Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Smart City Platform Specification
Communication Level 1.0

C. Novelli, A. Frascella, G. Santomauro

Report RdS/PAR2017/108

SMART CITY PLATFORM SPECIFICATION COMMUNICATION LEVEL 1.0

C.Novelli, A.Frascella, G.Santomauro (ENEA)

Settembre 2018

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2017
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici
Progetto: Sviluppo di un modello integrato di smart district urbano
Obiettivo: Piattaforma ICT per la gestione dello Smart District
Responsabile del Progetto: Claudia Meloni, ENEA

Indice

Introduzione

Questa specifica tecnica è parte delle **Smart City Platform Specification (SCPS)**
http://smartcityplatform.enea.it/specification/

Questo documento è la specifica **Smart City Platform Specification (SCPS) Communication Level**
http://smartcityplatform.enea.it/specification/communication/

Communication Level

Le Specifiche del Communication Level (o “Livello Comunicazione”) descrivono il protocollo di Trasporto per lo scambio dati che abilita la comunicazione tra Smart City Platform (SCP) orizzontale e una o più Solution verticali (si veda cap.1, “Functional Level”) allo scopo di raggiungere due principali obiettivi:

1. permettere alla Smart City Platform di recuperare i dati da ogni Solution;
2. abilitare la comunicazione tra Solution verticali (inter-solution communication).

Prima di descrivere il protocollo di trasporto definito, riprendiamo alcuni concetti espressi nella descrizione dell’Architettura di Riferimento (livello Functional, capitolo 1):

- ogni Solution verticale è una piattaforma software locale a un Contesto Applicativo specifico della Smart City, e si occuperà della comunicazione con la Smart City Platform;
- la comunicazione è abilitata presso la SCP tramite una serie di informazioni concordate e immagazzinate nel Registry centrale (livello Collaboration, capitolo 2);
- ogni Solution è producer e/o consumer di set di dati, denominati Urban Dataset;
- gli Urban Dataset hanno una definizione semantica e sintattica centralizzata (livello Semantic, capitolo 3) e sono rappresentati nel formato comune (livello Information, capitolo 4).

Il protocollo di trasporto delle specifiche SCPS si può riassumere in due azioni principali che una Solution verticale può decidere di intraprendere per comunicare con la Smart City Platform:

1. Produzione di Urban Dataset: esportazione, da parte di una Solution verticale (p.es. Smart Building platform), di dati che saranno recuperati dalla Smart City Platform;
2. Accesso a Urban Dataset: accesso ai dati che una Solution verticale (p.es. WebGIS platform) attua presso la Smart City Platform che li ha precedentemente pubblicati.

Se la prima azione abilita il recupero di dati da Solution a Smart City Platform, e la seconda azione abilita l’accesso ai dati da Smart City Platform a Solution, l’insieme coordinato delle due azioni permette lo scambio di dati da una Solution verticale a un’altra tramite la Smart City Platform.

La comunicazione può quindi avvenire in diversi modi, dipendentemente dal pattern di scambio dati scelto, dal protocollo di trasporto, dal web service implementato: la SCPS Communication Level descrive un insieme di possibili configurazioni per attuare tale comunicazione.

Nei prossimi paragrafi verrà descritto il protocollo di trasporto che permette di abilitare questa comunicazione, in termini di:

- Pattern;
- Interfaccia Web Service.

1 Pattern

Nei seguenti paragrafi vengono presentati alcuni pattern architetturali che sono delineati nelle specifiche “Smart City Platform Specification - Communication Level” e che quindi devono essere implementati da una Smart City Platform che aderisce a tali specifiche.

Ricordiamo che nell'ambito dell'ingegneria del software:

- Un “design pattern” può essere definito come "una soluzione progettuale generale a un problema ricorrente". Si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software, ancor prima della definizione dell'algoritmo risolutivo della parte computazionale.
- Un “pattern architetturale” esprime uno schema per l’organizzazione strutturale di sistemi software.

Dunque, i design pattern sono orientati agli oggetti e mostrano relazioni e interazioni fra classi o oggetti; i pattern architetturali, invece, si pongono a un livello più alto e descrivono un pattern complessivo adottato dall’intero sistema.

In questo paragrafo si presenteranno 2 pattern architetturali:

1. Request/Response;
2. Push;

che potranno essere utilizzati per implementare la comunicazione tra Solution Verticali e Smart City Platform, dipendentemente dalle necessità insite nel caso da gestire.

1.1 Pattern REQUEST/RESPONSE

"REQUEST/RESPONSE" è uno dei pattern più classici, è lo stesso che è usato, ad esempio, nella navigazione delle pagine web: un Client digita un URL e in risposta ottiene la pagina web desiderata dal Server.

