Link documento: Data prima stesura: Ultima modifica: Versione: <u>installazione\_gateway\_1.6</u> 14/05/2019 19/03/2020 1.6.0

# Indice del documento

Indice del documento	1
Installazione del framework Sme.UP Gateway	2
Modalità uno: installazione manuale su singolo server	2
Prerequisiti software per installazione manuale	2
Procedura per l'Installazione manuale	3
Modalità due: installazione docker	6
Prerequisiti software per l'installazione docker	6
Struttura installazione come macchina docker	6
Procedura dettagliata per l'installazione sotto docker	10
Inizializzazione del sistema SG in docker	11
Avvio e spegnimento del sistema SG in docker	12
Descrizione tecnica dell'installazione docker	14
Modalità tre: installazione come macchina virtuale	15
Modalità di installazione	15
Configurazione del framework	16
Installazione dei servizi A37 e A38 nel framework SG	22
Analisi dei log di sistema	24
Dove trovare i file di log	24
Nomenclatura dei file di log	24
La console UPP LO_080 per la gestione dei microservices	27
Premesse	27
Funzioni disponibili nella console di gestione	27
Dashboard	27
A37 Plugins	28
A38 Plugins	29
Queue Rabbit	31
Logs	32

# Installazione del framework Sme.UP Gateway

In questo documento verrà descritta in dettaglio la procedura di installazione di una istanza di Sme.UP Gateway (d'ora in avanti abbreviato con SG). Per la sua natura di applicazione web, SG può essere installato con diverse configurazioni operative, sia su singola macchina server che su macchine multiple in rete locale o geografica. Qui verrà però descritta la configurazione standard, quella che si presume possa soddisfare la maggior parte delle esigenze tipiche di utilizzo del framework.

E' possibile installare SG in tre modalità diverse:

- 1) Installazione manuale su singolo server
- 2) Installazione docker
- 3) Installazione con virtual machine precompilate

A seguire la descrizione della procedura di installazione per ognuna delle tre modalità.

Modalità uno: installazione manuale su singolo server

Prerequisiti software per installazione manuale

Per il corretto funzionamento di una istanza SG è necessario che sul sistema ospitante siano installati i seguenti software:

- 1. Sistema operativo di tipo server. Si consiglia vivamente un sistema GNU Linux server a 64 bit (ad esempio, Ubuntu Server di Canonical)
- 2. Java Virtual Machine in versione 1.8 o superiore
- 3. Payara Web Server in versione 5.183 o superiore
- 4. Apache Maven, versione 3.5 o superiore
- 5. RabbitMQ, versione 3.5 o superiore

Le procedure di installazione di questi software possono essere trovate facilmente nei rispettivi siti di riferimento ed esulano dagli scopi di questo documento. Non esistono limitazioni per quel che riguarda il sistema operativo del sistema ospitante visto che i software richiesti sono disponibili per tutti i principali sistemi. E' comunque consigliata l'installazione su un sistema GNU Linux server a 64 bit. Procedura per l'Installazione manuale

- 1. Controllare che sul sistema siano installati i software richiesti (lista nel punto precedente) e che siano correttamente funzionanti.
- Configurare Payara per abilitare il supporto al framework SG. Queste configurazioni sono molto importanti perché senza di esse il framework SG non funzionerebbe correttamente. I setup possono essere inseriti dalla console web di gestione del server Payara, solitamente accessibile all'indirizzo <u>http://localhost:4848</u>. Le impostazioni da modificare sono le seguenti:
  - 2.1. Configurazione della JVM. Andare alla voce

Configuration  $\rightarrow$  server-config  $\rightarrow$  JVM settings  $\rightarrow$  JVM options

ed inserire (o modificare se già presenti) le seguenti opzioni:

```
-server
-Xmx2048m
-Xms512m
-Dfish.payara.classloading.delegate=false
-Dfile.encoding=UTF-8
```

2.2. Configurazione del numero massimo di thread http attivabili. Andare alla voce:

Configuration  $\rightarrow$  server-config  $\rightarrow$  Thread pools  $\rightarrow$  httpd-thread-pool

ed impostare i valori:

Min Thread Pool Size = 100 Max Thread Pool Size = 2000

2.3. Configurazione del numero massimo di thread attivabili. Alla voce:

Configuration  $\rightarrow$  server-config  $\rightarrow$  Thread pools  $\rightarrow$  thread-pool-1

ed impostare i valori:

Min Thread Pool Size = 100 Max Thread Pool Size = 2000

- 3. Scaricare dal sito di distribuzione il pacchetto zip di installazione, denominato etc-dir-1.6.0.zip (questo il <u>link diretto</u>). Il file contiene la cartella di lavoro del gateway denominata **etc** e va copiata nella directory home del sistema operativo ospitante quindi /home/<username> per sistemi Linux oppure C:\Utenti\<username per i sistemi Windows (o similare, a seconda delle versioni)</p>
- 4. Scaricare dal sito di distribuzione il pacchetto zip di installazione, denominato smeup-gateway-1.6.0.zip (questo il <u>link diretto</u>). Il file contiene una serie di file in formato war che contengono le applicazioni web che compongono il framework. Scompattare i file e copiarli in una cartella locale al sistema, non è importante il posizionamento di questa cartella.
- 5. Avviare la console di gestione Payara (di solito attiva su <u>http://localhost:4848</u> a meno di installazioni particolari) e deployare le applicazioni web contenute nel file zip scaricato al punto precedente. E' consigliabile (ma non necessario) seguire un ordine prefissato nell'installazione delle applicazioni. Inoltre è consigliabile definire in fase di deploy un "deployment order" che garantisca che le web application vengano avviate con un ordine specifico. Le application da installare sono le seguenti:

