

## - Specifiche di Interfaccia Servizi degli Aderenti -

# API Manager

Codice Documento: **CRS-ISAU-SIAU#115**

Revisione del Documento: **06**

Data revisione: **16/11/2017**

## Cronologia delle Revisioni

Revisione	Data	Sintesi delle Modifiche
01	29-06-2016	Prima versione
02	23-11-2016	Precisazioni nelle informazioni di contesto per il tenant "operatori.siss"
03	06-07-2017	Introdotta nel JWT l'informazione Reale/Virtuale per il tenant cittadini.rl
04	22-09-2017	Introdotta la possibilità d'ottenere le risposte d'errore nell'invocazione delle API in formato JSON anziché XML
05	09-11-2017	Cambiamento di denominazione da CRS-ISAU-SIAI#115 (ad uso interno) a CRS-ISAU-SIAU#115 (pubblico) per consentire/supportare l'integrazione di applicazioni realizzate da enti esterni. Specificata la necessità di verificare l'attendibilità del certificato SSL/TLS dell'API Manager prima di effettuare le chiamate alle API
06	16-11-2017	Specificata la necessità di verificare l'attendibilità del certificato SSL/TLS dell'API Manager anche nel capitolo riguardante i metodi per la gestione dei token d'autorizzazione

## Limiti di utilizzo del documento

In base alla classificazione del documento.

# INDICE

<b>1</b>	<b>INTRODUZIONE.....</b>	<b>5</b>
1.1	SCOPO E CAMPO DI APPLICAZIONE .....	5
1.2	RIFERIMENTI.....	5
1.3	ACRONIMI E DEFINIZIONI .....	5
<b>2</b>	<b>GENERALITA' .....</b>	<b>6</b>
2.1	BREVE OVERVIEW DEL PROTOCOLLO OAUTH2.0 .....	6
<b>3</b>	<b>SCENARI SUPPORTATI E COMPARTIMENTAZIONE DELLE API .....</b>	<b>8</b>
3.1	SCENARIO OPERATORI.SISS .....	8
3.2	SCENARIO CITTADINI.RL .....	10
3.3	SCENARIO SERVIZI.RL .....	10
3.4	GESTIONE DEGLI "SCOPE" .....	11
3.5	UTILIZZO DI MOLTEPLICI DISPOSITIVI DA PARTE DELLO STESSO UTENTE .....	11
3.6	MODALITÀ D'UTILIZZO DELLE API.....	11
3.7	MODALITÀ D'INTEGRAZIONE NELLE APP AL CITTADINO.....	12
<b>4</b>	<b>INTERFACCE ESPOSTE .....</b>	<b>13</b>
4.1	AUTENTICAZIONE DEL SERVER .....	13
4.2	AUTORIZZAZIONE DELL'APPLICAZIONE DA PARTE DELL'UTENTE .....	13
4.2.1	Authorize .....	13
4.2.1.1	Invocazione.....	14
4.2.1.2	Risposta .....	14
4.2.1.3	Esempio d'invocazione.....	15
4.2.2	Utilizzo dell'Authorization Code per ottenere i token .....	16
4.2.2.1	Invocazione.....	16
4.2.2.2	Risposta .....	16
4.2.2.3	Esempio d'invocazione.....	17
4.3	AUTORIZZAZIONE DELLA SOLA APPLICAZIONE .....	18
4.3.1	Rilascio Access Token .....	18
4.3.1.1	Invocazione.....	18
4.3.1.2	Risposta .....	19
4.3.1.3	Esempio d'invocazione.....	19
4.4	FRUIZIONE DELLE API.....	21
4.4.1	API Call .....	22
4.4.1.1	Autenticazione del server.....	22
4.4.1.2	Invocazione.....	22
4.4.1.3	Risposta .....	23
4.4.1.4	Esempio d'invocazione.....	23
4.4.2	Sottomissione authorization PIN (Codice di Sicurezza) .....	25
4.4.2.1	Invocazione.....	25
4.4.2.2	Risposta .....	26
4.4.2.3	Esempio d'invocazione.....	26
4.4.3	Refresh token .....	28
4.4.3.1	Invocazione.....	28
4.4.3.2	Risposta .....	28
4.4.3.3	Esempio d'invocazione.....	29
4.5	REVOCA DEL REFRESH TOKEN .....	30
4.5.1	API di Revoca.....	30
4.5.1.1	Invocazione.....	30
4.5.1.2	Risposta .....	30
4.5.1.3	Esempio d'invocazione.....	31
4.6	CONTESTO ASSOCIATO ALLE API CALL.....	32
4.6.1	Json Web Token (JWT) .....	32
4.6.1.1	Modalità di inoltro del JWT alle API esposte .....	32



4.6.1.2	Esempio di JWT .....	33
4.7	API D'UTILITÀ .....	34
4.7.1	API WhoAmI .....	34
4.7.1.1	Invocazione.....	34
4.7.1.2	Risposta .....	34
4.7.1.3	Esempio d'invocazione.....	34
4.8	REQUISITI DI SICUREZZA AGGIUNTIVI PER LE APP MOBILE.....	36
4.8.1	Riservatezza dei segreti statici “immersi” nell'applicazione .....	36
4.8.2	Riservatezza del Refresh Token.....	36
4.8.3	Riservatezza dell'Access Token e del Grant Token.....	36
4.9	APPENDICE 1 – FAULT GENERATI DALL'API MANAGER E CODICI D'ERRORE .....	38
4.9.1	Fault e codici d'errore.....	40
4.10	APPENDICE 2 – ESEMPIO DI ALBERO DELLE INTERAZIONI DI UN'APP MOBILE.....	42

## 1 INTRODUZIONE

---

### 1.1 *Scopo e campo di applicazione*

Il documento dettaglia le interfacce messe a disposizione alle applicazioni per l'accesso programmatico da dispositivi mobili (smartphones e tablets) a metodi REST esposti da web application di Lombardia Informatica.

---

### 1.2 *Riferimenti*

N/A

---

### 1.3 *Acronimi e definizioni*

IdPC	Identity Provider Cittadino
OTP	One Time Password
PIN	Personal Identification Number
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol

## 2 GENERALITA'

Il presente documento descrive le interfacce per l'accesso programmatico a servizi REST o SOAP da parte di applicazioni disposte al di fuori del dominio di sicurezza di Lombardia Informatica che accedono ai servizi tramite internet o, in casi specifici, l'Extranet del SISS.

La soluzione è basata sull'integrazione con l'infrastruttura di autenticazione di LI di un prodotto di API Management di mercato. Questo documento si focalizza sulla sola fase di fruizione dei servizi mentre quella di adesione come "developer" e di creazione di applicazioni che sottoscrivano le API esposte è oggetto di altri documenti.

Le soluzioni di sicurezza adottate rilevanti dal punto di vista dell'integratore sono le seguenti:

Scopo	Standard / versione
Identificazione e autenticazione del cittadino	Sistema IdPC di LISP A nella modalità TS-CNS o <i>strong authentication</i> con Userid/Password/OTP (in particolare per dispositivi mobili)
Identificazione e autenticazione dell'operatore SISS	Sistemi SSO ed IdPC di LISP A nella modalità autenticazione SISS da PdL o <i>strong authentication</i> con Userid/Password/OTP per gli accessi in mobilità
Protocollo d'autenticazione/autorizzazione API	OAuth2.0 esteso con la gestione di un PIN di sicurezza per l'accesso ad API che richiedono criteri d'accesso più restrittivi
Sicurezza delle comunicazioni	HTTP/S con protocollo TLS utilizzato in modalità <i>server authentication</i>

### 2.1 Breve overview del protocollo OAuth2.0

Si raccomanda vivamente che l'integratore abbia una conoscenza adeguata del protocollo OAuth2.0, comunque per aiutarlo ad orientarsi tra i molteplici scenari d'integrazione previsti da tale standard si riporta nel seguito una breve overview del protocollo.

Nel seguente sequence diagram l'attore APP può essere costituito da un'APP installata su un dispositivo mobile, in questo caso deve avere accesso al browser di sistema o avere una web view embedded al proprio interno, oppure da una web application erogata da un web server collocato al di fuori del dominio di LI.

Il protocollo OAuth2.0 prevede che, veicolato dall'applicazione client, l'utente si autentichi all'espositore di API; l'espositore una volta autenticato l'utente interagisce con lui allo scopo di richiedere l'autorizzazione all'accesso alle sue risorse. Se l'utente ha autorizzato l'accesso, all'applicazione client viene restituito un codice d'autorizzazione "una tantum" che deve essere tempestivamente scambiato con un token d'accesso (AT) tramite l'invocazione di una API esposta dall'API Manager. Unitamente all'Access Token (AT), che ha una validità limitata nel tempo, viene restituito anche un Token di Refresh (RT – Refresh Token) di durata elevata, utilizzabile per ottenere nuovi AT validi quando questi sono scaduti.

Prima di poter operare, l'applicazione deve essere stata registrata nel sistema di API Management da parte di un'utente *developer* esplicitamente autorizzato da LI o da LI stessa per suo conto.

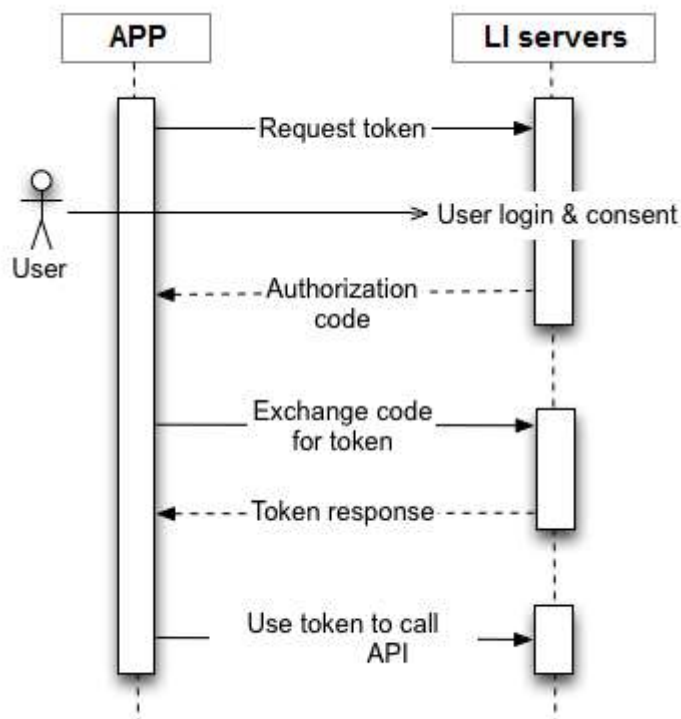
Le informazioni più rilevanti legate all'applicazione sono:

- Nome e descrizione dell'applicazione integrata;
- ClientID, quantità di sicurezza prodotta dal sistema di API Management ed assegnata all'applicazione;
- ClientSecret, quantità di sicurezza prodotta dal sistema di API Management ed assegnata all'applicazione;
- RedirectURI, URL di "rientro" a cui verrà fatto navigare il browser dell'utente al termine della fase d'autenticazione e autorizzazione, registrata nel sistema di API Management.