Nelle chiamate Web Service su HTTP, il pattern REQUEST/RESPONSE viene implementato in modo sincrono: una volta effettuata la richiesta dal componente Client, si mantiene una connessione aperta fino a quando la risposta del componente Server viene consegnata, oppure scade un timer (dunque è bloccante).

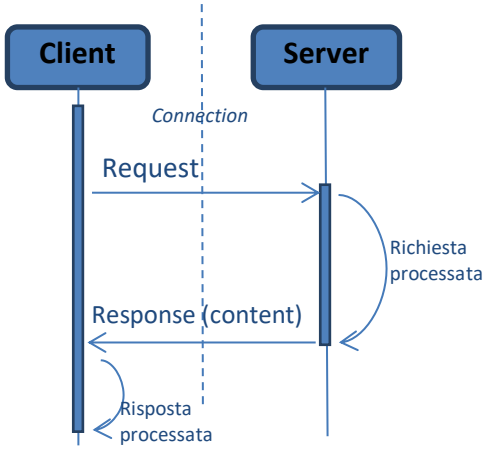

Figura 1 - Interazione Client-Server nel pattern Request/Response

Adottando il pattern REQUEST/RESPONSE per abilitare la comunicazione basata su SCPS, si intende la seguente implementazione di servizio:

1- Il componente Client effettua una REQUEST per RICHIEDERE un determinato Urban Dataset;

2- Il componente Server restituisce la RESPONSE contenente l'Urban Dataset richiesto.

Si noti che, dipendentemente dalla situazione, Client e Server possono essere ruoli interpretati da una Solution Verticale e dalla Smart City Platform e viceversa.

1.2 Pattern PUSH

Il pattern "PUSH" è molto al pattern "request/response" ma il Client, invece di richiedere qualcosa, invia nella sua chiamata un contenuto che vuole sia ricevuto dal Server, che potrà rispondere con un acknowledgment (si veda in proposito il paragrafo **Errore. L'origine riferimento non è stata trovata.** "Consegna affidabile").

Nelle chiamate Web Service su HTTP, il pattern PUSH viene implementato tipicamente in modo sincrono: una volta effettuata la chiamata dal componente Client, si mantiene una connessione aperta fino a quando l'ack del componente Server viene consegnato, oppure scade un timer (dunque è bloccante).

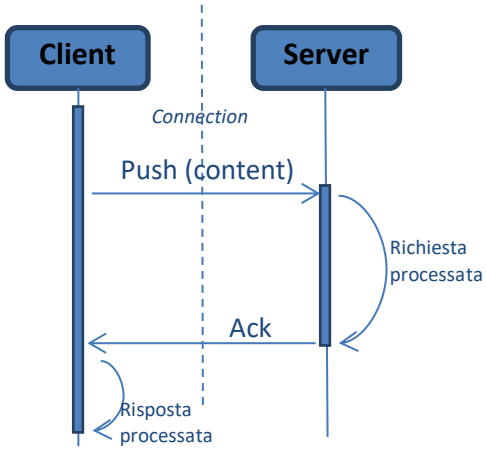

Figura 2 - Interazione Client-Server nel pattern Push

Adottando il pattern PUSH per abilitare la comunicazione basata su SCPS, si intende la seguente implementazione di servizio:

1- Il componente Client effettua una chiamata PUSH per INVIARE un determinato UrbanDataset;

2- Il componente Server riceve l'UrbanDataset, lo processa e risponde con un Ack.

Si noti che, dipendentemente dalla situazione, Client e Server possono essere ruoli interpretati da una Solution Verticale e dalla Smart City Platform e viceversa.

2 Web Service Interface

In questo paragrafo viene descritta l'interfaccia del web service *UrbanDatasetGateway*; l'implementazione di questo web service è necessaria per aderire alle specifiche SCPS Communication Level.

Il web service *UrbanDatasetGateway* è sviluppato su protocollo HTTP/HTTPS con architettura REST (REpresentational State Transfer), e implementa i pattern architetturali presentati nel par. 0:

- REQUEST/RESPONSE: per consentire a un client di richiedere un Urban Dataset a un server in ascolto (p.es. la Smart City Platform, come client, richiede un Urban Dataset alla Solution verticale che espone il servizio, come server, il quale risponde con l'Urban Dataset richiesto);
- PUSH: per consentire a un client di inviare direttamente un Urban Dataset a un destinatario in ascolto (p.es. una Solution verticale, come client, invia un Urban Dataset alla Smart City Platform che espone il servizio, come server, e riceve l'Urban Dataset).

Viene definito un singolo servizio web che, implementato sia lato Smart City Platform, sia lato Solution verticale, permette la comunicazione in entrambe le direzioni secondo i pattern PUSH e REQUEST/RESPONSE.

La seguente tabella descrive l'interfaccia del Web Service a prescindere dall'implementazione.