Nome applicazione	Deployment order
gtw-hub.war (obbligatorio)	50
gtw-logger (obbligatorio)	70
gtw-resource-manager	100
gtw-config-manager.war	100
gtw-a37-smeup-dispatcher.war	100
gtw-a37-mqtt-dispatcher.war	100
gtw-a37-influxdb-dispatcher.war	100
gtw-dummy-producer	100



Installazione del framework Sme.UP Gateway
--

gtw-dummy-consumer	100
gtw-a37-http-dispatcher.war	100
gtw-as400-connector.war	100
gtw-smeup-adapter.war	100
gtw-as400-listener.war	100
gtw-deployer.war	100
gtw-smeup-ws.war	100

N.B. la precedente tabella riporta tutti i microservices oggi disponibili in SG. A seconda delle installazioni e delle funzionalità desiderate è possibile installare ed attivare solo una parte dei microservices disponibili. I soli microservices che devono sempre essere installati perchè indispensabili al funzionamento del framework sono il gtw-hub e il gtw-logger.

La situazione finale dovrà essere la seguente:

🝐 Smeup Gateway - Google Driv 🗙	Installazion	e del Smeup-gate 🗙 🕴 🔟 (	Google Calendar - ottobre 20 🛛 🗙	<pre>✔ Applications × +</pre>				
$\leftrightarrow$ $\rightarrow$ C $\triangle$ (i) localhost:48-	18/common/in	idex.jsf					* 🗣 🖬 🖋 🚯	🛯 🗳 🏘 🏮 🚺 🗄
👖 App 🖿 Cloud 🖿 Email via web	Interessa	anti 🖿 Internet 🖿 Lavo	ro 🖿 Link Utili 🖿 Satellit	e TV 🖿 Servizi 🖿 Soldi 🖿 Streaming 🖿	Uso frequente 🗋 Api Index 👪 Cerca torrent TNT	D iDope - the torre	nt 💿 Cronologia delle p	Download Consul     w
User: admin Domain: domain1 S	erver: localhost					ш	me <u>About</u> d <sup>e</sup> <u>Help</u>	C Enable Asadmin Recorder
🗲 payara serve	er 🥯							
Common Tasks **	Applications of	tions an be enterprise or web applica	ations, or various kinds of modul	es. Restart an application or module by clicking on the	reload link, this action will apply only to the targets that the a	pplication or module is	enabled on.	
Instances	Deployed	Applications (10)						
Clusters (Deprecated)	87 8	Deploy Undeploy	Enable Disable	Filter:				
Applications	Select		Deployment Order	Time Taken To Deploy (Milliseconds)	Date/Time When Application was deployed	Enabled 👌		Action
Monitoring Data		gtw-a37-dispatcher	100	10079	8-ott-2018 15,22,56		elb, web	Launch D   Redeploy   Reload
Resources     Goncurrent Resources		gtw-as400-adapter	100	3598	8-ott-2018 15.23.00		elb, web	Launch D   Redeploy   Reload
E Connectors		otw-as400-listeper	100	3088	8 ott 2018 15 23 03		elb web	Launch D   Redectory   Beload
JDBC JMS Resources		atw-config-mensoer	100	5671	8-0H-2018 15 23 00		eih weh	I sunch <sup>G</sup> I Redenloy I Reload
► Y JNDI		atu daplayar	100	3330	8 off 2018 15 23 12		olb web	Loundh D   Redenley   Reland
JavaMail Sessions Adapter Configs		STIL-SERIES IST			0.0000000000000000000000000000000000000		eju, weu	
V E Configurations		g <u>tw-nub</u>			8-08-2018 15:23.16		ejo, web	Launch C   Redeploy   Reiced
default-config      server-config			100	3141	8-ott-2018 15.23.19		ejb, web	Launch <sup>G</sup>   Redeploy   Reload
La Support		gtw-resource-manager	100	3448	8-ott-2018 15.27.58		ejb, web	Launch <sup>of</sup>   Redeploy   Reload
				3485	8-ott-2018 15.23.26		ejb, web	
				5445	8-ott-2018 15.23.31		ejb, webservices, web	
								4

A questo punto il sistema SG è correttamente installato ma per poter essere utilizzato deve essere opportunamente configurato, come da indicazioni fornite nel successivo paragrafo <u>"Configurazione del framework"</u>



# Modalità due: installazione docker

## Prerequisiti software per l'installazione docker

Per semplificare la distribuzione del framework SG è stata prevista una modalità di installazione semplificata da utilizzare in tutti i casi in cui è sufficiente una architettura mono server (tutti i microservizi installati sulla stessa macchina fisica). Questo tipo di installazione è basata su Docker, un framework open source per la distribuzione di applicazioni distribuite sotto forma di applicazioni virtualizzate (containers).