Si noti che ClientID e ClientSecret possono essere rilasciate (e sono differenti) per un'istanza di test/sviluppo piuttosto che di produzione dell'applicazione. Lato LI per ciascuna delle due istanze è possibile offrire interfacce identiche ma che utilizzano risorse d'elaborazione e dati di test piuttosto che di produzione.

Flusso tipico:

1. L'applicazione reindirizza un browser (una web view o una pagina del browser) alla URI d'autorizzazione dell'API Manager. I parametri nella *query string* della URL indicano il tipo di accesso alle API che richiede tale applicazione;
2. Le pagine web di LI eseguono l'autenticazione dell'utente e la sua volontà di autorizzare l'access; il risultato di tale processo è l'emissione di un codice d'autorizzazione (Authorization Code), tale codice è restituito sotto forma di parametro della *query string* della redirect (preconfigurata) con cui si deve concludere la navigazione. Si noti che l'implementazione di LI, in uno degli scenari supportati, questa fase richiede all'utente la definizione di un PIN d'accesso (Codice di Sicurezza) che sarà successivamente richiesto all'utente per l'accesso alle API più "privilegiate";
3. L'applicazione scambia il codice d'autorizzazione con un token d'accesso (Access Token) e uno di rinnovo (Refresh Token). Durante questo scambio l'applicazione fornisce il proprio Client ID e il proprio Client Secret (ottenuti da LI in sede di registrazione dell'applicazione nell'API Manager);
4. L'applicazione utilizza l'Access Token per eseguire chiamate alle API e memorizza localmente il Refresh Token per usi futuri (i.e. la generazione di un nuovo Access Token quando questo scade); la conservazione del Refresh Token viene normalmente effettuata solo nelle APP per dispositivi mobili al fine di realizzare un *pairing* permanente tra dispositivo e servizi esposti in modo di non necessitare un'autenticazione ed autorizzazione ad ogni attivazione dell'APP.



### 3 Scenari supportati e compartimentazione delle API

Sono supportati molteplici scenari con differenti esigenze d'autenticazione, autorizzazione ed accesso ai servizi. Ad ogni scenario è associato un "tenant" a cui corrispondono le specifiche policy: ogni utenza "developer" e le applicazioni client da lui create sono associate al "tenant" a cui appartiene l'utenza dello sviluppatore.

Gli scenari individuati sono i seguenti:

- operatori.siss – Per accesso alle API dei servizi SISS esposti in Extranet e/o Internet;
- cittadini.rl – Per l'accesso ad API da parte di APP fruite dal cittadino (da dispositivi mobili);
- servizi.rl – Per l'accesso ad API per cui è necessaria la sola autenticazione dell'applicazione chiamante senza quella aggiuntiva dell'utente che la sta utilizzando.

#### 3.1 Scenario operatori.siss

Questo scenario è stato definito per supportare l'accesso ai servizi del SISS da parte di applicazioni web dispiegate al di fuori del dominio di Lombardia Informatica.

Nella modalità "tradizionale" un'applicazione web deve invocare i metodi di SISS della PdL operatore tramite chiamate http in localhost:8000 per tramite del browser web che l'utente sta utilizzando nell'accesso all'applicazione dell'ente: con l'irrigidirsi delle politiche di sicurezza dei browser in termini di Applet Java e Activex, tale integrazione è sempre più complessa e limitante nonché destinata prima o poi ad essere dismessa.

Questo scenario prevede che l'applicazione web aziendale, una volta accettato e riconosciuto il proprio utente, esegua la navigazione alla pagina d'autorizzazione dell'API Manager, qui possono accadere due cose:

- L'utente è già autenticato al SISS e, pertanto, il sistema ne acquisisce l'identità procedendo direttamente alla fase d'autorizzazione all'accesso;
- L'utente non è autenticato al SISS od opera da una postazione non SISS e, quindi, viene rediretto alle pagine d'autenticazione web per l'operatore SISS in mobilità; va da sé che l'utente per autenticarsi in questa modalità deve aver precedentemente predisposto tale modalità d'accesso. Se l'operatore non dispone delle credenziali per l'autenticazione in mobilità deve eseguire l'autenticazione al SISS e, tramite l'applicazione aziendale, ripetere l'accesso alla pagina di autorizzazione dell'API Manager.

*ATTENZIONE: nella modalità d'autenticazione SISS in mobilità potrebbero non essere disponibili tutti i servizi in quanto le regole d'autorizzazione coincidono con quelle dell'accesso in mobilità da Internet.*

La fase d'autorizzazione presenta all'operatore la possibilità di negare l'accesso o di autorizzarlo per la sessione corrente. Al termine del processo d'autorizzazione, comunque, il sistema redirige la navigazione alla RedirectURI associata all'applicazione client.

Si noti che la fase d'autenticazione/autorizzazione richiede che la postazione utente possa accedere ad internet anche se la redirezione finale al web server aziendale fa sì che i token d'accesso siano acquisiti dal web server e, conseguentemente, le chiamate ai servizi SISS a nome dell'utente saranno effettuate direttamente dal web server tramite Extranet SISS e non più tramite il browser della PdL.

Anche se il Refresh Token ha una durata potenzialmente elevata, ad ogni accesso viene verificato che l'utente abbia ancora una sessione di login SISS attiva inoltre, come per le altre modalità d'accesso ai servizi, l'accesso è consentito solo con le autorizzazioni connesse all'ultimo logon al SISS, cioè se l'utente si logga nuovamente l'applicazione deve ripetere la procedura d'autorizzazione dell'API Manager per ottenere i token connessi alla nuova credenziale rilasciata con l'ultimo logon.

All'applicazione integrata è richiesto di non conservare il token di refresh tra diverse sessioni utente ma di eseguire ad ogni nuova sessione la navigazione alla pagina d'autorizzazione il modo di acquisire un'autorizzazione all'accesso





coerente con l'ultima login SISS effettuata dall'operatore, in ogni caso i token diventano inutilizzabili al Logout dell'operatore, allo scadere della sessione di lavoro o per un successivo logon.

---

### 3.2 *Scenario cittadini.rl*

Questo scenario è stato definito per supportare l'accesso ai servizi Lombardia Informatica da parte di APP per dispositivi mobili realizzate da, o sotto il controllo, di Lombardia Informatica o da terze parti che abbiano ottenuto l'autorizzazione da parte di LI / Regione Lombardia.

Lo scenario è tipico di un'applicazione installata, pertanto le interazioni web sono effettuate istanziando una webview aperta sulle URL dell'API Manager. La navigazione si conclude sempre con la redirectione alla RedirectURI configurata per la specifica applicazione che, conseguentemente, ha la possibilità di "intecettare" la navigazione della web view ed estrarre il risultato impostato dall'API Manager.

Rispetto allo scenario precedente si sottolineano due differenze:

- L'opzione "autorizza sempre" non è disponibile in quanto l'obiettivo è quello di creare sempre rapporti "permanenti" tra APP e API;
- Al cittadino è richiesto di definire un PIN (codice di sicurezza) che dovrà essere successivamente digitato per consentire all'APP di invocare delle API che sono state classificate come protette.

L'impostazione e la verifica di un PIN rappresenta un'estensione ad un normale scenario OAuth2 ma si è resa necessaria in quanto l'acquisizione (e la conservazione) del Token di Refresh fornisce l'evidenza di ciò che l'utente possiede ma non di quello che conosce. L'utilizzo di un PIN auto-definito è stato preferito ad una soluzione SMS-OTP in quanto gestibile completamente in-APP, quindi molto più usabile in quanto lo richiede il task-switching che sugli smartphone è sempre macchinoso.

Il PIN deve essere definito dall'utente solo per il primo dei suoi dispositivi in quanto è associato all'utente e non ai suoi device.

La dimostrazione di conoscenza del PIN viene innescata dall'APP quando ha l'esigenza di utilizzare un'API privilegiata e, con un processo simile a quello dell'autorizzazione iniziale, ottiene il rilascio di un Token di Grant derivato dal Token d'Accesso, di cui quindi condivide il periodo di validità.

---

### 3.3 *Scenario servizi.rl*

Questo scenario è stato definito per supportare l'accesso ai servizi del SISS da parte di applicazioni dispiagate all'interno o al di fuori del dominio di Lombardia Informatica per l'accesso a servizi che richiedano unicamente l'autenticazione dell'applicazione e non quella dell'utente finale; tale rapporto è caratteristico nella collaborazione tra servizi di back-end in cui l'accesso alla API non coinvolge esplicitamente l'utente finale.

Nell'ambito di questo scenario è possibile erogare servizi sulla base dell'identità dell'utente finale ma l'identificativo dell'utente deve essere collocato tra i dati applicativi per cui l'API Manager non applica alcun meccanismo d'autorizzazione ulteriore all'autenticazione e autorizzazione dell'applicazione chiamante.

La fase d'autorizzazione utente tramite web browser / web view, quindi, non si applica a questo scenario, in cui le credenziali del servizio chiamante (ClientID e ClientSecret) sono trasmesse come attributi d'autenticazione nell'invocazione della API che rilascia il token d'accesso da allegare alle invocazioni alle API dei servizi esposti.