Servizio	UrbanDatasetGateway	
Metodi	test()	permette a un client di testare la presenza del web service
	login (username, password)	permette a un client di autenticarsi presso il servizio che espone questo metodo tramite username e password e ricevendo un token JWT (stringa) che utilizzerà nelle successive chiamate
	logout ( )	permette a un client di annullare l'autenticazione presso il servizio che espone questo metodo rendendo invalido il token ricevuto nella precedente chiamata di login
	isAlive ( )	permette di verificare che il token sia ancora valido
	push (dataset)	permette di inviare un Urban Dataset tramite una singola chiamata PUSH
	basicRequest (resource_id)	permette di richiedere un Urban Dataset tramite una singola chiamata REQUEST/RESPONSE senza raffinamento della ricerca.
	specificRequest (resource_id, timestamp)	permette di richiedere un particolare Urban Dataset generato a uno specifico Timestamp, tramite una singola chiamata REQUEST/RESPONSE
	delete (resource_id, timestamp)	permette di eliminare uno specifico Urban Dataset
	searchingRequest (resource_id, period_start*, period_end*, center_latitude*, center_longitude, radius*)	permette di richiedere un Urban Dataset tramite una singola chiamata REQUEST/RESPONSE con raffinamento spazio-temporale della ricerca a livello di context (elemento di contestualizzazione UD presente nel formato)
	deepSearchingRequest (resource_id, period_start*, period_end*, center_latitude*, center_longitude, radius*)	permette di richiedere un Urban Dataset tramite una singola chiamata REQUEST/RESPONSE con raffinamento spazio-temporale della ricerca a livello di linea (elementi di specificazione dei record di valori del dataset, presenti nel formato)

*valori opzionali

N.B. il metodo searchingRequest con tutti i parametri opzionali mancanti, richiama la basicRequest

Per gestire i possibili errori che possono verificarsi, vengono indicati alcuni codici relativi alla comunicazione basata su specifiche SCPS che devono essere contenuti nella risposta.

Si veda l'appendice A per una descrizione completa di tutti i codici.

3 REST

Il formato del servizio REST, rispettante l'interfaccia descritta nel precedente paragrafo) è il seguente:

[URL_BASE]/UrbanDatasetGateway/[METODO]

Dove:

- *[URL_BASE]* : è l'URL che identifica dove il servizio è stato pubblicato,
p.es. *https://[IP]:[PORTA]/webservices/rest/*
dove *[IP]* è la macchina su cui il servizio è pubblicato e
dove *[PORTA]* è la porta aperta su cui il servizio è in ascolto
e l'insieme [IP]:[PORTA] può essere mappato con un nome di dominio;

- *UrbanDatasetGateway*: è il nome del web service in oggetto,
p.es. `https://[IP]:[PORTA]/webservices/rest/UrbanDatasetGateway/`

- *[METODO]* : è uno dei metodi esposti dal servizio,
p.es. `https://[IP]:[PORTA]/webservices/rest/UrbanDatasetGateway/push`

Le chiamate al web service sono prevalentemente di tipo POST, e forniscono la possibilità di inserire una serie di parametri corrispondenti all'input per i metodi del servizio.

Seguono le descrizioni dei metodi per il web service implementato con tecnologia REST
(descrizioni rigorosamente fedeli all'interfaccia fornita nel capitolo 0).

3.1 test REST method

Il metodo “test” permette a un client di testare la presenza del web service RESTful.

E’ l’unico metodo del servizio richiamabile tramite una GET.

General	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/test
Tipo Chiamata	GET/POST
Header	
	vuoto
Body	
	vuoto
Return	
Success	{ “code”: “00”, “message”: “Successful” }

Il valore di ritorno riporta il codice scps di ritorno (si veda Appendice A), il relativo messaggio.

Si faccia sempre riferimento alla descrizione astratta dell’interfaccia del Web Service (nel capitolo precedente) per la lista dei codici “code”/”message” ammissibili per il metodo in oggetto.

3.2 login REST method

Il metodo “login” permette a un client di autenticarsi presso il servizio che espone questo metodo tramite username e password e ricevendo un token (stringa) che utilizzerà nelle successive chiamate.