Per questo tipo di installazione i prerequisiti richiesti al sistema sono i seguenti:

- Un sistema operativo GNU Linux server a 64 bit. Docker è disponibile anche per ambienti Windows ma per la sua natura è molto più efficiente se utilizzato su macchine con sistema operativo Linux, che sono quindi consigliate. Consigliato Ubuntu in versione server ma vanno bene anche le altre distribuzioni standard.
- 2. Docker CE (community edition) ver 18.06.0 o superiore
- 3. Docker compose ver. 1.22 o superiore

L'installazione di questi software non verrà trattata in questo documento perché tutte le informazioni necessarie sono ampiamente disponibili in rete. La base di partenza per una installazione di un framework SG è quindi una macchina (fisica o virtuale, non fa differenza) su cui siano già stati installati il sistema operativo e i software richiesti.

Struttura installazione come macchina docker

Tutto il necessario per l'esecuzione sotto docker di una istanza si SG è normalmente contenuto in una singola cartella, solitamente chiamata **smeup-gateway-docker** (ma il nome non è rilevante) e può essere scaricato come singolo file zip dalla sezione prodotti del sito http://www.smeup.com

L'installazione di una istanza del framework SG consiste pertanto in tre semplici passaggi:

1. Download del prodotto dal sito <u>http://www.smeup.com</u>. Per comodità indichiamo anche il <u>link diretto</u> al file in formato zip da scaricare.



- Unzip e copia della cartella smeup-gateway-docker sul sistema di destinazione. Non è importante la cartella di destinazione ma è necessario che l'utente di sistema abbia le autorizzazioni di lettura e scrittura su questa cartella.
- 3. Configurazione dei servizi e avvio del sistema

Su una singola macchina possono convivere contemporaneamente più istanze attive di SG, ognuna installata all'interno di una cartella specifica e configurate in modo tale da non andare in conflitto sulle risorse di sistema (tipicamente le porte di comunicazione, che devono essere diverse per le varie istanze per evitare conflitti).

La struttura della cartella di installazione di SG (smeup-gateway-docker) è mostrata nella figura seguente.



# Installazione del framework Sme.UP Gateway

smeup-gateway-docker-1.2.1
🕨 🚞 docker-images
▼ 📄 etc
🕨 🚞 gtw-configuration
🕨 🚞 gtw-deployable
🕨 🚞 gtw-plugin-configuration
🕨 🚞 gtw-plugin-template
🕨 🚞 gtw-postboot
🕨 🚞 gtw-workingdir
🕨 🚞 gtw-yamlconfig
🕨 🚞 gtw-framework
maven_work_dirs
payara_work_dirs
📄 build.bat
と build.sh
📄 delete_images.bat
delete_images.sh
docker_commands.txt
Jocker-compose.override.yml
docker-compose.yml
Dockerfile-smeup-gateway-base
Dockerfile-smeup-gateway-rabbitm
📄 down.bat
down.sh
📄 load_images.bat
load_images.sh
save_images.bat
save_images.sh
📄 start.bat
start.sh
stop.bat
≥ stop.sh
📃 up.bat
💵 up.sh



Diamo una descrizione in dettaglio della struttura:

- Cartella **docker-images**: cartella che ospita le eventuali immagini locali dei container docker
- Cartella etc: cartella di lavoro per Smeup Gateway. Contiene una serie di sottocartella molto importanti
  - Cartella gtw-configuration: contiene i file di configurazione dei singoli microservices che costituiscono il framework SG, sotto forma di file con estensione \*.properties. I file di properties possono anche essere modificati a mano (se si sa cosa fare) ma si consiglia di gestirli con le procedure guidate previste dal sistema, descritte in seguito.
  - Cartella gtw-deployable: contiene i file dei microservices prodotti dal servizio di deploy. Fisicamente sono file in formato war da caricare su application server oppure file jar da eseguire come plugin esterni. I file sono prodotti da una procedura automatica quindi il contenuto di questa cartella non va modificato a mano.
  - Cartella **gtw-plugin-configuration**: contiene i file di configurazione dei microservices A37 e A38.
  - Cartella **gtw-plugin-templates**: contiene i template base per la creazione di microservices A37 e A38.
  - Cartella **gtw-postboot:** contiene il file che indica le applicazioni web da caricare nel framework in fase di avvio
  - Cartella **gtw-workingdir:** contiene file di lavoro generati da diverse funzionalità del framework.
  - Cartella **gtw-yamlconfig:** contiene i file in formato yaml che rappresentano le configurazioni dei plugin A37 e A38 in formato script Sme.UP
- Cartella **gtw-framework**: contiene di file war dei microservice base del framework SG. Sono i microservices necessari al funzionamento del sistema che vengono caricati in automatico in fase di avvio.
- Cartella **maven\_work\_dir**: cartella di lavoro per le procedure maven. E' inizialmente vuota.
- Cartella **payara\_work\_dir**: cartella di lavoro per l'application server Payara. Questa cartella contiene due importanti sottocartelle, entrambe vuote prima che il sistema venga avviato:
  - autodeploy: è la cartella di autodeploy del server Payara attivo. Se un file war viene copiato in questa cartella viene automaticamente pubblicato come web application in Payara
  - log: è la cartella in cui vengono prodotti tutti i file di log del sistema. In questa cartella troveremo i file di log dell'application server Payara ma anche i file di log dei singoli microservices SG prodotti dal servizio gtw-logger (servizio di log centralizzato). E' questa la cartella che



dovremo andare ad analizzare ogni volta che sarà necessario capire cosa è successo nel framework SG.

