:
         description: The number of seconds left in the current period
         schema:
            format: int32
            type: integer
   schemas:
      MType:
         type: object
         properties:
            a:
               $ref: '#/components/schemas/AComplexType'
            b:
               type: string
      MResponseType:
         type: object
         properties:
            c:
               type: string
      AComplexType:
         type: object
         properties:
            a1s:
               type: array
```

(continues on next page)

(continua dalla pagina precedente)

```
            items:
               type: integer
               format: int32
            a2:
               type: string
      ErrorMessage:
         type: object
         properties:
            detail:
               description: |
                  A human readable explanation specific to this occurrence of the
                  problem.
               type: string
            instance:
               description: |
                 An absolute URI that identifies the specific occurrence of the
                 problem.
                 It may or may not yield further information if dereferenced.
               format: uri
               type: string
            status:
               description: |
                  The HTTP status code generated by the origin server for this
                  occurrence
                  of the problem.
               exclusiveMaximum: true
               format: int32
               maximum: 600
               minimum: 100
               type: integer
            title:
               description: |
                  A short, summary of the problem type. Written in english and
                  readable
                  for engineers (usually not suited for non technical stakeholders
                  and not localized);
               example: Service Unavailable
               type: string
            type:
               default: about:blank
               description: |
                 An absolute URI that identifies the problem type. When
                 dereferenced,
                 it SHOULD provide human-readable documentation for the problem
                 type
                 (e.g., using HTML).
               format: uri
               type: string
```

Di seguito, un esempio di chiamata alle API, con risposta nel caso in cui i limiti non siano ancora stati raggiunti e nel caso in cui invece il fruitore debba attendere per presentare nuove richieste.

Endpoint

https://api.ente.example/rest/nome-api/v1/resources/1234/M

1. Request

```
POST /rest/nome-api/v1/resources/1234/M HTTP/1.1
Host: api.ente.example
Content-Type: application/json

```

(continues on next page)

(continua dalla pagina precedente)

```
{
"a": {
        "a1": [1, "...", 2],
        "a2": "RGFuJ3MgVG9vbHMgYXJlIGNvb2wh"
        },
"b": "Stringa di esempio"
}
```

2. Response 200 con rate limiting

```
HTTP/1.1 200 OK
X-RateLimit-Limit: 30
X-RateLimit-Remaining: 11
X-RateLimit-Reset: 44

{"c" :"risultato"}
```

2. Response 429 Too Many Requests

```
HTTP/1.1 429 Too Many Requests
Content-Type: application/problem+json
Retry-After: 60

{
"status": 429,
"title": "Hai superato la quota di richieste."
}
```

2. Response 503 Service Unavailable

```
HTTP/1.1 503 Service Unavailable
Content-Type: application/problem+json
Retry-After: 3600

{
"status": 503,
"title": "Servizio in manutenzione."
}
```

#### 5.4.5.3 [RAC_ROBUSTEZZA_003] Uniformità di Indicatori ed Obiettivi di Servizio

Gli SLI pubblicati DEVONO:

- utilizzare unità di misure referenziate dal Sistema Internazionale (ad esempio, secondi o bytes);
- indicare nel nome identificativo l'eventuale periodo di aggregazione con i soli suffissi s (secondi), m (minuti), d (giorni) e y (anni), utilizzando al posto dei mesi il numero di giorni;
- includere la latenza aggiuntiva dovuta ad eventuali componenti infrastrutturali e di rete (ad esempio, proxy o gateway).

Gli SLO e gli SLA DOVREBBERO essere in relazione diretta con i valori associati (ad esempio, indicare il success rate anziché l'error rate), in modo che a valori più alti corrispondano risultati positivi.

Alcuni esempi di indicatori a cui è possibile associare degli obiettivi o degli accordi:

- dimensione massima di ogni richiesta accettata. Le richieste più grandi possono essere rifiutate
- latenza al 90º percentile. Utilizzata per calcolare la responsività
- percentuale di minuti negli ultimi 30 giorni in cui l'interfaccia di servizio è stata disponibile

- valori a 30 giorni del success rate, ovvero il numero di chiamate terminate con successo rispetto al numero totale di chiamate
- Application Performance inDEX[3], indice su scala percentuale di qualità del servizio misurato a 30 giorni
- tempo di risposta medio delle richieste totali (includendo le richieste rifiutate a causa del throttling) negli ultimi 30 giorni
- throughput misurato in byte/s

## 5.5 Raccomandazioni tecniche per REST

In questo capitolo si raccolgono delle indicazioni relative alla tecnologia REST, al fine di favorire l'interoperabilità.

### 5.5.1 Raccomandazioni sul formato dei dati

#### 5.5.1.1 [RAC_REST_FORMAT_001] Utilizzo oggetti JSON

Nella tecnologia REST la comunicazione DOVREBBE avvenire tramite oggetti JSON **RFC 8259**[132] con il relativo media-type application/json.

È possibile fare eccezione in presenza di specifiche in cui gli oggetti di comunicazione sono formalizzati in forma diversa da JSON (es. INSPIRE, HL7).

#### 5.5.1.2 [RAC_REST_FORMAT_002] Codificare dati strutturati con oggetti JSON

I dati strutturati in formato JSON **RFC 8259**[133] DOVREBBERO essere trasferiti tramite oggetti, in modo da permettere l'estensione retrocompatibile della response con ulteriori attributi, ad esempio paginazione.

Cioè:

- il payload di una response contenente una entry ritorna un oggetto