General	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/login
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Body	
Type	URLencoded form data
Parametri	username=[string/string]
	password=[string/string]

Esempio	{
	"username": "myusername",
	"password": "mypassword"
	}

Return	
Success	{ “code”: “01”, “message”: “Authentication Successful”, “token”: “[JWT-TOKEN]”] } Esempio di [JWT-TOKEN] “token”:“eyJraWQiOiIzMUUzRDZaM0xaMVdFSEJGWVRQRksxRzY4IiwiYWxnIjoiSFMyNTYifQ.eyJqdGkiOiI2a3NjVFMyUjZuYlU3c1RhZ0h0aWFXIiwiaWF0IjoxNDQ1ODU0Njk0LCJpc3MiOiJodHRwczovL2FwaS5zdG9ybXBhdGguY29tL3YxL2FwcGxpY2F0aW9ucy8zUUlNbEpLS04yd2hHQ1l6WFh3MXQ4Iiwic3ViIjoiaHR0cHM6Ly9hcGkuc3Rvcm1wYXRoLmNvbS92MS9hY2NvdW50cy8xeG15U0dLMXB5VVc1c25qOENvcmU1IiwiZXhwIjoxNDQ1ODU4Mjk0LCJydGkiOiI2a3NjVE9pTUNESVZWM05qVTIyUnlTIn0.VJyMOicMOdcOCtytsx4hoPHy3Hl3AfGNfi2ydy8AmG4” }
Failure	{ “code”: “11”, “message”: “Authentication Failure”, “token”: “” }
Esempio di Success	{ “code”: “01”, “message”: “Authentication Successful”, “token”: “eyJraWQiOiIzMUUzRDZaM0xaMVdFSEJGWVRQRksxRzY4IiwiYWxnIjoiSFMyNTYifQ.eyJqdGkiOiI2a3NjVFMyUjZuYlU3c1RhZ0h0aWFXIiwiaWF0IjoxNDQ1ODU0Njk0LCJpc3MiOiJodHRwczovL2FwaS5zdG9ybXBhdGguY29tL3YxL2FwcGxpY2F0aW9ucy8zUUlNbEpLS04yd2hHQ1l6WFh3MXQ4Iiwic3ViIjoiaHR0cHM6Ly9hcGkuc3Rvcm1wYXRoLmNvbS92MS9hY2NvdW50cy8xeG15U0dLMXB5VVc1c25qOENvcmU1IiwiZXhwIjoxNDQ1ODU4Mjk0LCJydGkiOiI2a3NjVE9pTUNESVZWM05qVTIyUnlTIn0.VJyMOicMOdcOCtytsx4hoPHy3Hl3AfGNfi2ydy8AmG4” }

Internamente, il componente server che espone il servizio e riceve la chiamata al metodo “login”, effettua il servizio di autenticazione per il client chiamante; può essere chiamata una terza parte (p.es. un server di autenticazione dedicato).

Il valore di ritorno riporta il codice scps di ritorno (si veda Appendice A), il relativo messaggio e, in caso di successo, il JSON Web Token, altrimenti una stringa vuota “”.

3.3 IsAlive

Metodo per verificare che il token sia ancora valido.

In caso sia invalido è necessario far ripetere all'utente la procedura di login.

General	
Servizio	[URL_BASE]/UrbanDatasetGateway/isAlive
Tipo Chiamata	POST
Header	
Content-type	application-json
Accept	application-json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login
Body	
Return	
Success	{ "code": "00", "message": "Successful", "username": "[USERNAME]" } dove [USERNAME] è lo username associato al token
Failure	{ "code": "12", "message" : "Invalid Token" }

Questo metodo deve essere chiamato prima di ogni altro metodo.

3.4 logout REST method

Il metodo “logout” permette a un client di annullare l’autenticazione presso il servizio che espone questo metodo rendendo invalido il token ricevuto nella precedente chiamata di login.

General	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/logout
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l’accesso ai dati, rilasciato dal metodo di login
Body	
Return	
Success	{ “code”: “09”, “message”: “Logout Successful” }

Failure	{
	“code”: “[CODE]”,
	“message”: “[MSG]”
	}
	Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Internamente, il componente server che espone il servizio e riceve la chiamata al metodo “logout”, effettua il servizio di logout per il client chiamante; può essere chiamata una terza parte (p.es. un server di autenticazione dedicato).

3.5 *push REST method*

Il metodo “push” permette di inviare un Urban Dataset tramite una singola chiamata PUSH.

General	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/push
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l’accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un Urban Dataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
	dataset=[string/dataset] la rappresentazione dell’Urban Dataset secondo specifiche SCPS in formato JSON

Esempio	{
	"resource_id": "SCP-1_SmartBuildingCasaccia-3_SmartBuildingAnomalies-1.0_20180125120000",
	"dataset": { "UrbanDataset": { "specification": { .... } … } …}
	}

Return	

Success	{
	“code”: “02”,
	“message”: “Push Successful”
	}

Failure	{
	“code”: “[CODE]”,
	“message”: “[MSG]”
	}
	
	Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Nel metodo “push”:
- si verifica il token e si recupera il ruolo per quello user;
- verifica la “buona forma” del resource_id (ovvero la coerenza delle 4 parti);
- se lo user ha ruolo “solution”, la solution è abilitata, la produzione è abilitata, l’UD è valido, allora l’UD è accettato e memorizzato nel database;
- la tabella *history_production* viene aggiornata (in particolare *history_state_id=1, STORED*);
- se l’UD è in OVERWRITE, il precedente viene cancellato (e aggiornato *history_state_id=2, OVERWRITTEN*);
- viene ritornato un codice di ritorno (si veda Appendice A) con relativo messaggio.