- File **docker-compose.yml**: file che definisce la struttura docker dell'intera applicazione come composizione di singoli container. <u>Questo file non va</u> <u>modificato nell'ambito di installazioni standard</u>.
- File **docker-compose.override.yml**: file di override di docker-compose. Consente di configurare l'istanza specifica di SG, attraverso la definizione delle porte necessarie al funzionamento del sistema. Consente inoltre di estendere la configurazione base del container docker con delle impostazioni specifiche per l'installazione corrente, ad esempio, impostazioni specifiche richieste da servizi A37 che si vogliono attivare nel sistema.
- File dockerfile-smeup-gateway-base: file che definisce il container Docker che fa da base per il framework SG. Abilità i servizi necessari (Payara, Maven, RabbitMQ) e li configura in modo opportuno. <u>Questo file non va</u> <u>modificato nell'abito di installazioni standard</u>.
- Files **build.sh**, **up.sh**, **start.sh**, **stop.sh** e **down.sh**: shell scripts la gestione del framework in docker. Presenti anche gli analoghi in versione \*.bat per installazioni su sistemi Windows.
- Files delete\_images.sh, save\_images.sh e load\_images.sh per la gestione delle immagini locali dei container docker. Presenti anche gli analoghi \*.bat per installazioni su sistemi Windows.

Procedura dettagliata per l'installazione sotto docker

- 1. Verificare cha la macchina di destinazione soddisfi i requisiti richiesti:
  - a. Sistema operativo Linux a 64 bit (ad esempio, Ubuntu server)
  - b. Docker versione 1.18 o superiore
  - c. Docker-compose versione 1.21 o superiore
- 2. Copiare sul sistema la cartella smeup-gateway-docker (il contenuto si è visto nel paragrafo precedente)
- 3. Controllare che tutte le cartelle contenute in docker-deploy siano accessibili in lettura/scrittura dagli utenti del gruppo **docker** (gruppo creato sul sistema da Docker in fase di installazione).
- 4. E' necessario configurare le porte socket usate dalla macchina Docker: i container Docker utilizzano servizi interni attestati su specifiche porte. Per accedere a questi servizi, i container pubblicano queste porte interne su delle porte locali alla macchina. E' importante scegliere delle porte locali che non siano già occupate da altri servizi e che non vadano in conflitto con altre installazioni di SG attive sulla stessa macchina. I prossimi due punti indicheranno le configurazioni necessarie.
- 5. Configurazione delle porte Payara



- Aprire il file docker-compose.override.yml
- Andare alla sezione ports sotto smeup-gateway

```
services:
smeup-gateway:
...
ports:
- "8080:8080"
- "8081:8081"
- "4848:4848"
```

le porte indicate in neretto sono le porte locali, quelle che possono essere modificate. La prima è la porta locale su cui verrà attestato il servizio HTTP interno a docker, la seconda è la porta HTTPS e la terza è la porta della console di gestione Payara.

- Andare alla sezione "Environment necessarie al framework" e modificare i valori delle variabili GTW\_ADDRESS con l'indirizzo fisico del server, e GTW\_PORT con la stessa porta di payara (default 8080);
- 6. Configurazione porte RabbitMQ
  - Sempre nel file docker-compose.override.yml
  - Andare alla sezione rabbitmq

rabbitmq: ports: - "**15672**:15672"

la porta 15672 in neretto è la porta locale su cui viene indirizzata la console di configurazione di RabbitMQ attiva all'interno del container Docker.

Inizializzazione del sistema SG in docker

Prima di avviare il sistema SG sotto docker è necessario eseguire una operazione di inizializzazione mirata alla creazione dei container docker necessari al funzionamento del sistema. Questa fase prevede due diverse procedure a seconda che il server di installazione abbia accesso o meno alla rete internet. <u>Tutti i comandi descritti nei punti successivi sono script che funzionano solo se lanciati dall'interno della cartella di root dell'installazione</u>.



- Server con accesso internet disponibile: eseguire il comando build.sh per consentire al sistema docker di scaricare da repository pubblici disponibili su rete internet i container necessari al funzionamento del sistema. Questa fase può richiedere parecchio tempo, soprattutto nel caso di sistemi con linee internet particolarmente lente
- Server con accesso internet non disponibile: in questo caso il server su cui viene installato SG in versione docker non ha accesso alla rete e non è in grado di scaricare i container docker necessari da repository pubblici. Le soluzioni possibili sono due:
  - a. Connettere temporaneamente il server a rete internet ed eseguire il comando build.sh per la creazione dei container. Una volta terminato, eseguire il comando save\_images.sh: questo comando salva nella cartella docker-images una copia locale dei container, che potranno essere utilizzati una volta che il server sarà sconnesso dalla rete internet. Il sistema SG potrà essere attivato utilizzando le copie locali dei container.
  - b. Se non è possibile connettere temporaneamente il server alla rete internet, è possibile scaricare dal sito smeup.com i container necessari (sotto forma di file gz, attenzione che possono avere dimensioni consistenti). Una volta scaricati i file, copiarli a mano nella cartella docker-images e lanciare il comando load\_images.sh che crea i container docker partendo dai file salvati. I file da scaricare sono il container smeup-gateway-base (<u>link diretto</u>) e il container smeup-gateway-rabbitmq (<u>link diretto</u>)

La fase di inizializzazione deve essere eseguita una sola volta e normalmente non deve essere più ripetuta fino a quando non si installano nuove versioni del sistema SG. Nel caso di danneggiamento o cancellazione accidentale dei container docker, i container stessi possono essere ricostruiti ripetendo la procedura. La ripetizione della procedura di inizializzazione produce ex novo i container ma non altera eventuali configurazioni.