---

### 3.4 *Gestione degli “scope”*

Una o più API possono essere raggruppate nell’ambito di uno “*scope*” d’autorizzazione. La definizione degli *scope* a cui appartengono le API è effettuata da LI.

E’ responsabilità di LI configurare gli *scope* utilizzabili dalle applicazioni client mentre le applicazioni hanno l’onere di elencare, in sede d’autorizzazione, tutti gli *scope* che gli necessitano per operare. Le API di cui non è stato specificato lo *scope* in sede d’autorizzazione non saranno disponibili a meno di non ripetere nuovamente e correttamente il processo d’autorizzazione.

---

### 3.5 *Utilizzo di molteplici dispositivi da parte dello stesso utente*

Se un’applicazione client è previsto possa essere utilizzata da parte dello stesso utente su più dispositivi, è necessario che il rilascio dei token sia contestualizzato allo specifico dispositivo al fine di evitare interferenze incrociate.

A parità d’applicazione, le quantità ClientID, ClientSecret e RedirectURI dell’applicazione sono le stesse e se l’utente fosse il medesimo non sarebbe possibile rilasciare RT contestualizzati al dispositivo sui cui dovranno persistere.

Per ovviare a questo inconveniente l’applicazione client ha la possibilità di specificare l’identificativo del dispositivo da cui opera (tramite lo speciale *scope device\_XXXX*); l’identificativo può essere determinato dal dispositivo dell’utente o richiesto all’utente stesso ma, in ogni caso, non deve coincidere con quello di altri dispositivi in uso dall’utente per la medesima applicazione client.

L’utilizzo dello *scope device\_* è disponibile anche per la realizzazioni di applicazioni web di terze parti come nel caso dello scenario operatori.siss. In questo caso se ne suggerisce l’uso per impostare un identificativo del server fisico che sta erogando l’applicazione (nel caso di configurazioni ridondate) in modo da ottenere la piena indipendenza dei cicli di vita dei RT rilasciati alle singole istanze dell’applicazione.

---

### 3.6 *Modalità d’utilizzo delle API*

Il sistema di API Management consente la definizione di policy di sicurezza applicabili alle API ed ai sottoscrittori delle stesse.

A livello API possono essere definite limitazioni sui metodi http utilizzabili nell’accesso alle API (ad es. solo POST o GET) o criteri d’autorizzazione legati allo *scope* d’appartenenza di una risorsa piuttosto che la protezione aggiuntiva tramite PIN.

A livello sottoscrittore possono essere imposte ulteriori policy quali il *throttling*, cioè un limite predefinito nel numero di chiamate ad una API che possono essere effettuate nell’unità di tempo. In questo specifico caso l’API Manager prevede condizioni d’errore specifiche per identificare la ragione del rifiuto del servizio.

In ogni caso si faccia riferimento alla documentazione specifica della API per l’identificazione delle condizioni e delle limitazioni d’utilizzo; nel presente documento si illustrano le modalità di fruizione e le condizioni d’errore supportati, indipendentemente dalla loro applicabilità ad una specifica API.

---

### **3.7 Modalità D'Integrazione nelle APP al cittadino**

Le pagine di autorizzazione sono erogate da IdPC che, senza informazioni esplicite, non è in grado di determinare la natura dell'applicazione chiamante. Tramite un opportuno parametro nella richiesta d'autorizzazione e sottomissione PIN è possibile indicare al sistema di API Management la natura dell'applicazione client, cioè se si tratta di una APP per dispositivi mobili o una "normale" navigazione da web browser (non importa se da dispositivo mobile o desktop).

L'indicazione che la navigazione viene effettuata in-APP è utilizzata dall'API Manager per escludere dalle pagine presentate all'utente le testate ed i footer che caratterizzano la navigazione da web browser al fine di consentire un'integrazione semplificata nell'ambito della WebView che l'APP istanzia nelle fasi d'autorizzazione e sottomissione PIN.

Tale modalità pertanto si applica unicamente allo scenario cittadini.rl .

## 4 Interfacce esposte

### 4.1 Autenticazione del server

Le interfacce sono esposte tramite protocollo HTTP/S, cioè tramite protocollo SSL/TLS in “server authentication”, cioè con una modalità analoga a quella con cui viene esposto un sito web che faccia uso di certificati SSL/TLS *attendibili*. “Server Authentication” infatti, significa che il **client (il chiamante) ha la possibilità e la responsabilità (i.e. OBBLIGO) di autenticare il server con cui sta interagendo**, cioè l’API Manager.

**L’autenticazione del server con cui è stata aperta la connessione deve essere completata con esito positivo prima di trasmettere informazioni d’autenticazione o dati applicativi** al fine di prevenire un possibile attacco *man-in-the-middle*.

L’autenticazione del server deve comprendere almeno le seguenti verifiche:

- **Verifica del “Common Name” del certificato server** ottenuto all’apertura della connessione SSL/TLS che deve essere valorizzato con **api.lispa.it** quando viene invocato il sistema di produzione o, quando viene invocato quello di integrazione/test, **api.integrazione.lispa.it**;
- **Verifica che il certificato server ottenuto all’apertura della connessione SSL/TLS sia stato emesso dalla CA utilizzata da Lombardia Informatica / Regione Lombardia** per l’emissione dei certificati dei server.

Il processo di verifica deve tenere conto che nel tempo i certificati dell’API Manager vengono sostituiti per scadenza e/o cambiamento del fornitore dei servizi di CA di Lombardia Informatica che viene periodicamente riassegnato tramite gara; ciò comporta, con cadenza più o meno annuale, la sostituzione del certificato del server e, con una frequenza tipicamente inferiore, quello della CA.

*Considerata l’importanza dei requisiti espressi in questo capitolo questi contenuti sono ripetuti nella parte riguardante le modalità di invocazione delle API.*

### 4.2 Autorizzazione dell’applicazione da parte dell’utente

NOTA: Non applicabile allo scenario servizi.rl

In fase iniziale, cioè quando l’applicazione client dispone ancora di un Refresh Token valido, è necessario effettuare la fase d’autorizzazione al fine di ottenere il rilascio di un Authorization Code; tale quantità è prodotta al termine di un processo che richiede l’autenticazione web dell’utente e l’espressione della volontà di questi di autorizzare l’accesso.

Una volta ottenuto l’Authorization Code, l’applicazione deve invocare immediatamente l’API del API Manager che esegue lo scambio dell’Authorization Code con i *tokens* necessari all’attività operativa. Una volta ottenuti i *tokens* l’applicazione ha concluso la fase d’autorizzazione e, in linea generale, può proseguire nella propria operatività senza doverla ripetere.

Si noti, però, che potrebbe essere necessario ripetere la fase di autorizzazione a fronte di eventi, quali:

- la cancellazione dell’autorizzazione da parte dell’utente effettuabile tramite la specifica API [cittadini.rl, operatori.siss];
- l’invalidazione causata dall’esaurimento del numero di sottomissioni errate del PIN autorizzativo [cittadini.rl];
- il logout dal SISS [operatori.siss].

#### 4.2.1 Authorize

Questa interfaccia consente all'APP di ottenere l'Authorization Code dal servizio dell'API Manager. Poiché questa fase richiede interazione non mediata con l'utente da parte dell'API Manager, l'applicazione deve attivare l'interfaccia tramite la navigazione in una propria web view/page verso la specifica URL dell'API Manager componendo la query string della URL come specificato nel seguito.

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/oauth2/authorize>

#### 4.2.1.1 Invocazione

Parametri della query string:

Parametro	Valore	Descrizione
<code>response_type</code>	<code>code</code>	Utilizzare quanto specificato nella colonna Valore. Si richiede la generazione di un authorization <i>code</i>
<code>client_id</code>		Codice identificativo della specifica applicazione, generato centralmente e messo a disposizione dello sviluppatore dell'applicazione
<code>redirect_uri</code>		URL finale di redirect nella forma, ad es., <a href="http://localhost:port/...">http://localhost:port/...</a> a cui l'applicazione web d'autorizzazione dovrà effettuare la redirect per comunicare l'esito della fase di autorizzazione. Il valore deve corrispondere a quanto associato centralmente allo specifica applicazione identificata tramite un <code>client_id</code> univoco.
<code>Scope</code>	Set di stringhe separate da uno spazio	Identifica gli ambiti (sottoservizi) per cui si richiede l'autorizzazione. Questi parametri possono essere utilizzati nella schermata di autorizzazione presentata all'utente . <i>Opzionale se non richiesto dalle API o dall'esigenza di specificare un device_</i>
<code>friendlyName</code>	<code>SISSMobile</code>	Parametro OPZIONALE che, se specificato, ottimizza la presentazione per l'impiego all'interno di WebView di APP per dispositivi mobili; solo per lo scenario [cittadini.rl].

#### 4.2.1.2 Risposta

La navigazione all'interno della web view è pilotata dalla web application d'autorizzazione dell'API Manager che la conclude, comunicando l'esito, tramite una redirect alla `redirect_uri` specificata in sede d'invocazione del servizio, che deve corrispondere a quella registrata nel sistema al momento della generazione dell'applicazione client.

Alla URI viene aggiunta una *query string* in cui è codificato l'esito dell'operazione.

##### Risposta positiva

Parametro	Valore	Descrizione
<code>code</code>		Valore dell'Authorization Code da fornire con l'API

##### Risposta negativa

Parametro	Valore	Descrizione
<code>error</code>	<code>access_denied</code>	La fase d'autenticazione o autorizzazione non si è conclusa con successo o è stata annullata dall'utente.