Esempio: la Solution “Smart Building Casaccia” invoca il metodo “push” per inviare l’UrbanDataset “Building Anomalies” alla Smart City Platform “Smart Village Casaccia”.

3.6 specificRequest REST method

Il metodo “specificRequest” permette di richiedere un particolare Urban Dataset generato a uno specifico Timestamp, tramite una singola chiamata REQUEST/RESPONSE.

Generale	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/specificRequest
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l’accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un Urban Dataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
	timestamp=[string/datetime] Identifica il timestamp di generazione dell’UrbanDataset, utilizzato dal produttore di quel dato in fase di produzione.
Esempio	{ "resource_id": "SCP-1_SmartBuildingCasaccia-3_SmartBuildingAnomalies-1.0_20180125120000", “timestamp”: “2018-02-28T16:10:00” }
Return	
Success	{ “code”: “03”, “message”: “Request-Response Successful”, “dataset”: { "UrbanDataset": { … } } } Dove { "UrbanDataset": { … } } è la rappresentazione JSON di un singolo Urban Dataset secondo le specifiche SCPS Information Level.
Failure	{ “code”: “[CODE]”, “message”: “[MSG]” } Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Nel metodo “specificRequest”:
- si verifica il token e si recupera il ruolo per quello user;
- si verifica la “buona forma” del resource_id (ovvero la coerenza delle 4 parti);
- se l’UD è OPENDATA, l’UD viene ritornato;
- se lo user ha ruolo “admin”, l’UD viene ritornato;
- se lo user ha ruolo “solution”, la solution è abilitata e l’accesso è abilitato, l’UD viene ritornato.
- la tabella *history_access* viene aggiornata;
- viene ritornato un codice di ritorno (si veda Appendice A) con relativo messaggio.

Esempio: l’amministratore della Smart City Platform “Smart Village Casaccia”, vuole vedere nella GUI l’UrbanDataset “Building Anomalies” ricevuto l’ultimo giorno, di cui ha il timestamp (si veda la tabella *history_production*).

3.7 basicRequest REST method

Il metodo “basicRequest” permette di richiedere un Urban Dataset tramite una singola chiamata REQUEST/RESPONSE senza raffinamento della ricerca.

Generale	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/basicRequest
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l’accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un Urban Dataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
Esempio	{ "resource_id": "SCP-1_SmartBuildingCasaccia-3_SmartBuildingAnomalies-1.0_20180125120000" }
Return	

Success	{
	“code”: “03”,
	“message”: “Request-Response Successful”,
	“dataset”: [{ "UrbanDataset": { … } }, { … }]
	}
	Dove [{ "UrbanDataset": { … } }, { … }] è la rappresentazione JSON di uno o più Urban Dataset secondo le specifiche SCPS Information Level.

Failure	{
	“code”: “[CODE]”,
	“message”: “[MSG]”,
	“dataset”: [ ]
	}
	Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Nel metodo “basicRequest”:
- si verifica il token e si recupera il ruolo per quello user;
- si verifica la “buona forma” del resource_id (ovvero la coerenza delle 4 parti);
- se l’UD è OPENDATA, l’UD viene ritornato qualunque sia l’utente;
- se lo user ha ruolo “admin”, l’UD viene ritornato;
- se lo user ha ruolo “solution”, la solution è abilitata e l’accesso è abilitato, l’UD viene ritornato;
- la tabella *history_access* viene aggiornata;
- viene ritornato un codice di ritorno (si veda Appendice A) con relativo messaggio.

Esempio: la Solution “WebGIS Casaccia” invoca il metodo “basicRequest” per recuperare l’UrbanDataset “Building Anomalies” dalla Smart City Platform “Smart Village Casaccia”.

3.8 delete REST method

Il metodo “delete” permette di eliminare uno specifico Urban Dataset.

Generale	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/delete
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l’accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un Urban Dataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
	timestamp=[string/datetime] Identifica il timestamp di generazione dell’UrbanDataset, utilizzato dal produttore di quel dato in fase di produzione e utilizzato ora per la sua cancellazione.
Esempio	{ "resource_id": "SCP-1_SmartBuildingCasaccia-3_SmartBuildingAnomalies-1.0_20180125120000", “timestamp”: “2018-02-28T16:10:00” }
Return	
Success	{ “code”: “08”, “message”: “Delete Successful” }
Failure	{ “code”: “[CODE]”, “message”: “[MSG]” } Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Solo un utente Administrator o l’utente Solution producer dell’UrbanDataset possono effettuare la cancellazione. In caso contrario deve essere ritornato

code: "28", message: "No Permit to delete this UrbanDataset".