Avvio e spegnimento del sistema SG in docker

Dopo aver eseguito l'installazione e l'inizializzazione, come da paragrafi precedenti, è ora possibile avviare per la prima volta il sistema SG sotto docker. Vediamo in dettaglio la procedura di avvio e di spegnimento:

- Avvio e spegnimento con gestione delle risorse
  - a. Comando **up.sh**: inizializza il framework, crea le risorse ed avvia i container docker.



- b. Comando down.sh: ferma i container docker e rilascia le risorse
- Avvio e spegnimento semplici:
  - a. Comando **start.sh**: avvia i container ripristinando lo stato presente al momento del precedente spegnimento
  - b. Comado **stop.sh**: ferma i container e congela lo stato delle risorse

Le regole per la gestione dell'avvio e dello spegnimento possono essere riassunte nei seguenti punti:

- Dopo l'installazione e la fase di build, <u>il primo avvio del framework SG deve</u> <u>essere fatto con un comando up</u>. Non è possibile eseguire un comando start se prima non si è fatto almeno una volta un up.
- Il framework in esecuzione può essere fermato in due modi: o con il comando stop se si intende mettere in pausa il sistema oppure con il comando down se si deve spegnere il sistema e rilasciare le risorse (ad esempio perchè si deve effettuare un aggiornamento).
- Dopo un comando di stop il sistema può essere riavviato solo con un comando di start. Analogamente, dopo un comando down il riavvio richiede un comando up
- Il sistema può essere aggiornato solo dopo che è stato eseguito uno spegnimento con rilascio delle risorse (down)

A titolo di riepilogo, la seguente immagine mostra la timeline del processo di attivazione e avvio del framework SG

Sme.UP

### Installazione del framework Sme.UP Gateway



## Descrizione tecnica dell'installazione docker

L'installazione di un applicativo in un contesto docker prevede la creazione di singoli container che rappresentano una istanza dell'applicativo e che possono essere caricati ed eseguiti in docker in modo dinamico. Il framework SG è però composto da più applicativi che devono essere attivati contemporaneamente e che presentano delle specifiche dipendenze tra loro. Per questo motivo, il framework SG non utilizza direttamente docker ma sfrutta le funzionalità offerte da docker-compose: come dice il nome stesso, docker-compose è un compositore dinamico di container docker che consente di creare strutture complesse attraverso l'aggregazione di singoli container docker in un unico ambiente condiviso. Docker-compose basa il suo funzionamento sull'esecuzione dinamica di alcuni script scritti in formato yaml, presenti nella cartella di installazione di SG. Quando si attiva una istanza SG utilizzando lo script docker-compose si verificano le seguenti cose:

- 1. Viene attivato un container docker chiamato smeup-gateway-base che contiene una istanza di Payara Server opportunamente configurata secondo le necessità di SG.
- Viene attivato un secondo container, chiamato smeup-gateway-rabbitmq, che contiene un'istanza del server rabbitmq opportunamente configurata per SG e la console HTTP per la sua gestione
- 3. Nel container smeup-gateway-base vengono installate tutte le webapp necessarie al funzionamento di SG, sotto forma di file war da deployare
- 4. Vengono montate all'interno dei container docker delle cartelle locali del server. Questo consente di accedere a specifiche cartelle del container come se fossero delle cartelle locali del sistema: ad esempio, docker-compose replica in una cartella locale il contenuto della cartella utilizzata dal server Payara (attivo nel container smeup-gateway) per produrre i log del sistema. Questo facilita molto la consultazione dei log di Payara attivo all'interno del container.
- 5. Docker-compose si occupa anche di gestire le comunicazioni tra i container Docker e il server locale: definisce le risorse condivise, apre o chiude le porte di accesso ai servizi, crea una rete interna tra le istanze docker attive.

Si ricorda a tal proposito che i singoli container docker sono istanze basate su Linux e quindi la sintassi dei comandi per la gestione dei container è quella delle shell bash.

# Modalità tre: installazione come macchina virtuale

Il framework SG viene distribuito anche in formato VM (virtual machine) pronto per essere installato nei principali motori di gestione di macchine virtuali. La VM è una macchina virtuale basata su Linux CentOS e già configurata con tutte le dipendenze necessarie per il corretto funzionamento del framework SG. La VM contiene anche una istanza preinstallata di SG quindi è pronta per essere utilizzata (previa configurazione, descritta nella sezione "<u>Configurazione del framework</u>" di questo documento).

Le VM sono distribuite come file in formato \*.ova e sono disponibile per i tre principali gestori di macchine virtuali presenti sul mercato:

- VirtualBox: disponibile per sistemi Windows e Linux (<u>link per il download</u>)
- VMWare: disponibile per sistemi Windows e Linux (link per il download)
- HyperV: disponibile per sistemi Windows Server (link per il download)



Modalità di installazione

Le modalità di installazione delle VM in formato OVA all'interno dei relativi motori di gestione non rientra nelle finalità di questo documento. Fare pertanto riferimento ai manuali dei singoli prodotti.