---

### 4.2.1.3 Esempio d'invocazione

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Ad esempio supponendo:

ClientID = "yPR99PlfQGC8rHthisHq6halrB0a"

ClientSecret = "A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a"

RedirectURI = "<https://www.mycallback.it:9000/>"

Volendo ottenere l'autorizzazione per lo scope "jwt" associato al dispositivo "ipad":

Elemento	Contenuto
URL	<a href="https://api.lispa.it/oauth2/authorize?response_type=code&amp;client_id=yPR99PlfQGC8rHthisHq6halrB0a&amp;scope=jwt%20device_ipad&amp;redirect_uri=https://www.mycallback.it:9000/">https://api.lispa.it/oauth2/authorize?response_type=code&amp;client_id=yPR99PlfQGC8rHthisHq6halrB0a&amp;scope=jwt%20device_ipad&amp;redirect_uri=https://www.mycallback.it:9000/</a>

#### Esempio di risposta positiva

Se l'autorizzazione è andata a buon fine viene eseguita la redirect in alla "redirect URI" con l'autorization code da utilizzare immediatamente per ottenere un AT ed un RT.

```
https://www.mycallback.it:9000/?code=0b336ff0b4f354118e80781d5dfac2ee
```

#### Esempio di risposta negativa

In caso d'errore viene eseguita la redirect alla "redirect URI" dell'applicazione con l'indicazione della condizione d'errore.

```
https://www.mycallback.it:9000/?error=access_denied
```

## 4.2.2 Utilizzo dell'Authorization Code per ottenere i token

Questa interfaccia consente all'applicazione client, fornendo un Authorization Code valido, di ottenere i *token* necessari per attivare l'erogazione delle proprie funzionalità applicative e disporre delle quantità necessarie per il rinnovo dei *token* necessari per l'accesso alle API.

I *token* prodotti dalla API sono:

- Refresh Token      Mantenuto persistente nel dispositivo dall'APP [cittadini.rl] o temporaneo nella sessione del web server [operatori.siss] per poter successivamente richiedere la generazione di un nuovo Access Token valido;
- Access Token      Token d'accesso utilizzato nell'invocazione delle API dei servizi esposti; ha una validità temporale limitata (15-30 minuti) e deve essere rinnovato, all'occorrenza, tramite il Refresh Token;

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/oauth2/token>

### 4.2.2.1 Invocazione

Una volta ottenuto l'Authorization Code, l'applicazione deve utilizzarlo tempestivamente per chiederne la conversione nei *token* sopra descritti.

Questa operazione viene effettuata tramite una **POST** con le seguenti caratteristiche:

Elemento	Contenuto		
Metodo	POST		
URL	<a href="https://api.lispa.it/oauth2/token">https://api.lispa.it/oauth2/token</a>		
Headers	Authorization: Basic <u>valore</u> Content-Type: application/x-www-form-urlencoded		
Content	Parametro	Valore	Descrizione
	grant_type	authorization_code	Utilizzare quanto specificato nella colonna Valore. Come definito nelle specifiche di OAuth2 deve contenere la stringa <u>authorization_code</u>
	code		Valore dell' <i>authorization code</i> ottenuto in fase d'autenticazione
	redirect_uri		Il valore deve corrispondere a quanto associato centralmente alla specifica applicazione

Dove valore è costituito dalla codifica BASE64 della stringa *ClientID:ClientSecret*

### 4.2.2.2 Risposta

La risposta dell'invocazione al metodo è determinata in prima istanza dallo status code http e, in seconda istanza, dall'eventuale documento XML di fault restituito dal metodo.

Gli status code http sono utilizzati nel seguente modo:

- 20X      Esito ok, l'oggetto JSON restituito contiene la risposta applicativa;
- 40X      Richiesta malformata o che non ha soddisfatto i criteri d'autorizzazione viene restituito un documento XML di fault con i dettagli dell'eccezione
- 50X      Errore interno del server;



## Risposta positiva

Nel caso in cui la richiesta sia stata soddisfatta viene restituito l'http status code 200 con il seguente content:

Parametro	Valore	Descrizione
access_token		Valore dell'Access Token
refresh_token		Valore del Refresh Token
expires_in		Durata rimanente dell'Access Token in secondi
token_type	Bearer	Viene sempre restituito il valore Bearer
scope		Elenco degli scope autorizzati separati da 'spazio'. Opzionale se non richiesto dalle API o dall'esigenza di specificare un device_

### 4.2.2.3 Esempio d'invocazione

Ad esempio supponendo:

ClientID = "yPR99PlfQGC8rHthisHq6halrB0a"

ClientSecret = "A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a"

RedirectURI = "<https://www.mycallback.it:9000/>"

Avendo richiesto l'autorizzazione per lo scope "jwt" associato al dispositivo "ipad" il valore dell'header d'autorizzazione vale Base64(yPR99PlfQGC8rHthisHq6halrB0a:A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a), cioè eVBSOTlQbGZRR0M4ckh0aGlzSHE2aGFsckIwYTpBNU5GMXM5bkJtX1FfUXhfVkJzNDNlazdSRjRh

Supponendo inoltre che l'authorization code ottenuto precedentemente valga 0b336ff0b4f354118e80781d5dfac2ee

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
Metodo	POST
URL	<a href="https://api.lispa.it/oauth2/token">https://api.lispa.it/oauth2/token</a>
Headers	Authorization: Basic eVBSOTlQbGZRR0M4ckh0aGlzSHE2aGFsckIwYTpBNU5GMXM5bkJtX1FfUXhfVkJzNDNlazdSRjRh Content-Type: application/x-www-form-urlencoded
Content	grant_type=authorization_code&code=0b336ff0b4f354118e80781d5dfac2ee& redirect_uri=https://www.mycallback.it:9000/

### Esempio di risposta positiva

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

```
{
  "scope": "jwt device_ipad",
  "token_type": "Bearer",
  "expires_in": 1800,
  "refresh_token": "2a5eb3e57e53d43fa88f1b057cab7732",
  "access_token": "71dd5ae7f2c47eeca445f7375a9d80c"
}
```

## 4.3 Autorizzazione della sola applicazione

Questo è lo scenario inerente lo scenario (tenant) **servizi.rl**, nel quale l'autenticazione è limitata alla sola applicazione senza coinvolgimento dell'utente finale dell'applicazione stessa, adatto quindi a scenari B2B (business to business).

### 4.3.1 Rilascio Access Token

Il rilascio di un Access Token avviene tramite la presentazione delle credenziali dell'applicazione (ClientID e ClientSecret). Il periodo di validità può essere determinato in modo preventivo utilizzando il valore dell'attributo *expires\_in* restituito dalle API dell'API Manager o, più semplicemente, come reazione automatica alla specifica risposta d'errore prodotta dalla richiesta di una API quanto viene utilizzato un Access Token scaduto.

Questo scenario non prevede il rinnovo dell'AT tramite RT; l'AT viene rinnovato eseguendo nuovamente la richiesta dell'Access Token con le credenziali dell'applicazione client.

Come per il processo d'autorizzazione, dove richiesto, vanno specificati gli *scope* per cui si richiede l'accesso. Nel caso di un'applicazione client in configurazione multinodo/cluster si raccomanda l'utilizzo dello scope *device\_XXX* per specificare un riferimento al nodo fisico che ha richiesto l'AT (dove ad es XXX è costituito dal nome macchina del server); con tale accorgimento gli AT rilasciati sono differenti così come sono indipendenti i cicli di rilascio e la scadenza del periodo di validità.

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/oauth2/token>

#### 4.3.1.1 Invocazione

Una volta determinato che l'Access Token ha esaurito la propria l'applicazione deve utilizzare questa interfaccia per ottenerne uno nuovo.

Questa operazione viene effettuata tramite una **POST** con le seguenti caratteristiche:

Elemento	Contenuto		
Metodo	POST		
URL	<a href="https://api.lispa.it/oauth2/token">https://api.lispa.it/oauth2/token</a>		
Headers	Authorization: Basic <u>valore</u> Content-Type: application/x-www-form-urlencoded		
Content	Parametro	Valore	Descrizione
	<b>grant_type</b>	<b>client_credentials</b>	Utilizzare quanto specificato nella colonna Valore. Come definito nelle specifiche di OAuth2 deve contenere la stringa <b>client_credentials</b>
	<b>scope</b>	Set di stringhe separate da uno spazio	Identifica gli ambiti (sottoservizi) per cui si richiede l'autorizzazione. Questi parametri possono essere utilizzati nella schermata di autorizzazione presentata all'utente. Opzionale se non richiesto dalle API o dall'esigenza di specificare un <i>device_</i>

Dove valore è costituito dalla codifica BASE64 della stringa *ClientID:ClientSecret*

### 4.3.1.2 Risposta

La risposta dell'invocazione al metodo è determinata in prima istanza dallo status code http e, in seconda istanza, dall'eventuale documento XML di fault restituito dal metodo.

Gli status code http sono utilizzati nel seguente modo:

- 20X Esito ok, l'oggetto JSON restituito contiene la risposta applicativa;
- 40X Richiesta malformata o che non ha soddisfatto i criteri d'autorizzazione viene restituito un documento XML di fault con i dettagli dell'eccezione
- 50X Errore interno del server;

#### Risposta positiva

Nel caso in cui la richiesta sia stata soddisfatta viene restituito l'http status code 200 con il seguente content:

Parametro	Valore	Descrizione
<b>access_token</b>		Valore dell'Access Token
<b>expires_in</b>		Durata rimanente dell'Access Token in secondi
<b>token_type</b>	<b>Bearer</b>	Viene sempre restituito il valore Bearer
<b>scope</b>		Elenco degli scope autorizzati separati da ' spazio'. Opzionale se non richiesto dalle API o dall'esigenza di specificare un <i>device_</i>

### 4.3.1.3 Esempio d'invocazione

Ad esempio supponendo:

ClientID = "yPR99PlfQGC8rHthisHq6halrB0a"

ClientSecret = "A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a"

RedirectURI = "<https://www.mycallback.it:9000/>"

Avendo richiesto l'autorizzazione per lo scope "jwt" associato al dispositivo "ipad" il valore dell'header d'autorizzazione vale Base64(yPR99PlfQGC8rHthisHq6halrB0a:A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a), cioè eVBSOTIQbGZRR0M4ckh0aGlzSHE2aGFscklwYTpBNU5GMXM5bkJtX1FfUXhfVkJxNDNlazdSRjRh

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
<b>Metodo</b>	POST
<b>URL</b>	<a href="https://api.lispa.it/oauth2/token">https://api.lispa.it/oauth2/token</a>
<b>Headers</b>	Authorization: Basic eVBSOTIQbGZRR0M4ckh0aGlzSHE2aGFscklwYTpBNU5GMXM5bkJtX1FfUXhfVkJxNDNlazdSRjRh Content-Type: application/x-www-form-urlencoded
<b>Content</b>	grant_type=client_credentials&scope=documentale