Nel momento in cui il dataset è cancellato fisicamente, vengono aggiornati anche i relativi storici

- la tabella *history_production* viene aggiornata (in particolare *history_state_id=3, DELETED*).

N.B. gli UD configurati in modalità OVERWRITE, non possono essere cancellati, basta sovrascriverli.

3.9 searchingRequest REST method

Il metodo “searchingRequest” permette di richiedere un Urban Dataset tramite una singola chiamata REQUEST/RESPONSE con raffinamento spazio-temporale della ricerca a livello di context (elemento di contestualizzazione UD presente nel formato).

Generale	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/searchingRequest
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l’accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un Urban Dataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
	period_start*=[DATETIME] Data e ora da cui si vuole specificare la partenza di un intervallo temporale. Nel caso sia assente, la finestra temporale include tutti i valori possibili.
	period_end*=[DATETIME] Data e ora da cui si vuole specificare la fine di un intervallo temporale. Nel caso sia assente, la finestra temporale include tutti i valori possibili da *period_start* fino al momento in cui viene effettuata la chiamata.
	center_latitude*=[COORDINATE_WGS84] Latitudine del centro su cui si effettuerà la ricerca spaziale.
	center_longitude*=[COORDINATE_WGS84] Longitudine del centro su cui si effettuerà la ricerca spaziale.
	distance*=[DISTANCE_WGS84] Raggio del cerchio su cui si effettuerà la ricerca spaziale.
	* parametri opzionali
Return	

Success	{
	“code”: “03”,
	“message”: “Request-Response Successful”,
	“dataset”: [{ "UrbanDataset": { … } }, { … }]
	}
	Dove [{ "UrbanDataset": { … } }, { … }] è un array di uno o più Urban Dataset JSON secondo le specifiche SCPS Information Level.

Failure	{ “code”: “[CODE]”,

	"message": "[MSG]", "dataset": [ ] } Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Il metodo "searchingRequest" senza i parametri di ricerca *opzionali, corrisponde al "basicRequest".

Nel metodo "searchingRequest":
- si verifica il token e si recupera il ruolo per quello user;
- si verifica la "buona forma" del resource_id (ovvero la coerenza delle 4 parti);
- se l'UD è OPENDATA, l'UD viene ritornato;
- se lo user ha ruolo "admin", l'UD viene ritornato;
- se lo user ha ruolo "solution", la solution è abilitata e l'accesso è abilitato, l'UD viene ritornato;
- si effettua la ricerca a livello di CONTEXT (elemento *context* dell'UD) tenendo conto della finestra spazio-temporale indicata nella richiesta;
- la tabella *history_access* viene aggiornata;
- viene ritornato un codice di ritorno (si veda Appendice A) con relativo messaggio.

Esempio: la Smart City Platform "Smart Village Casaccia", espone un servizio GUI sottoforma di tabella che, tramite la chiamata a *searchingRequest*, permette di vedere tutti gli UrbanDataset prodotti nell'ultimo giorno (o mese, o settimana) facenti riferimento alla sola area geografica della Casaccia.

3.10 deepSearchingRequest REST method

Il metodo "deepSearchingRequest" permette di richiedere un Urban Dataset tramite una singola chiamata REQUEST/RESPONSE con raffinamento spazio-temporale della ricerca a livello di linea (elementi di specificazione dei record di valori del dataset, presenti nel formato).

Generale	
Servizio	[URL_BASE]/UrbanDatasetGateway
Metodo	[URL_BASE]/UrbanDatasetGateway/deepSearchingRequest
Tipo Chiamata	POST
Header	
Content-Type	application/json
Accept	application/json
Authorization	Bearer [JWT-TOKEN] Dove [JWT-TOKEN] = token di autenticazione per l'accesso ai dati, rilasciato dal metodo di login
Body	
Parametri	resource_id=[string/resource_id] identifica univocamente un Urban Dataset prodotto da una specifica Solution producer (segue la sintassi definita nelle SCPS Core (Appendice B.3)
	period_start*=[DATETIME] Data e ora da cui si vuole specificare la partenza di un intervallo temporale. Nel caso sia assente, la finestra temporale include tutti i valori possibili.
	period_end*=[DATETIME]

	Data e ora da cui si vuole specificare la fine di un intervallo temporale. Nel caso sia assente, la finestra temporale include tutti i valori possibili da *period_start* fino al momento in cui viene effettuata la chiamata.
	center_latitude*=[COORDINATE_WGS84] Latitudine del centro su cui si effettuerà la ricerca spaziale.
	center_longitude*=[COORDINATE_WGS84] Longitudine del centro su cui si effettuerà la ricerca spaziale.
	distance*=[DISTANCE_WGS84] Raggio del cerchio su cui si effettuerà la ricerca spaziale.
	* parametri opzionali
Return	

Success	{
	“code”: “03”,
	“message”: “Request-Response Successful”,
	“dataset”: [{ "UrbanDataset": { … } }, { … }]
	}
	Dove [{ "UrbanDataset": { … } }, { … }] è un array di uno o più Urban Dataset JSON secondo le specifiche SCPS Information Level.