```
{
        "given_name": "Paolo",
        "last_name": "Rossi",
        "id": 313
}
```

- il payload di una response contenente più entry ritorna un oggetto contenente una lista e non direttamente una lista.

```
{
        "items": [
                {
                        "given_name": "Carlo",
                        "family_name": "Bianchi",
                        "id": 314
                },
                {
                        "given_name": "Giuseppe",
                        "family_name": "Verdi",
                        "id": 315
                }
        ]
}
```

[3] Cf. https://en.wikipedia.org/wiki/Apdex
[132] https://tools.ietf.org/html/rfc8259.html
[133] https://tools.ietf.org/html/rfc8259.html

#### 5.5.1.3 [RAC_REST_FORMAT_003] Convenzioni di rappresentazione

DEVONO usarsi le seguenti convenzioni di rappresentazione:

- I booleani non DEVONO essere null.
- Gli array vuoti non DEVONO essere null, ma liste vuote, ad es. [].
- Le enumeration DEVONO essere rappresentate da stringhe non nulle.

#### 5.5.1.4 [RAC_REST_FORMAT_004] Definire format quando si usano i tipi Number ed Integer

I numeri e gli interi DEVONO indicare la dimensione utilizzando il parametro format.

La seguente tabella - non esaustiva - elenca un set minimo di formati.

Tabella 5.3: Set minimo dei formati

| TYPE | FORMAT | VALORI AMMESSI |
|---|---|---|
| integer | int32 | interi tra -2^31 e 2^31-1 |
| integer | int64 | interi tra -2^63 e 2^63-1 |
| number | decimal32 / float | IEEE 754-2008/IS 60559:2011 decimale a 32 bit |
| number | decimal64 / double | IEEE 754-2008/IS 60559:2011 decimale a 64 bit |
| number | decimal128 | IEEE 754-2008/IS 60559:2011 decimale a 128 bit |

Le implementazioni DEVONO utilizzare il tipo più adatto.

#### 5.5.1.5 [RAC_REST_FORMAT_005] Usare link relations registrate

DEVONO usarsi le specifiche indicate in IANA registered link relations[1] per rappresentare link e riferimenti a risorse HTTP esterne.

### 5.5.2 Raccomandazioni su progettazione e naming

In assenza di specifiche regole per l'API Naming (es. HL7, INSPIRE, ..) DOVREBBERO essere adottate le seguenti regole.

#### 5.5.2.1 [RAC_REST_NAME_001] Uso corretto dei metodi HTTP

I metodi HTTP DEVONO essere utilizzati rispettando la semantica indicata in **RFC 7231#section-4.3**[134].

#### 5.5.2.2 [RAC_REST_NAME_002] Usare parole separate da trattino «-» per i path (kebab-case)

Nella definizione dei path si DEVE utilizzare il separatore «-» (kebab-case).

Esempio: `/tax-code/{tax_code_id}`

Il path DOVREBBE essere semplice, intuitivo e coerente.

[1] Cfr. https://www.iana.org/assignments/link-relations/link-relations.xhtml
[134] https://tools.ietf.org/html/rfc7231.html#section-4.3

#### 5.5.2.3 [RAC_REST_NAME_003] Preferire Hyphenated-Pascal-Case per gli header HTTP

DOVREBBE preferirsi Hyphenated-Pascal-Case per gli header HTTP.

Esempio:

```
Accept-Encoding
Apply-To-Redirect-Ref
Disposition-Notification-Options
Message-ID
```

#### 5.5.2.4 [RAC_REST_NAME_004] Le collezioni di risorse possono usare nomi al plurale

Si consiglia di differenziare il nome delle collezioni e delle risorse. Questo permette di separare a livello di URI, endpoint che sono in larga parte funzionalmente differenti.

Esempio 1: ricerca di documenti per data in una collezione

Request:

```
GET /documenti?data=2018-05-01 HTTP/1.1
Host: api.example
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "items": [ ".." ],
  "limit": 10,
  "next_cursor": 21314123
}
```

Esempio 2: recupera un singolo documento

Request:

```
GET /documento/21314123 HTTP/1.1
Host: api.example
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": 21314123,
  "title": "Atto di nascita ..."
}
```

#### 5.5.2.5 [RAC_REST_NAME_005] Utilizzare Query String standardizzate

Esempio 1: La paginazione DEVE essere implementata tramite i parametri:

```
cursor, limit, offset, sort
```

Esempio 2: La ricerca, il filtering e l'embedding dei parametri DEVE essere implementata tramite i parametri:

```
q, fields, embed
```

#### 5.5.2.6 [RAC_REST_NAME_006] Non passare tramite l'header Link informazioni fornite nella response JSON

Eventuali link a risorse utili al flusso applicativo DEVONO essere restituiti nel payload JSON e non nell' **HTTP header Link** [135] definito in **RFC 8288**[136]. Questo semplifica l'implementazione dei client. È comunque possibile usare l'"**HTTP header Link** [137] per passare informazioni di tipo diverso.

Nell'esempio seguente i riferimenti alla paginazione sono riportati nel payload.

Listato 5.2: Link applicativo nel payload JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "items": [ ".." ],
   "_links": {
     "next": "https://api.example/rest/library/v1/books?cursor=432123"
   }
}
```

E" corretto ad esempio estendere la risposta precedente referenziando nell'**HTTP header Link** [138] l'OAS3 del servizio usando il link relation `service-desc` definito in **RFC 8631**[139]. In questo caso il link ad openapi.yaml non serve al flusso applicativo e non è utile comunicarlo nel payload JSON.

Listato 5.3: Link applicativo nel payload JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://api.example/rest/users/v1/openapi.yaml>; rel="service-desc"; title=
↪"Books API specifications"

{
   "items": [ ".." ],
   "_links": {
     "next": "https://api.example/rest/library/v1/books?cursor=432123"
   }
}
```

#### 5.5.2.7 [RAC_REST_NAME_007] Usare URI assoluti nei risultati

Le response DOVREBBERO restituire URI assoluti, al fine di indicare chiaramente al client l'indirizzo delle risorse di destinazione e non obbligare i client a fare «inferenza» dal contesto.

#### 5.5.2.8 [RAC_REST_NAME_008] Usare lo schema Problem JSON per le risposte di errore

In caso di errori si DEVONO ritornare:

- un payload di tipo Problem definito in **RFC 7807**[140]
- il media type application/problem+json

[135] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Link
[136] https://tools.ietf.org/html/rfc8288.html
[137] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Link
[138] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Link
[139] https://tools.ietf.org/html/rfc8631.html
[140] https://tools.ietf.org/html/rfc7807.html

- uno status code esplicativo
- l'oggetto, possibilmente esteso

Quando si restituisce un errore è importante non esporre dati interni delle applicazioni. Per prevenire il rischio di user-enumeration, i messaggi di errore di autenticazione non devono fornire informazioni sull'esistenza o meno dell'utenza.

Dopo aver validato il contenuto delle richieste si DEVE ritornare:

- **HTTP status 415 Unsupported Media Type**[141] se il valore in **HTTP header Content-Type** [142] non è supportato;
- **HTTP status 400 Bad Request**[143] o **HTTP status 404 Not Found**[144] se si ipotizza che la richiesta sia malevola;
- **HTTP status 422 Unprocessable Entity**[145] se la representation contenuta nella richiesta è sintatticamente corretta ma semanticamente non processabile.

#### 5.5.2.9 [RAC_REST_NAME_009] Ottimizzare l'uso della banda e migliorare la responsività

Si DOVREBBERO utilizzare:

- tecniche di compressione;
- paginazione;
- un filtro sugli attributi necessari;
- le specifiche di optimistic locking (**HTTP header ETag** [146], if-(none-)match) **RFC 7232**[147].

È possibile ridurre l'uso della banda e velocizzare le richieste filtrando i campi delle risorse restituite.

Esempio 1: Non filtrato

Request:

```
GET /resources/123 HTTP/1.1
Host: api.example
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{