#### Esempio di risposta positiva

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

{
  "scope": "am_application_scope documentale",
  "token_type": "Bearer",
  "expires_in": 1800,
  "access_token": "0f5afe9143935da856e1a45fb623f64f"
}
```



---

## 4.4 *Fruizione delle API*

Per l'accesso alle API è necessario disporre di un Access Token valido. Nel caso d'accesso ad API che richiedono il PIN (Codice di Sicurezza) come criterio aggiuntivo d'autorizzazione è necessario disporre anche di un Grant Token correlato al *token* d'accesso, anch'esso in corso di validità. Le modalità per ottenere un Grant Token sono descritte nei capitoli successivi.

Nel seguito sono definite le interfacce che supportano l'operatività di fruizione:

### **API Call**

Modalità per l'invocazione di una API del servizio esposto da RL/LI; si noti che il ruolo dell'API Manager, in questo caso, è di puro "proxy" delle richieste REST/SOAP effettuate dall'applicazione a servizi applicativi che il Manager inoltra in modo trasparente a valle dell'attività d'autorizzazione. L'invocazione delle API richiede l'uso di un Access Token e, se richiesto dalla specifica API, anche di un Grant Token.

### **Authorization PIN**

Interfaccia per fare in modo che l'API Manager interagisca in modalità web browsing con l'utente al fine di ottenere l'inserimento del PIN d'autorizzazione (Codice di Sicurezza) e, se questo corrisponde a quello definito in sede di autorizzazione/rinnovo, produrre un Grant Token correlato all'Access Token trasmesso dall'applicazione tra i parametri dell'interfaccia. Il Grant Token perde di validità con lo scadere dell'Access Token a cui è correlato.

Questa interfaccia non è applicabile ai servizi del contesto [servizi.rl] .

### **Refresh token**

Lo scadere di un Access Token può essere determinato in modo preventivo utilizzando il valore dell'attributo *expires\_in* restituito dalle API del Manager o, più semplicemente, come reazione automatica alla specifica risposta d'errore prodotta dalla richiesta di una API quanto viene utilizzato un Access Token scaduto. Nel secondo caso è richiesto all'APP di eseguire il rinnovo del token e risottomettere la richiesta con token valido automaticamente senza informare l'utente o richiedere un suo intervento.

L'API di rinnovo richiede tra i parametri un Refresh Token valido e restituisce un nuovo Access Token; si noti che l'API rigenera anche il Refresh Token, il cui valore dovrà essere sostituito nella memoria locale persistente dell'applicazione o nella sessione del web server ed essere utilizzato nella successiva fase di rinnovo. Se venisse perso il sincronismo sarebbe necessario ripetere la fase d'autorizzazione.

---

#### 4.4.1 API Call

Modalità per l'invocazione di una API del servizio esposto da RL/LI; si noti che il ruolo dell'API Manager, in questo caso, è di puro "proxy" delle richieste REST/SOAP effettuate dall'applicazione client a servizi applicativi a cui il Manager inoltra in modo trasparente a valle dell'attività d'autorizzazione. L'invocazione delle API richiede l'uso di un Access Token valido e, se richiesto dalla specifica API, anche di un Grant Token.

La URI d'esposizione e l'interfaccia REST/SOAP delle API sono specificate dai servizi espositori negli appositi documenti.

Nel seguito sono specificate le necessità aggiuntive richieste dall'API Manager per poter espletare il ruolo di componente d'autorizzazione e, nel fare questo, produrre specifiche risposte d'errore qualora fosse determinato un problema prima di propagare la richiesta al servizio erogante l'API. Nel caso in cui l'errore non sia generato da l'API Manager occorre riferirsi alle specifiche d'interfaccia dell'API invocata.

---

##### 4.4.1.1 Autenticazione del server

Le API sono esposte tramite protocollo HTTP/S, cioè tramite protocollo SSL/TLS in "server authentication", cioè con una modalità analoga a quella con cui viene esposto un sito web che faccia uso di certificati SSL/TLS *attendibili*. "Server Authentication" infatti, significa che il **client (il chiamante delle API) ha la possibilità e la responsabilità (i.e. OBBLIGO) di autenticare il server con cui sta interagendo**, cioè l'API Manager.

**L'autenticazione del server con cui è stata aperta la connessione deve essere completata con esito positivo prima di trasmettere informazioni d'autenticazione o dati applicativi al fine di prevenire un possibile attacco *man-in-the-middle*.**

L'autenticazione del server deve comprendere almeno le seguenti verifiche:

- **Verifica del "Common Name" del certificato server** ottenuto all'apertura della connessione SSL/TLS che deve essere valorizzato con **api.lispa.it** quando viene invocato il sistema di produzione o, quando viene invocato quello di integrazione/test, **api.integrazione.lispa.it**;
- **Verifica che il certificato server ottenuto all'apertura della connessione SSL/TLS sia stato emesso dalla CA utilizzata da Lombardia Informatica / Regione Lombardia** per l'emissione dei certificati dei server.

Il processo di verifica deve tenere conto che nel tempo i certificati dell'API Manager vengono sostituiti per scadenza e/o cambiamento del fornitore dei servizi di CA di Lombardia Informatica che viene periodicamente riassegnato tramite gara; ciò comporta, con cadenza più o meno annuale, la sostituzione del certificato del server e, con una frequenza tipicamente inferiore, quello della CA.

---

##### 4.4.1.2 Invocazione

L'invocazione di un'API REST/SOAP richiede l'inserimento di uno specifico http *header* aggiuntivo per la trasmissione dell'Access Token. L'API Manager utilizza l'Access Token ed i profili del servizio e dell'utente referenziati dal *token* per applicare i criteri d'autorizzazione.

Nel caso in cui l'API sia stata definita centralmente come assogettata a PIN, l'applicazione client deve inserire nella richiesta REST/SOAP un ulteriore http *header* aggiuntivo per la trasmissione del Grant Token.

L'invocazione deve essere effettuata tramite una chiamata REST/SOAP, cioè una http GET o POST contenente l'http *header* aggiuntivo **Authorization** a sua volta contenente la stringa **Bearer** seguita da uno spazio e, successivamente, dalla stringa rappresentante il valore dell'Access Token ottenuto in precedenza utilizzando le interfacce definite in questo documento.

Nel caso sia richiesto anche il PIN, l'invocazione deve essere inserendo anche l'ulteriore http header **AuthorizationGrant** a sua volta contenente la stringa **Bearer** seguita da uno spazio e, successivamente, dalla stringa rappresentante il valore del Grant Token ottenuto in precedenza utilizzando le interfacce definite in questo documento.

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/...>

dove i puntini finali indicano la risorsa invocata (es. <https://api.lispa.it/t/cittadini.rl/calc/1.0/multiply>) come indicato dalla documentazione della API.

---

### 4.4.1.3 Risposta

#### Risposta positiva

Quando l'API Manager rileva una condizione per cui deve rifiutare la chiamata all'API genera una risposta di fault in XML veicolata nel *content* della risposta ed identificata da uno status code http 40X. Lo status code http 200 (OK) o qualunque altra condizione d'errore sono prodotte dal servizio invocato.

#### Risposta negativa

La risposta d'errore prodotta dal Manager è caratterizzata in prima istanza dallo status code http nel range 40X e, in seconda istanza, da un documento XML di fault restituito nel *content* della risposta. Se nella richiesta viene inserito l'header http:

**Accept: application/json**

La risposta d'errore viene restituita in forma JSON anziché nel formato di default XML.

E' necessario che l'applicazione gestisca in modo controllato i fault relativi ad Access Token e Grant Token invalidi o scaduti in quanto la situazione può essere recuperata in modo automatico per l'AT ed in modo controllato per il GT, senza allarmare l'utente finale.

---

### 4.4.1.4 Esempio d'invocazione

Ad esempio supponendo di chiamare un ipotetico servizio REST "calc" che espone il metodo "multiply", si utilizza l'Access Token ottenuto precedentemente (ad es. "71dd5ae7f2c47eeca445f7375a9d80c") per invocare l'API. Si noti che i valori del Content-Type e del Content sono definiti dall'espositore dell'API.

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
Metodo	GET
URL	<a href="https://api.lispa.it/t/cittadini.rl/calc/1.0/multiply?x=7&amp;y=5">https://api.lispa.it/t/cittadini.rl/calc/1.0/multiply?x=7&amp;y=5</a>
Headers	Authorization: Bearer 45739fd26e4569298420788557fc7ee9
Content	

**Esempio di richiesta con Grant Token** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Nel caso in cui l'API avesse richiesto la dimostrazione della sottomissione del PIN sarebbe stato trasmesso anche il Grant Token (ad es. di valore "DtFj3Z9FGj2wtGy6NIMgyr7u%2BZrvxflWKgqPIWG5XZKmrwaYXNpOqZaMEADV%2Fdvu").

Elemento	Contenuto
Metodo	GET
URL	<a href="https://api.lispa.it/t/cittadini.rl/calc/1.0/multiply?x=7&amp;y=5">https://api.lispa.it/t/cittadini.rl/calc/1.0/multiply?x=7&amp;y=5</a>
Headers	Authorization: Bearer 45739fd26e4569298420788557fc7ee9 AuthorizationGrant: Bearer DtFj3Z9FGj2wtGy6NIMgyr7u%2BZrvxflWKgqPIWG5XZKmrwaYXNpOqZaMEADV%2Fdvu

Content	
---------	--

### Esempio di risposta positiva

```
200 OK
Content-Type: text/plain; charset=UTF-8

{
  "answer": "35.0"
}
```

### Esempio di risposta negativa

```
403 Forbidden
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0"?>
<ams:fault xmlns:ams="http://wso2.org/apimanager/security">
  <ams:code>900908</ams:code>
  <ams:message>Resource forbidden</ams:message>
  <ams:description>Access failure for API: /t/cittadini.rl/calc/multiply/1.0, version:
1.0</ams:description>
</ams:fault>
```



Oppure se è stata richiesta la restituzione degli errori in formato JSON (tramite l'header http **Accept: application/json**):