Failure	{ “code”: “[CODE]”, “message”: “[MSG]”, “dataset”: [ ] } Dove [CODE] e [MSG] assumono specifici valori dipendentemente dal tipo di errore riscontrato (si veda la lista completa dei codici di ritorno in Appendice A)

Nel metodo “deepSearchingRequest”:
- si verifica il token e si recupera il ruolo per quello user;
- si verifica la “buona forma” del resource_id (ovvero la coerenza delle 4 parti);
- se l’UD è OPENDATA, l’UD viene ritornato;
- se lo user ha ruolo “admin”, l’UD viene ritornato;
- se lo user ha ruolo “solution”, la solution è abilitata e l’accesso è abilitato, l’UD viene ritornato;
- si effettua la ricerca a livello di LINEA (elemento *values* dell’UD), andando a selezionare le linee i cui periodi rientrano completamente nell’intervallo temporale e spaziale dati;
- la tabella *history_access* viene aggiornata;
- viene ritornato un codice di ritorno (si veda Appendice A) con relativo messaggio.

Esempio1: la Smart City Platform “Smart Village Casaccia”, espone un servizio GUI di georeferenziazione (mappa) che, tramite la chiamata a *deepSearchingRequest*, permette di vedere tutti i DATI relativi a una zona specifica (p.es. un palazzo) interna all’area della Casaccia.

Esempio2: la Smart City Platform “Smart Village Casaccia”, espone un servizio GUI sottoforma di tabella che, tramite la chiamata a *deepSearchingRequest*, permette di vedere tutti i DATI relativi a una fascia temporale specifica (per esempio di un particolare giorno, in una particolare fascia oraria.

N.B. la ricerca spaziale viene effettuata in termini assoluti, confrontando la distanza tra i due punti forniti in sistema WGS84 (distanza ricavata con il teorema di pitagora, tra il punto fornito in input e il punto della linea valutata) con il raggio. La conversione in metri o chilometri è delegata al sistema chiamante.

4 SOAP

[IMPORTANTE] LO SVILUPPO DEL WS SOAP E' MOMENTANEAMENTE SOSPESO, IN ATTESA DELLA SPERIMENTAZIONE FINALE CON WS REST.

Il formato del servizio SOAP (rispettante l'interfaccia descritta nel cap.2) è il seguente:

[URL_BASE]/UrbanDatasetGateway/[METODO]

Dove:

- *[URL_BASE]* : è l'URL che identifica dove il servizio è stato pubblicato,

p.es. *http://[IP]:[PORTA]/webservices/soap/*

dove *[IP]* è la macchina su cui il servizio è pubblicato e

dove *[PORTA]* è la porta aperta su cui il servizio è in ascolto

e l'insieme [IP]:[PORTA] può essere mappato con un nome di dominio;

- *UrbanDatasetGateway*: è il nome del web service in oggetto,

p.es. `http://[IP]:[PORTA]/webservices/soap/UrbanDatasetGateway/`

Il web service SOAP è descritto nel WSDL (Web Service Description Language) la cui ultima versione è disponibile a questo URL
http://smartcityplatform.enea.it/specification/communication/1.0/wsdl/

[IMPORTANTE] LO SVILUPPO DEL WS SOAP E' MOMENTANEAMENTE SOSPESO, IN ATTESA DELLA SPERIMENTAZIONE FINALE CON WS REST.

Appendice A - Codici di Ritorno

Le chiamate ai diversi metodi del web service "*UrbanDatasetGateway*" devono gestire i diversi casi di errore che possono verificarsi, oltre ai comuni già noti supportati dal protocollo http.

Tutti i casi di errore sono stati individuati e catalogati nella seguente lista che fornisce, per le diverse categorie, i codici errori e i relativi messaggi, in modo tale che il client possa sempre riconoscere e gestire la risposta del web service.