"id": "cddd5e44-dae0-11e5-8c01-63ed66ab2da5",
"name": "Mario Rossi",
"address": "via del Corso, Roma, Lazio, Italia",
"birthday": "1984-09-13",
"partner": {
        "id": "1fb43648-dae1-11e5-aa01-1fbc3abb1cd0",
        "name": "Maria Rossi",
        "address": "via del Corso, Roma, Lazio, Italia",
        "birthday": "1988-04-07"
        }
}
```

[141] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/415
[142] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type
[143] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400
[144] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404
[145] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/422
[146] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag
[147] https://tools.ietf.org/html/rfc7232.html

Esempio 2: Filtrato

Request:

```
GET /resources/123?fields=(name,partner(name)) HTTP/1.1
Host: api.example
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
"name": "Mario Rossi",
"partner": {
        "name": "Maria Rossi"
        }
}
```

Si DOVREBBE effettuare la Resource Expansion per ritornare risorse correlate tra loro, in modo da ridurre il numero di richieste.

In tal caso va usato:

- il parametro embed utilizzando lo stesso formato dei campi per il filtering;
- l'attributo _embedded contenente le entry espanse.

Esempio 3: Resource Expansion, utile a ritornare i dati di una persona associati ad un codice fiscale.

Request:

```
GET /tax_code/MRORSS12T05E472W?embed=(person) HTTP/1.1
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "tax_code":"MRORSS12T05E472W",
  "_embedded": {
          "person":{
                "given_name":"Mario",
                "family_name":"Rossi",
                "id":"1234-ABCD-7890"
     }
  }
}
```

#### 5.5.2.10 [RAC_REST_NAME_010] Il caching http deve essere disabilitato

Il caching DOVREBBE essere disabilitato tramite **HTTP header Cache-Control** [148] per evitare che delle richieste vengano inopportunamente messe in cache. Il mancato rispetto di questa raccomandazione può portare all'esposizione accidentale di dati personali.

Le API che supportano il caching DEVONO documentare le varie limitazioni e modalità di utilizzo tramite gli header definiti in **RFC 7234**[149]:

- **HTTP header Cache-Control** [150]

[148] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control
[149] https://tools.ietf.org/html/rfc7234.html
[150] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control

- **HTTP header Vary** [151].

Eventuali conflitti nella creazione di risorse DEVONO essere gestiti tramite gli header:

- **HTTP header ETag** [152];
- **HTTP header If-Match** [153];
- **HTTP header If-None-Match** [154];

che contengono un identificatore opaco che il server è in grado di associare univocamente alla versione della risorsa erogata. Si veda **RFC 7232#section-2.3**[155] per ulteriori informazioni su come implementare questi header.

#### 5.5.2.11 [RAC_REST_NAME_011] Esporre lo stato del servizio

L'API DEVE esporre lo stato del servizio al path '/status' e ritornare un oggetto con media-type application/problem+json (**RFC 7807**[156]). Se il servizio funziona correttamente, l'API ritorna **HTTP status 200 OK**[157].

Segue un esempio di specifica del path in formato OpenAPI 3.

ESEMPIO: Esposizione stato del servizio

```
openapi: 3.0.2

...

paths:

...

/status:

get:

summary: Ritorna lo stato dell'applicazione.

tags:

- public

description: \|

Ritorna lo stato dell'applicazione in formato problem+json

responses:

'200':

content:

application/problem+json:

schema:

$ref: '#/components/schemas/Problem'
```

(continues on next page)

[151] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Vary
[152] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag
[153] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-Match
[154] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None-Match
[155] https://tools.ietf.org/html/rfc7232.html#section-2.3
[156] https://tools.ietf.org/html/rfc7807.html
[157] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200

(continua dalla pagina precedente)

```
  description: \|

  Il servizio funziona correttamente.

  default:

  content:

  application/problem+json:

  schema:

  $ref: '#/components/schemas/Problem'

  description: \|

  Il server ha riscontrato un problema.
```

## 5.6 Raccomandazioni tecniche per SOAP

In questo capitolo si raccolgono delle indicazioni per la tecnologia SOAP, al fine di favorire l'interoperabilità.

### 5.6.1 Raccomandazioni globali

#### 5.6.1.1 [RAC_SOAP_001] Le API SOAP DEVONO rispettare il WS-I Basic Profile versione 2.0

Questo profilo è definito dal WS-I (Web Services Interoperability Organization), ora confluito in OASIS. Questa specifica è implementata dai framework più diffusi.

#### 5.6.1.2 [RAC_SOAP_002] Utilizzo di camelCase e PascalCase

Per i nomi dei servizi si DOVREBBE utilizzare PascalCase.

Per le operazioni implementate e gli argomenti si DOVREBBE utilizzare il camelCase.

#### 5.6.1.3 [RAC_SOAP_003] Unicità dei namespace e utilizzo di pattern fissi

All'interno del WSDL DOVREBBE essere presente un namespace unico.

#### 5.6.1.4 [RAC_SOAP_004] Esporre lo stato del servizio

L'API DEVE includere un metodo 'echo' per restituire lo stato della stessa.

ESEMPIO: Esposizione stato del servizio

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://
↪www.w3.org/2001/XMLSchema" xmlns:ws="http://www.example.com/webservice"␣
↪targetNamespace="http://www.example.com/webservice">
  <wsdl:types>
    <xs:schema>
      <xs:complexType name="userDefinedFault">
        <xs:sequence>
          <xs:element name="errorCode" type="xs:int"/>
```