```
403 Forbidden
Content-Type: application/json

{
  "fault":
  {
    "code": 900908,
    "message": "Resource forbidden",
    "description": "Access failure for API: /t/cittadini.rl/calc/multiply/1.0, version: 1.0"
  }
}
```

## 4.4.2 Sottomissione authorization PIN (Codice di Sicurezza)

Interfaccia per fare in modo che l'API Manager interagisca in modalità web browsing con l'utente al fine di ottenere l'inserimento del PIN d'autorizzazione e, se questo corrisponde a quello definito in sede di autorizzazione, produrre un Grant Token correlato all'Access Token inoltrato dall'applicazione client tra i parametri dell'interfaccia. Il Grant Token perde di validità con lo scadere dell'Access Token a cui è correlato.

Questa interfaccia consente all'applicazione di ottenere un Grant Token dal servizio dell'API Manager. Poiché questa fase richiede interazione non mediata con l'utente da parte del Manager, l'applicazione deve attivare l'interfaccia tramite la navigazione ad una specifica URL del Gateway in una propria web view componendo la *query string* della URL come specificato nel seguito.

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/lispaauthenticationendpoint/authpin.do>

### 4.4.2.1 Invocazione

Parametri della query string:

Parametro	Valore	Descrizione
<b>redirect_uri</b>		URL finale di redirect nella forma <b>http://localhost:port/...</b> a cui l'applicazione web d'autorizzazione dovrà effettuare la redirect per comunicare l'esito della fase di autorizzazione. Il valore deve corrispondere a quanto associato centralmente allo specifica APP identificata tramite un <b>client_id</b> univoco.
<b>friendlyName</b>	<b>SISSMobile</b>	Parametro OPZIONALE che, se specificato, ottimizza la presentazione per l'impiego all'interno di WebView di APP per dispositivi mobili; solo per lo scenario [cittadini.rl].

Nell'http header Authorization deve essere utilizzato per trasmettere il valore dell'Access Token corrente, in corso di validità.

http header	Valore	Descrizione
<b>Authorization</b>		Access Token ottenuto in fase di autorizzazione o tramite l'uso dell'interfaccia di refresh dei token

Il ContentType deve essere: application/x-www-form-urlencoded

---

#### 4.4.2.2 Risposta

La navigazione all'interno della web view è pilotata dalla web application d'autorizzazione dell'API Manager che la conclude, comunicando l'esito, con una redirect alla **redirect\_uri** specificata in sede d'invocazione del servizio.

Alla URI viene aggiunta una *query string* in cui è codificato l'esito dell'operazione.

Nel caso in cui la risposta negativa consista in una segnalazione di PIN bloccato è responsabilità dell'applicazione client ripetere la fase di autorizzazione dell'utente. L'API Manager nel ripetere l'autorizzazione rileverà il blocco del PIN e guiderà l'utente nel reimpostarlo.

La funzione di cambio PIN su iniziativa dell'utente o del sistema è effettuata nell'ambito della GUI web inerente il processo di sottomissione del PIN stesso e non richiede alcuna integrazione da parte dell'applicazione client.

##### Risposta positiva

Parametro	Valore	Descrizione
<b>grant</b>		Valore del Grant Token da fornire con l'API

##### Risposta negativa

Parametro	Valore	Descrizione
<b>error</b>	<b>access_denied</b>	La fase di sottomissione del PIN all'API Manager non si è conclusa con successo o è stata annullata dall'utente; non sono stati esauriti i tentativi di sottomissione errata del PIN. Questa interfaccia può essere attivata nuovamente in futuro.
<b>error</b>	<b>access_blocked</b>	La fase di sottomissione del PIN all'API Manager non si è conclusa con ed inoltre sono stati esauriti i tentativi di sottomissione errata del PIN. Il PIN è stato annullato dal Manager e l'unico modo di ripristinare il funzionamento dell'applicazione è ripetere la fase di autorizzazione conformemente a quanto specificato nel presente documento, definendo nuovamente un PIN d'accesso.

---

#### 4.4.2.3 Esempio d'invocazione

Supponendo che l'Access Code ottenuto precedentemente valga 45739fd26e4569298420788557fc7ee9

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
<b>Metodo</b>	GET
<b>URL</b>	https://api.lispa.it/lispaauthenticationendpoint/authpin.do?redirect_uri=https://www.mycallback.it:9000/
<b>Headers</b>	Authorization: 45739fd26e4569298420788557fc7ee9 Content-Type: application/x-www-form-urlencoded
<b>Content</b>	

### Esempio di risposta positiva

Se l'autorizzazione è andata a buon fine, esegue la redirect in alla redirect URI con il Grant Toche da utilizzare unitamente all'AT a cui è correlato.

```
https://www.mycallback.it:9000/?grant=DtFj3Z9FGj2wtGy6NlMgyr7u%2BZrvxf1WKgqPlWG5XZKmrwaYXNpOqZaMEADV%2Fdvu
```

### Esempi di risposta negativa

In caso d'errore esegue la redirect in local host con l'indicazione della condizione riscontrata: in questo caso il PIN non è stato sottomesso ma non si sono esauriti i tentativi disponibili: è possibile riprovare in futuro.

```
https://www.mycallback.it:9000/?error=access_denied
```

In questo caso il PIN si sono esauriti i tentativi disponibili: è necessario ripetere il processo di Pairing.

```
https://www.mycallback.it:9000/?error=access_blocked
```

---

### 4.4.3 Refresh token

Lo scadere di un Access Token può essere determinato in modo preventivo utilizzando il valore dell'attributo *expires\_in* restituito dalle API del Manager o, più semplicemente, come reazione automatica alla specifica risposta d'errore prodotta dalla richiesta di una API quanto viene speso un Access Token scaduto. Nel secondo caso è richiesto all'applicazione client di eseguire il rinnovo del token e risottomettere la richiesta con token valido automaticamente senza informare l'utente o richiedere un suo intervento.

L'API di rinnovo richiede tra i parametri un Refresh Token valido e restituisce un nuovo Access Token; si noti che l'API rigenera anche il Refresh Token, il cui valore dovrà essere sostituito nella memoria locale persistente dell'APP o nella sessione del web server.

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/oauth2/token>

---

#### 4.4.3.1 Invocazione

Una volta determinato che l'Access Token ha esaurito la propria validità, l'APP può utilizzare il Refresh Token per ottenerne uno nuovo.

Questa operazione viene effettuata tramite una **POST** con le seguenti caratteristiche:

Elemento	Contenuto		
Metodo	POST		
URL	<a href="https://api.lispa.it/oauth2/token">https://api.lispa.it/oauth2/token</a>		
Headers	Authorization: Basic <i>valore</i> Content-Type: application/x-www-form-urlencoded		
Content	Parametro	Valore	Descrizione
	grant_type	refresh_token	Utilizzare quanto specificato nella colonna Valore. Come definito nelle specifiche di OAuth2 deve contenere la stringa <i>authorization_code</i>
	refresh_token		Valore del Refresh Token ottenuto in fase d'autorizzazione o di precedenti rinnovi

Dove *valore* è costituito dalla codifica BASE64 della stringa *ClientID:ClientSecret*

---

#### 4.4.3.2 Risposta

La risposta dell'invocazione al metodo è determinata in prima istanza dallo status code http e, in seconda istanza, dall'eventuale documento XML di fault restituito dal metodo.

Gli status code http sono utilizzati nel seguente modo:

- 20X Esito ok, l'oggetto JSON restituito contiene la risposta applicativa;
- 40X Richiesta malformata o che non ha soddisfatto i criteri d'autorizzazione viene restituito un documento XML di fault con i dettagli dell'eccezione
- 50X Errore interno del server;

### Risposta positiva

Nel caso in cui la richiesta sia stata soddisfatta viene restituito l'http status code 200 con il seguente content:

Parametro	Valore	Descrizione
access_token		Valore dell'Access Token
refresh_token		Valore del Refresh Token
expires_in		Durata rimanente dell'Access Token in secondi
token_type	Bearer	Viene sempre restituito il valore Bearer
scope		Elenco degli scope autorizzati separati da 'spazio'. Opzionale se non richiesto dalle API o dall'esigenza di specificare un device_

#### 4.4.3.3 Esempio d'invocazione

Ad esempio supponendo:

ClientID = "yPR99PlfQGC8rHthisHq6halrB0a"

ClientSecret = "A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a"

RedirectURI = "<https://www.mycallback.it:9000/>"

Avendo richiesto l'autorizzazione per lo scope "jwt" associato al dispositivo "ipad" il valore dell'header d'autorizzazione vale Base64(yPR99PlfQGC8rHthisHq6halrB0a:A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a), cioè eVBSOTIqbGZRR0M4ckh0aGlzSHE2aGFsckIwYTpBNU5GMXM5bkJtX1FfUXhfVkJzNDNlazdSRjRh

Supponendo inoltre che l'authorization code ottenuto precedentemente valga 0b336ff0b4f354118e80781d5dfac2ee

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
Metodo	POST
URL	<a href="https://api.lispa.it/oauth2/token">https://api.lispa.it/oauth2/token</a>
Headers	Authorization: Basic eVBSOTIqbGZRR0M4ckh0aGlzSHE2aGFsckIwYTpBNU5GMXM5bkJtX1FfUXhfVkJzNDNlazdSRjRh Content-Type: application/x-www-form-urlencoded
Content	grant_type=refresh_token&refresh_token=8bb3783a6c1215a47174598721433539

#### Esempio di risposta positiva

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

{
  "scope": "jwt device_ipad",
  "token_type": "Bearer",
  "expires_in": 1800,
  "refresh_token": "c947ae51c8fe921a723075ddcf8a7da2",
  "access_token": "0f5afe9143935da856e1a45fb623f64f"
}
```

---

## 4.5 Revoca del Refresh Token

Un'applicazione client può revocare un Refresh Token in qualunque momento, come ad esempio nel caso di richiesta esplicita dell'utente o disinstallazione dell'applicazione dal dispositivo.

---

### 4.5.1 API di Revoca

L'API di revoca consente la revoca del RT disponendo del RT o di un AT in corso di validità.

Si noti che la revoca impatta solo il Refresh Token e non AT e GT che saranno comunque utilizzabili fino allo scadere dell'AT.