Categoria "0X" - Success

- code: "00", message:"Successful";
- code: "01", message:"Authentication Successful";
- code: "02", message:"Push Successful";
- code: "03", message: "Request-Response Successful";
- code: "08", message: "Delete Successful";
- code: "09", message: "Logout Successful";

Categoria "1X" - Failure Communication

- code: "10", message:"Failure";
- code: "11", message:"Authentication Failure";
- code: "12", message:"Invalid Token";
- code: "13", message:"Bad Request";
- code: "14", message:"System Error";

Categoria "2X" - Failure Collaboration

- code: "20", message: "Unknow Resource";
- code: "21", message: "No Permit to produce this UrbanDataset";
- code: "22", message: "No Permit to access to this UrbanDataset";
- code: "23", message: "UrbanDataset Inconsistency Error";
(p.es. timestamp di generazione UD vecchio, rispetto a uno già inviato precedentemente)
- code: "24", message: "Collaboration Not Respected";
- - detail: "UrbanDataset sending wrong frequency"
- code: "28", message: "No Permit to delete this UrbanDataset";

Categoria "3X" - Failure Information

- code: "30", message: "Unknow UrbanDataset";
- code: "31", message: "Invalid UrbanDataset against JSON Schema";
- code: "32", message: "Invalid UrbanDataset against XML Schema";
- code: "33", message: "Invalid UrbanDataset against Schematron";
- code: "34", message: "UrbanDataset not found";
- code: "35", message: "UrbanDataset file size too big".

Appendice B - Metodi opzionali

Il web service *UrbanDatasetGateway* presenta diversi metodi la cui implementazione è necessaria per aderire alle specifiche SCPS Communication Level.

Si faccia sempre riferimento alla descrizione dell'interfaccia del Web Service nel capitolo 0.

In questo paragrafo si descriverà un ulteriore metodo "opzionale" per il web service definito.

Explore method interface

Il metodo "explore" permette di esplorare un Urban Dataset articolato definito su più livelli logici, tramite una serie di chiamate i cui metodi sono essi stessi parametrizzabili).

Viene utilizzato solitamente per navigare la struttura di un Urban Dataset statico.

Dopo la tabella verrà fornito un esempio.

	Metodo	explore
	Parametro	token tipo: string/auth-token descrizione: il token è una stringa di autenticazione (minimo 20 caratteri), viene rilasciato ad ogni client che effettua la login con successo e ha una validità temporale
	Parametro	resource_id tipo: string/resource_id descrizione: identifica univocamente un Urban Dataset prodotto da una specifica Solution producer nell'ambito di una specifica collaborazione, in una specifica Smart City Platform. Se il web service è pubblicato sulla Smart City Platform, il valore del *resource_id* può essere recuperato da una Solution registrata che ha accesso a tale UrbanDataset. Se il web service è pubblicato sulla Solution verticale, il valore del *resource_id* coincide con l'identificatore dell'UrbanDataset.
	Parametro	requestResource=[string/string] nome dello specifica risorsa richiesta (ovvero che il client sta richiedendo al server) N.B. si può richiedere solo una risorsa per volta
	Parametro	inputParameter=[string/string] nome del parametro fornito in input N.B. si possono fornire in input N coppie inputParameter, inputValue
	Parametro	inputValue=[string/string] valore dello parametro fornito in input N.B. si possono fornire in input N coppie inputParameter, inputValue
	Valore di Ritorno	code tipo: string/string (si veda Appendice A)
	Valore di Ritorno	message tipo: string/string (si veda Appendice A)

Explore method example

Si ipotizzi di avere un Urban Dataset di tipo statico: “Rifiuti Smart”.

Il “contenitore” può essere di due tipi: “cestino”, “cassonetto”.

Ogni cestino è associato a un “utente”.

Ogni cassonetto è associato a una “via” e riconosciuto tramite un “cassonetto_id”.

Ogni contenitore ha un parametro “stato” che indica il livello di riempimento.

Si intende, con questo metodo, fornire un sistema di esplorazione dell’UrbanDataset senza recuperare tutti i dati ma solo quelli interessanti, per esempio: “voglio lo stato del primo cassonetto in via XX Settembre”.

La soluzione, rispettante l’interfaccia descritta, viene proposta tramite le seguenti chiamate POST:

1. Chiamata1 a
   [URL_BASE]/UrbanDatasetGateway/explore
   con
   resourceParameter=contenitore
   Ritorno: { “cestino”, “cassonetto” }

2. Chiamata2 a
   [URL_BASE]/UrbanDatasetGateway/explore
   con
   resourceParameter= cassonetto_id
   inputParameter=contenitore
   inputValue=cassonetto
   inputParameter=via
   inputValue=XX Settembre
   Ritorno: { [cassonetto_id1], [cassonetto_id2] }
   (ovvero i cassonetti di via XX Settembre)

3. Chiamata3 a
   [URL_BASE]/UrbanDatasetGateway/explore
   con
   resourceParameter=stato
   inputParameter=cassonetto_id
   inputValue=[cassonetto_id1]
   Ritorno: { “full” }

Si noti che il *requestParameter* e gli *inputParamer* sono noti poiché compresi nella definizione ontologica dell’UrbanDataset, mentre gli *inputValue* sono ricavati dalle precedenti chiamate.