(continues on next page)

(continua dalla pagina precedente)

```
          <xs:element name="detail" type="xs:string"/>
          <xs:element name="message" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="requestEcho">
    <wsdl:part name="msg" type="xs:string"/>
  </wsdl:message>
  <wsdl:message name="responseEcho">
    <wsdl:part name="msg" type="xs:string"/>
  </wsdl:message>
  <wsdl:message name="UserDefinedException">
    <wsdl:part name="fault" type="userDefinedFault"/>
  </wsdl:message>
  <wsdl:portType name="portType">
    <wsdl:operation name="echo">
      <wsdl:input message="ws:requestEcho"/>
      <wsdl:output message="ws:responseEcho"/>
      <wsdl:fault name="fault" message="ws:UserDefinedException"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

## 5.7 Gestione degli allegati

Tra i parametri o i valori di ritorno di una interfaccia di servizio può esserci la presenza di allegati, dove per allegato si intende un contenuto binario, o la cui struttura comunque non è definita direttamente dall'interfaccia di servizio (e.g., un file XML all'interno di un messaggio di SOAP in cui lo schema che definisce il messaggio SOAP non specifica anche la struttura dell'XML allegato). In generale, un allegato può essere passato o ricevuto nelle seguenti forme:

- Codificato in modo da essere rappresentabile con un set predefinito di caratteri. Il caso più comune è quello della codifica Base64.
- Come URL ad una risorsa esterna o in ogni caso come endpoint di accesso ad una risorsa.
- Nel suo formato binario originale.

Nei primi due casi, l'allegato fa parte del contenuto XML o JSON del messaggio, mentre nell'ultimo caso si ricorre a risposte di tipo multipart.

Ognuna di queste modalità presenta vantaggi e svantaggi. L'utilizzo di codifiche come Base64 oppure di URL ha il vantaggio di potere inserire un allegato all'interno del contenuto principale del messaggio. Si noti come anche nel caso di XML l'utilizzo di formati binari all'interno dei campi CDATA sia sconsigliato poiché esiste il rischio che il contenuto binario possa talvolta chiudere il campo CDATA stesso. D'altro canto, l'utilizzo di codifiche comporta un incremento nella banda richiesta poiché l'occupazione dei dati non è, in generale, ottimale. L'utilizzo di URL comporta invece un potenziale rischio potenziale poiché la risorsa collegata può essere successivamente modificata o rimossa.

In ambito SOAP, la prima proposta di standard per l'invio di allegati binari è rappresentato da SWA - SOAP with Attachments, a cui il W3C ha però preferito come standard MTOM - Message Transmission Optimization Mechanism, che ha il vantaggio di estendere agli allegati i meccanismi di sicurezza quali WS-Encryption e WS-Signature. MTOM è solitamente utilizzato insieme a XOP - XML-binary Optimized Packaging quale meccanismo per fare riferimento agli allegati all'interno del messaggio Multipart/Related. L'utilizzo di MTOM con XOP è supportato da tutti i maggiori framework per lo sviluppo di interfacce di servizio SOAP, ma meccanismi similari, sempre basati su XOP, sono supportati anche dai maggiori framework per lo sviluppo di interfacce di servizio REST.

L'utilizzo di un approccio o di un altro dipende fortemente dallo scenario applicativo. Possono essere utilizzate seguenti regole:

- L'invio di allegati binari corrispondenti a file fa preferire solitamente l'invio di dati in formato binario, quindi mediante MTOM/XOP.
- L'utilizzo di Base64 è consigliato per l'invio di allegati di dimensioni ridotte quali ad esempio firme digitali o codici di controllo.
- L'utilizzo di URL può essere considerato nel caso in cui gli allegati siano di dimensioni tali da rendere il trasferimento via rete oneroso, a patto che si possa assicurare la persistenza della risorsa (in termini temporali) e che questa non venga modificata (a tal fine è possibile utilizzare tecniche ad esempio di hashing). Quest'ultimo caso richiede quindi solitamente trust tra fruitore ed erogatore.

# Indice

## H



## R