# Configurazione del framework

Dopo aver installato il framework SG in una delle modalità viste nel precedente capitolo, è necessario procedere alla configurazione del prodotto. Le istruzioni seguenti sono indipendenti dalla modalità di installazione scelta.

Da notare che dopo la fase di installazione il framework si attiva in modalità a funzionalità ridotta; le funzionalità complete del framework si attivano solo dopo che la procedura di installazione è stata terminata.

Prima di procedere alla configurazione vera e propria è necessario controllare l'avvenuto avvio del sistema. Per far questo è sufficiente richiamare la pagina di debug del servizio gtw-hub selezionando:

```
http://<ip-address>:<port>/gtw-hub/api/services/debug
```

dove al posto di <ip-address> e <port> si devono inserire rispettivamente l'IP del server su cui gira il servizio e la porta HTTP scelta in fase di installazione, ad esempio:

http://127.0.01:8080/gtw-hub/api/services/debug

Se il framework SG si è avviato con successo verrà mostrata una pagina di riepilogo dello stato del sistema, che mostra le informazioni di base e la lista dei microservices attivi.



#### Installazione del framework Sme.UP Gateway

Microservice Debug: HUR

			Status	
Key	Value		Key	Value
Id	HUB		Active	true
Url	Http://localhost:8080/gtw-hub		Ready	true
Routing key	HUB.out		External	false
Binding key	HUB.in		Delocalized	false
gtwMicroservice.providerName	SRVFOR		A37	false
gtwMicroservice.checkOther	A37-SMEUP-DISPATCHER A37-SMEUP-DISPATCHER_EVT AS400 CONNECTOR DUMMY-CONSUMER DUMMY-PRODUCER SMEUI	-LISTENER AS400- P-ADAPTER SMEUP-WS	A38	false
gtwMicroservice.statisticsdelay	30000		Configuration	
gtwMicroservice.statisticssleep	300000		Configuration	
gtwMicroservice.checkCore	CONFIG-MANAGER DEPLOYER HUB LOGGER RESOURCE-MAN	NAGER	Microservice definition in	ison
gtwMicroservice.durableOucues	1		Microservice definition in	1001
	l sann			
gtwMicroservice.pingsleep	60000		List of endpoints	
gtwMicroservice.pingsleep gtwMicroservice.pingdelav	00000 30000		List of endpoints	
gwMicroservice.pingsleep gwMicroservice.pingsleep gtwMicroservice.queueServerHost sime.UP Gateway version: 1.5.0 Configurazione iniziale microser	00000 00000		List of endpoints Swagger endpoints	
gwMicroservice.pingsleep gwMicroservice.queueServerHost sime.UP Gateway version: 1.5.0 Configurazione iniziale microser Microservice list	00000  30000   vices		List of endpoints Swagger endpoints	
gwMicroservice.pingsleep gwMicroservice.queueServerHost ime.UP Gateway version: 1.5.0 Configurazione iniziale microser dicroservice list Microservice	00000 30000 vices	Active Ready	List of endpoints Swagger endpoints	
gu-Microservice pingsleep gw-Microservice pingsleap gm-Microservice queueServerHost ime.UP Gateway version: 1.5.0 Configurazione iniziale microser dicroservice list Microservice 337-SMEUP-DISPATCHER	00000 30000 Vices Vices Url Http://ocalhost.8080/ptw-a37-smcup-dispatcher	Active Ready	List of endpoints Swagger endpoints	
gwMicroservice.pingsleep gwMicroservice.queueServerHost smc.UP Gateway version: 1.5.0 Configurazione iniziale microser Microservice list Microservice A37-SMEUP-DISPATCHER A37-SMEUP-DISPATCHER	00000 30000 Url Vices Url Http://ocallost:8080/gtv-a37-smcup-dispatcher Http://ocallost:8080/gtv-a37-smcup-dispatcher	Active Ready frue frue frue frue frue	List of endpoints Swagger endpoints	
givMicroservice.pingsleep gtvMicroservice.queueServerHost gtvMicroservice.queueServerHost sime.UP Gateway version: 1.5.0 Configurazione iniziale microser dicroservice list Microservice List Microservice A37-SMEUP-DISPATCHER A37-SMEUP-DISPATCHER_EVT Listou-CONNECTOR	00000 30000 Url Url Url Http://ocalhost:8080/gtv-a17-smcup-dispatcher Http://ocalhost:8080/gtv-a17-smcup-dispatcher Http://ocalhost:8080/gtv-a17-smcup-dispatcher Http://ocalhost:8080/gtv-a17-smcup-dispatcher	Active Ready frue frue frue frue frue frue	List of endpoints Swagger endpoints	
gwMicroservice.pingsleep gwMicroservice.queueServerHost sme.UP Gateway version: 1.5.0 Configurazione iniziale microser Alcroservice list Microservice 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER S400-LISTENER	00000 30000 Virial State	Active Ready true true true true true true true true	List of endpoints Swagger endpoints	
gwMicroservice.pingsleep gwMicroservice.queueServerHost ime.UP Gateway version: 1.5.0 Configurazione iniziale microser dicroservice list Microservice 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER S400-CONNECTOR S400-CONNECTOR	00000 30000 Vices Vices Url Http://localhost:8980/gtv=a37.smcup-dispatcher Http://localhost:8980 gtv=a47.smcup-dispatcher Http://localhost:8980 gtv=a47.smcup-dispatche	Active Ready frue frue frue frue frue frue frue frue frue frue frue frue	List of endpoints Swagger endpoints	
gu-Microservice.pingsleep gw-Microservice.queueServerHost gm-Microservice.queueServerHost sme.UP Gateway version: 1.5.0 Configurazione iniziale microser Alicroservice list Microservice Statup-DisPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 349-00-000000000000000000000000000000000	00000 30000 Vices Vices Vices Url Http://ocalhost.8080/gtva37.smcup-dispatcher Http://ocalhost.8080/gtva37.smcup-dispatcher Http://ocalhost.8080/gtva37.smcup-dispatcher Http://ocalhost.8080/gtva400-insteare Http://ocalhost.8080/gtva400-insteare Http://ocalhost.8080/gtva400-insteare Http://ocalhost.8080/gtva400-insteare Http://ocalhost.8080/gtva400-insteare Http://ocalhost.8080/gtva400-insteare	Active Ready frue frue frue frue frue frue frue frue frue frue frue frue frue frue frue frue	List of endpoints Swagger endpoints	
gwMicroservice.pingsleep gwMicroservice.pingsleap gwMicroservice.queueServerHost inne.UP Gateway version: 1.5.0 Configurazione iniziale microser Alicroservice list Microservice list Microservice A37-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER 337-SMEUP-DISPATCHER S400-LISTENER CONFIG-MANAGER DEPLOYER UB	00000       30000       30000       Viccs       Url       [http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a17.smcup-dispatcher       Http://localhost:3080/gtv-a400-smcup	Active         Ready           true         true	List of endpoints Swagger endpoints	
gn-Microservice.pingsleep gnwMicroservice.pingsleap gnwMicroservice.queueServerHost Sme.UP Gateway version: 1.5.0 Configurazione iniziale microser Microservice list Microservice list Microservice A37-SMEUP-DISPATCHER A37-SMEUP-DISPATCHER A390-CONNECTOR A5400-LISTENER CONFIG-MANAGER DEPLOYER HUB LOGGER	o0000 30000 Url Http://ocalhost:8080/gtv-a37-smeap-dispatcher Http://ocalhost:8080/gtv-a37-smeap-dispatcher Http://ocalhost:8080/gtv-a37-smeap-dispatcher Http://ocalhost:8080/gtv-a37-smeap-dispatcher Http://ocalhost:8080/gtv-a90-listener	Active         Ready           frue         frue           frue         frue	List of endpoints Swagger endpoints	