---

#### 4.5.1.1 Invocazione

L'invocazione di dell'API richiede l'inserimento di specifici http *header* aggiuntiva per la trasmissione del Refresh Token o dell'Access Token e dell'indicazione di quale di questi è stato utilizzato

Il metodo è esposto alla seguente URL:

<https://api.lispa.it/oauth2/revoke>

Questa operazione viene effettuata tramite una **POST** con le seguenti caratteristiche:

Elemento	Contenuto		
Metodo	POST		
URL	<a href="https://api.lispa.it/oauth2/revoke">https://api.lispa.it/oauth2/revoke</a>		
Headers	Authorization: Basic <i>valore</i> Content-Type: application/x-www-form-urlencoded		
Content	Parametro	Valore	Descrizione
	<i>token_type_hint</i>	<i>refresh_token</i> oppure <i>access_token</i>	Natura del token utilizzato per la revoca del RT deve assumere uno dei due valori indicati.
	<i>token</i>		Valore del RT o dell'AT conformemente al parametro precedente

Dove *valore* è costituito dalla codifica BASE64 della stringa *ClientID:ClientSecret*

---

#### 4.5.1.2 Risposta

##### Risposta positiva

Quando l'API Manager deve abortire la chiamata all'API prima di invocare il servizio l'errore viene generato dal Manager stesso, per tale ragione il Manager non restituirà mai su propria iniziativa lo status code http 200 ma solo gli codici d'errore 40X. Qualunque altra condizione d'errore può essere stata prodotta unicamente dal servizio invocato.

##### Risposta negativa

La risposta d'errore prodotta dal Manager è caratterizzata i prima istanza dallo status code http nel range 40X e, in seconda istanza, da un documento XML di fault restituito nel content della risposta.

---

### 4.5.1.3 Esempio d'invocazione

Ad esempio supponendo:

ClientID = "yPR99PlfQGC8rHthisHq6halrB0a"

ClientSecret = "A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a"

Avendo richiesto l'autorizzazione per lo scope "jwt" associato al dispositivo "ipad" il valore dell'header d'autorizzazione vale Base64(yPR99PlfQGC8rHthisHq6halrB0a:A5NF1s9nBm\_Q\_Qx\_VFq43ek7RF4a), cioè eVBSOTIQbGZRR0M4ckh0aGlzSHE2aGFscklwYTpBNU5GMXM5bkJtX1FfUXhfVkJzNDNlazdSRjRh

Supponendo inoltre che l'autorization code ottenuto precedentemente valga 0b336ff0b4f354118e80781d5dfac2ee

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
Metodo	POST
URL	<a href="https://api.lispa.it/oauth2/revoke">https://api.lispa.it/oauth2/revoke</a>
Headers	Authorization: Basic eVBSOTIQbGZRR0M4ckh0aGlzSHE2aGFscklwYTpBNU5GMXM5bkJtX1FfUXhfVkJzNDNlazdSRjRh Content-Type: application/x-www-form-urlencoded
Content	token_type_hint=refresh_token&token= c947ae51c8fe921a723075ddcf8a7da2

### Esempio di risposta positiva

```
200 OK
Content-Type: text/html; charset=UTF-8
```

---

## 4.6 *Contesto associato alle API Call*

L'API Manager associa ad ogni API Call una struttura dati di contesto all'interno della quale colloca informazioni di contesto utili per i servizi invocati.

Per quanto questo aspetto non sia strettamente pertinente l'attività del fruitore delle API è comunque utile comprendere quali siano le informazioni di contesto che vengono "iniettate" nella richiesta entrante dall'API Gateway alla conclusione del processo d'autorizzazione.

La descrizione del contesto è inoltre propedeutica alla definizione dell'interfaccia del metodo di servizio WhoAmI esposto direttamente dall'API Gateway descritto nel seguito.

---

### 4.6.1 **Json Web Token (JWT)**

Il contesto delle richieste è descritto tramite un oggetto JSON denominato JWT, Json Web Token, dallo standard OAuth2.0; il contenuto di questo oggetto è largamente personalizzabile dagli implementatori dei Gateway e, pertanto, ne viene fornita la struttura adottata dal Gateway di RL/LI.

Struttura JSON del JWT, sono documentati solo gli elementi rilevanti dal punto di vista applicativo:

Parametro	Descrizione
<a href="http://wso2.org/claims/subscriber">http://wso2.org/claims/subscriber</a>	Utenza developer che ha registrato l'applicazione client
<a href="http://wso2.org/claims/applicationname">http://wso2.org/claims/applicationname</a>	Nome identificativo dell'applicazione client
<a href="http://wso2.org/claims/apicontext">http://wso2.org/claims/apicontext</a>	Contesto della richiesta: "t", identificativo del "tenant", "/", API invocata
<a href="http://wso2.org/claims/version">http://wso2.org/claims/version</a>	Versione della API invocata
<a href="http://wso2.org/claims/keytype">http://wso2.org/claims/keytype</a>	Tipologia d'accesso: se il valore è SANDBOX si tratta della modalità di test dell'applicazione client
<a href="http://wso2.org/claims/enduser">http://wso2.org/claims/enduser</a>	Identificativo dell'utente nella forma "CodiceFiscale@tenant"
<a href="http://wso2.org/claims/reale">http://wso2.org/claims/reale</a>	Presente solo nel contesto cittadini.rl: assume i valori "true"/"false" ad indicare se l'utenza è rispettivamente di tipo Reale o Virtuale
<a href="http://wso2.org/claims/fullname">http://wso2.org/claims/fullname</a>	Nome dell'utente (non presente per il contesto servizi.rl)
<a href="http://wso2.org/claims/lastname">http://wso2.org/claims/lastname</a>	Cognome dell'utente (non presente per il contesto servizi.rl)
<a href="http://wso2.org/claims/username">http://wso2.org/claims/username</a>	Codice Fiscale dell'utente (non presente per il contesto servizi.rl)
...	Eventuali ulteriori attributi rimappati partendo dall'asserzione d'identità che IdPC associa all'utente che ha eseguito il Pairing del dispositivo

---

#### 4.6.1.1 **Modalità di inoltro del JWT alle API esposte**

Il JWT (Json Web Token) è trasmesso tramite alla API invocata l'header **X-JWT-Assertion**; il valore dell'header è costituito dalla codifica *Base64url* dei bytes della rappresentazione testuale UTF-8 prodotta della serializzazione in testo dell'oggetto Json che costituisce il JWT.

Si noti che nell'header sono codificati e concatenati due distinti contenuti base64 separati dal carattere punto (.). Il primo contenuto è inerente la natura del JWT che, in generale, potrebbe essere cifrato. Il secondo contenuto è quello a cui si riferisce il paragrafo precedente in quanto contiene informazioni pertinenti l'erogazione dell'API da parte del servizio esposto.



---

### 4.6.1.2 Esempio di JWT

**Esempio** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Unbase64() del contenuto che precede il carattere ".":

```
{
  "typ": "JWT",
  "alg": "none"
}
```

Unbase64() del contenuto che segue il carattere ".":

```
{
  "iss": "wso2.org/products/am",
  "exp": 1463483066757,
  "http://wso2.org/claims/subscriber": "store1@cittadini.rl",
  "http://wso2.org/claims/applicationid": "3",
  "http://wso2.org/claims/applicationname": "DemoApp1",
  "http://wso2.org/claims/applicationtier": "Unlimited",
  "http://wso2.org/claims/apicontext": "/t/cittadini.rl/jwt/1.0.0",
  "http://wso2.org/claims/version": "1.0.0",
  "http://wso2.org/claims/tier": "Unlimited",
  "http://wso2.org/claims/keytype": "SANDBOX",
  "http://wso2.org/claims/usertype": "APPLICATION_USER",
  "http://wso2.org/claims/enduser": "PLNPLS61B09F205K@cittadini.rl",
  "http://wso2.org/claims/reale": "true",
  "http://wso2.org/claims/enduserTenantId": "1",
  "http://wso2.org/claims/fullname": "UGO",
  "http://wso2.org/claims/lastname": "FOSCOLO",
  "http://wso2.org/claims/role": "Internal/everyone",
  "http://wso2.org/claims/username": "PLNPLS61B09F205K"
}
```

---

## 4.7 API d'utilità

In questo capitolo sono documentate le interfacce di API d'utilità esposte direttamente dall'API Manager che, conseguentemente, non sono presenti nei documenti d'interfaccia delle API dei servizi applicativi di RL/LI.

---

### 4.7.1 API WhoAmI

Il protocollo OAuth2.0 è sbilanciato a favore delle componenti server e non definisce una modalità tramite cui l'API invocata possa disporre di informazioni di contesto inerente l'utente che ha effettuato l'autorizzazione all'accesso, se è prevista l'autorizzazione dell'utente, e comunque dell'applicazione client chiamante.

Tale lacuna è colmata dall'API Manager esponendo l'API **WhoAmI** che ha il solo scopo di restituire all'applicazione client un oggetto JWT derivato da quello iniettato dalle funzioni d'autorizzazione.

La conoscenza del contesto consente all'APP di avere informazioni anagrafiche dell'utente (ad es. Nome e Cognome) nonché il suo Codice Fiscale, utilizzabile come chiave univoca per l'identificazione del contesto locale persistente dell'utente in caso del supporto della condivisione del medesimo dispositivo da parte di più utenti. Il caso emblematico è la persistenza di uno specifico Refresh Token in una APP mobile distinguendo contesti separati per ciascun utente che utilizza il medesimo dispositivo.

Il metodo è esposto alle seguenti URL, che sono differenziate per tenant:

<https://api.lispa.it/t/cittadini.rl/whoami>

<https://api.lispa.it/t/operatori.siss/whoami>

<https://api.lispa.it/t/servizi.rl/whoami>

---

#### 4.7.1.1 Invocazione

Trattandosi di una API omogenea a quelle dei servizi applicativi, la modalità d'invocazione è conforme a quanto specificato nel capitolo *API Call*.

Questa API non ha parametri e non richiede la preventiva sottomissione del PIN.

---

#### 4.7.1.2 Risposta

La risposta dell'invocazione al metodo REST è caratterizzata dallo status code http 200 e da un *content* costituito da un oggetto JWT.