Dopo aver controllato che il framework si sia correttamente avviato è necessario procedere alle configurazioni necessarie per l'abilitazione dei singoli servizi. I microservice da configurare sono tre, denominati **gtw-hub**, **gtw-as400-connector** e **gtw-a37-smeup-dispatcher**. La configurazione avviene richiamando le singole console HTTP di immissione dei dati e inserendo nel form visualizzato le informazioni richieste. Le configurazioni sono salvate come file di tipo properties all'interno di una cartella del sistema la cui posizione cambia a seconda della tipologia di installazione effettuata.

Tipo installazione	Cartella configurazione
Manuale	<user-dir>/etc/gtw-config/gtw-configuration</user-dir>
Docker	<docker-folder>/etc/gtw-config/gtw-configuration</docker-folder>
VM	<user-dir-in-vm>/etc/gtw-config/gtw-configuration</user-dir-in-vm>

Si consiglia comunque di non modificare a mano il contenuto dei file properties ma di affidarsi alle console HTTP di configurazione descritte di seguito (che agiscono sugli stessi file)

1. Configurazione di gtw-hub: richiamare la pagina



Installazione del framework Sme.UP Gateway

http://<ip-address>:<port>/gtw-hub/api/services/askconfig

e valorizzare il campo **gtwMicroservice.providerName** con il nome da assegnare alla istanza di SG. Questo nome (<u>lungo al max 6 caratteri</u>) è il nome con cui l'istanza SG sarà riconosciuta sul sistema AS400.

Configurable	properties	for microservice:	HUB
--------------	------------	-------------------	-----

Key	Value
Id	HUB
Url	Http://localhost:8080/gtw-hub
Routing key	HUB.out
BindingKey1	HUB.in
BindingKey2	
BindingKey3	
BindingKey4	
gtwMicroservice.providerName	SRVFOR
gtwMicroservice.statisticsdelay	30000
gtwMicroservice.statisticssleep	300000
gtwMicroservice.durableQueues	1
gtwMicroservice.pingsleep	60000
gtwMicroservice.pingdelay	30000
	Submit

Nell'esempio in figura il providerName è stato impostato a SRVFOR

2. Configurazione di gtw-as400-connector: richiamare la pagina

http://<ip-address>:<port>/gtw-as400-connector/api/services/askconfig

e valorizzare i campi:

gtwMicroservice.system: nome sistema AS400 gtwMicroservice.user: utente AS400 gtwMicroservice.pwd: password utente AS400 gtwMicroservice.env: ambiente Sme.UP su AS400 (codice numerico)

la configurazione del microservice gtw-as400.adapter configura in automatico anche il microservice gtw-as400-listener che condivide le stesse impostazioni.