L'API non prevede condizioni d'errore specifiche in quanto queste trovano piena copertura tra quelle definite nel capitolo *API Call*.

---

#### 4.7.1.3 Esempio d'invocazione

**Esempio di richiesta** (i ritorni a capo sono utilizzati ai soli fini della leggibilità):

Elemento	Contenuto
Metodo	GET
URL	<a href="https://api.lispa.it/t/cittadini.rl/whoami">https://api.lispa.it/t/cittadini.rl/whoami</a>
Headers	Authorization: Bearer 45739fd26e4569298420788557fc7ee9

Content	
---------	--

### Esempio di risposta

200 OK  
Content-Type: application/json

```
{
  "iss": "wso2.org/products/am",
  "exp": 1463483066757,
  "http://wso2.org/claims/subscriber": "store1@cittadini.rl",
  "http://wso2.org/claims/applicationid": "3",
  "http://wso2.org/claims/applicationname": "DemoApp1",
  "http://wso2.org/claims/applicationtier": "Unlimited",
  "http://wso2.org/claims/apicontext": "/t/cittadini.rl/jwt/1.0.0",
  "http://wso2.org/claims/version": "1.0.0",
  "http://wso2.org/claims/tier": "Unlimited",
  "http://wso2.org/claims/keytype": "SANDBOX",
  "http://wso2.org/claims/usertype": "APPLICATION_USER",
  "http://wso2.org/claims/enduser": "PLNPLS61B09F205K@cittadini.rl",
  "http://wso2.org/claims/enduserTenantId": "1",
  "http://wso2.org/claims/fullname": "UGO",
  "http://wso2.org/claims/lastname": "FOSCOLO",
  "http://wso2.org/claims/role": "Internal/everyone",
  "http://wso2.org/claims/username": "PLNPLS61B09F205K"
}
```

---

## 4.8 *Requisiti di sicurezza aggiuntivi per le APP mobile*

In questo capitolo sono identificati gli aspetti di sicurezza inerenti l'integrazione dei servizi dell'API Manager, con particolare attenzione alle APP per dispositivi mobile anche se i concetti sono mutuabili anche in contesti differenti.

I requisiti nel seguito esposti sono **aggiuntivi** a quelli che caratterizzano la realizzazione in sicurezza di una generica APP per dispositivo mobile che, in ogni caso, devono essere soddisfatti dall'applicazione client anche se non citati nel presente documento.

---

### 4.8.1 Riservatezza dei segreti statici “immersi” nell'applicazione

Lo standard OAuth2.0 è consapevole che la quantità di sicurezza *client\_id* e *client\_secret* non possono essere considerate dei segreti in senso stretto in quanto devono essere “immersi” nel codice sorgente dell'APP.

Il rischio da mitigare è la possibilità per l'attaccante di creare un clone modificato dell'APP in grado di presentarsi all'API Manager in modo indistinguibile dalla versione lecita; in tale eventualità deve comunque essere effettuata la fase di autorizzazione (che in tutti gli scenari tranne uno richiede l'identificazione forte dell'utente) il quale, però, potrebbe essere stato indotto ad installare la versione artefatta al posto di quella originale.

Poiché l'APP può essere decompilata da un utente malevolo è comunque opportuno rendere quanto più complessa possibile tale attività, ad esempio tramite:

- Offuscamento del codice;
- Segmentazione delle quantità di sicurezza in più variabili da ricongiungere a run-time;
- Assegnando nomi non auto esplicativi alle costanti/variabili coinvolte in tale processo.

Qualunque ulteriore misura di sicurezza che l'APP implementi per assicurare la propria integrità può solo rafforzare la robustezza complessiva della soluzione.

---

### 4.8.2 Riservatezza del Refresh Token

Il Refresh Token deve essere necessariamente mantenuto nella memoria locale persistente del dispositivo, se così non fosse andrebbe ripetuta la fase di autorizzazione ad ogni utilizzo dell'APP.

E' richiesto che l'APP mantenga il Refresh Token in forma cifrata e che, analogamente a quanto specificato nel capitolo precedente, la chiave di cifratura sia custodita/generata con particolare attenzione alla decompilazione del SW eseguibile.

E' inoltre necessario che tale chiave sia ulteriormente “perturbata” con un valore calcolato tramite una funzione di “fingerprinting” del dispositivo. Vi sono molte tecniche con diversi livelli di complessità non identificabili in questo documento, è però necessario che tali tecniche rispettino i seguenti principi fondamentali:

- Riproducibilità – In momenti diversi, preferibilmente anche dopo un aggiornamento SW dell'S.O., la funzione di *fingerprinting* deve fornire il medesimo valore;
- Robustezza – Su dispositivi diversi la funzione di *fingerprinting* deve fornire valori differenti;
- Univocità – Devono essere adottate tutte le ragionevoli precauzioni affinché la memoria persistente in cui sono memorizzate le quantità di sicurezza non sia copiata all'esterno del dispositivo, ad esempio facendola escludere dalle funzioni di backup automatico previsto dal S.O. del dispositivo.

---

### 4.8.3 Riservatezza dell'Access Token e del Grant Token

Access e Grant Token devono essere mantenuti nella memoria locale volatile del dispositivo (o di sessione del web server) in modo tale da impedirne a priori l'esportabilità dal dispositivo.

Access e Grant Token devono essere rimossi dalla memoria locale volatile non appena cessino di essere utili per il processo applicativo.

Nel caso specifico dello scenario [operatori.siss] il Refresh non deve essere conservato dal web server ma distrutto al termine della sessione utente.

Esempi di cessazione dell'utilità di tali token sono:

- Chiusura dell'applicazione;
- Inattività prolungata (specialmente quando posta in background) dell'APP.

## 4.9 *Appendice 1 – Fault generati dall’API manager e codici d’errore*

Quando alle API rest è richiesto di eseguire le varie azioni i vari HTTP status codes sono restituiti sulla base dello stato dell’azione (successo o fallimento) ed l’ HTTP method (POST, GET, DELETE) eseguito. Nel seguito sono elencati I codici restituiti nei due casi.

### Success HTTP status codes

Code	Code Summary	Description
100	Continue	The server has received the request headers, and the client should proceed to send the request body. This is applicable in the case of a request for which a body needs to be sent (e.g., POST request). This interim response is used to inform the client that the initial part of the request has been received and has not yet been rejected by the server. The server will send a final response after the request has been completed.
200	Ok	HTTP request was successful. The output corresponding to the HTTP request will be returned. Generally used as a response to a successful GET REST API HTTP method.
201	Created	HTTP request was successfully processed and a new resource was created. Generally used as a response to a successful POST REST API HTTP method.
202	Accepted	HTTP request was accepted for processing, but the processing has not been completed.

### Error HTTP status codes

Code	Code Summary	Description
400	Bad request	The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid JSON format).
401	Unauthorized	When authentication is required and has failed, or when authentication has not yet been provided.
406	Not Acceptable	The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request. The default accept header is application/json. If you do not specify an accept header, the default accept header is applied. However, if you specify an accept header that is not supported, this status code is displayed.
415	Unsupported Media Type	The request entity has a media type that the server or resource does not support. The following are the supported content types: application/json, application/x-www-form-urlencoded and multipart/form-data. If you send a content type that differs to the content type supported, this status code is displayed.
500	Internal server error	Server error occurred.
501	Not implemented	When the server either does not recognize the request method, or it lacks the ability to fulfill the request.



## 4.9.1 Fault e codici d'errore

Nel seguito è riportata la generica struttura del documento XML di Fault prodotto dall'API Manager quando restituisce *http status codes* nel range 40X.

```
<?xml version="1.0"?>
<ams:fault xmlns:ams="http://wso2.org/apimanager/security">
  <ams:code> </ams:code>
  <ams:message> </ams:message>
  <ams:description> </ams:description>
</ams:fault>
```

### Ad esempio

```
403 Forbidden
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0"?>
<ams:fault xmlns:ams="http://wso2.org/apimanager/security">
  <ams:code>900908</ams:code>
  <ams:message>Resource forbidden</ams:message>
  <ams:description>Access failure for API: /t/cittadini.rl/calc/multiply/1.0, version:
1.0</ams:description>
</ams:fault>
```

Code	Message	Description
900900	Unclassified Authentication Failure	An unspecified error has occurred
900901	Invalid Credentials	Invalid Authentication information provided
900902	Missing Credentials	No authentication information provided
900905	Incorrect Access Token Type is provided	The access token type used is not supported when invoking the API. The supported access token types are application and user access tokens. See <a href="#">Access Tokens</a> .
900906	No matching resource found in the API for the given request	A resource with the name in the request can not be found in the API.
900907	The requested API is temporarily blocked	The status of the API has been changed to an inaccessible/unavailable state.
900908	Resource forbidden	The user invoking the API has not been granted access to the required resource.
900909	The subscription to the API is inactive	Happens when the API user is blocked.
900910	The access token does not allow you to access the requested resource	Can not access the required resource with the provided access token. Check the valid resources that can be accessed with this token.
900800	Message throttled out	The maximum number of requests that can be made to the API within a designated time period is reached and the API is throttled for the user.





Code	Message	Description
700700	API blocked	This API has been blocked temporarily. Please try again later or contact the system administrators.
800800	Unclassified Authentication Failure	Missing grant token
800801	Unclassified Authentication Failure	Invalid grant token
800802	Unclassified Authentication Failure	Unclassified Grant Token Failure

---

#### **4.10 Appendice 2 – Esempio di albero delle interazioni di UN'APP MOBILE**

Il seguente schema fornisce un'interpretazione delle modalità di organizzazione del flusso funzionale di un'APP per dispositivi mobili inerente i soli aspetti di fruizione delle API.

Si noti come, tramite un'oculata gestione di alcune delle segnalazioni d'errore restituite da Gateway, sia possibile gestire il processo di rinnovo ed utilizzo dei token minimizzando il coinvolgimento dell'utente dell'APP.

La gestione multi-dispositivo per singolo utente non è trattata nello schema, si faccia riferimento al contenuto di questo documento gestendo in modo appropriato lo *scope device\_*.