# Configurable properties for microservice: AS400-CONNECTOR

Key	Value
Id	AS400-CONNECTOR
Url	Http://localhost:8080/gtw-as400-connector
Routing key	AS400-CONNECTOR.out
BindingKey1	AS400-CONNECTOR.in
BindingKey2	
BindingKey3	
BindingKey4	
gtwMicroservice.maxidle	5
gtwMicroservice.user	FORDAR
gtwMicroservice.ccsid	1144
gtwMicroservice.env	0030
gtwMicroservice.pwd	
gtwMicroservice.hub.URL	Http://localhost:8080/gtw-hub
gtwMicroservice.system	srvlab01.smeup.com
gtwMicroservice.maxtotal	10
gtwMicroservice.minidle	2

Submit

<u>N.B.: il parametro ambiente deve essere specificato in formato numerico (vedi figura)</u> e non nel formato alternativo di tipo testuale

3. Configurazione di gtw-a37-smeup-dispatcher: richiamare la pagina

http://<ip-address>:<port>/gtw-a37-smeup-dispatcher/api/services/askconfig

e valorizzare i campi:

gtwMicroservice.system: nome sistema AS400 gtwMicroservice.user: utente AS400 gtwMicroservice.pwd: password utente AS400 gtwMicroservice.env: ambiente Sme.UP su AS400 (codice numerico)

#### Configurable properties for microservice: A37-SMEUP-DISPATCHER

Key	Value
Id	A37-SMEUP-DISPATCHER
Url	Http://localhost:8080/gtw-a37-smeup-dispatcher
Routing key	A37-SMEUP-DISPATCHER.out
BindingKey1	A37-SMEUP-DISPATCHER.in
BindingKey2	#.out
BindingKey3	
BindingKey4	
BindingKey5	
gtwMicroservice.user	ASUP
gtwMicroservice.evtbindingKey1	
gtwMicroservice.env	0020
gtwMicroservice.hub.URL	Http://localhost:8080/gtw-hub
gtwMicroservice.maxtotal	200
gtwMicroservice.evtbindingKey2	
gtwMicroservice.maxidle	50
gtwMicroservice.ccsid	1144
gtwMicroservice.dispatcher.primary	1
gtwMicroservice.pwd	
gtwMicroservice.system	srvlab01.smeup.com
gtwMicroservice.mockedsleep	5
gtwMicroservice.mocked	1
gtwMicroservice.minidle	20
	Submit

# N.B.: il parametro ambiente deve essere specificato in formato numerico (vedi figura) e non nel formato alternativo di tipo testuale

La configurazione dei servizi (punti da- 9 al 12 della lista precedente) deve essere fatta solo al primo avvio del sistema. I valori immessi vengono memorizzati e mantenuti per gli avvii successivi. In ogni momento è comunque possibile accedere alla configurazione di uno qualsiasi dei servizi attivi e modificarne le impostazioni. E' sufficiente richiamare l'URL:

## http://<ip-address>:<port>/<nome servizio>/api/services/askconfig

sostituendo a <nome servizio> il nome del microservice che si vuole configurare. Ad esempio:



#### Installazione del framework Sme.UP Gateway

### http://localhost:8080/gtw-logger/api/services/askconfig

consente di accedere alla pagina di configurazione del microservice gtw-logger.

Ad ogni variazione della configurazione, il relativo servizio viene fermato e riavviato con i nuovi parametri di configurazione, consentendo così un setup del sistema senza la necessità di riavvio del framework.

Alla fine della procedura di installazione, il sistema SG è attivo e pronto ad ospitare i microservice operativi.

# Installazione dei servizi A37 e A38 nel framework SG

Come già accennato in precedenza, il framework SG può ospitare 3 tipologie diverse di servizi:

- 1. I servizi di tipo A37 (protocollo IOTSPI)
- 2. I servizi di tipo A38 (protocollo WSCSPI)
- 3. I web service Sme.UP

Il servizi relativi al punto 3 (pubblicazione di informazioni gestionali Sme.UP come web services), sono forniti da un unico microservice di sistema, il **gtw-smeup-ws.** Quindi, i web service Sme.UP possono essere abilitati o meno semplicemente attivando o meno il suddetto microservice.

Per i servizi di tipo A37 e A38 la procedura di attivazione è un pò più complessa perché si tratta di servizi ad istanza multipla. Questo vuol dire che in ogni momento sul sistema possono essere attivi più servizi A37 e A38 in contemporanea, servizi distinti tra loro solo dalla diversa configurazione in fase di creazione ed avvio.

Per capire il funzionamento, è opportuno riferirsi alla figura seguente, relativa alla creazione di un microservice di tipo A37. Le cartelle a cui si fa riferimento nella figura sono le cartelle presenti nella cartella di installazione docker. Il concetto di base è che una istanza specifica di un microservice A37 viene creata combinando un template predefinito e un file di configurazione.



La procedura di pubblicazione di un plugin A37 si basa su una serie di passaggi:

1. Scrittura su AS400 dello script specifico per il plugin A37. Lo script è solitamente un membro LOA\_XX contenuto in un file SCP\_SET. Lo script A37 definisce alcune informazioni basilari:



- II tipo di plugin IOTSPI utilizzato dal plugin A37, identificato dalla chiave GAV Maven
- Le informazioni di setup del plugin IOTSPI
- La struttura delle informazioni gestite
- Importazione della configurazione definita su AS400 come file di configurazione locale in formato YAML ospitato nella cartella gtw-yamlconfig. Questa operazione viene fatta con la UPP LO\_080 che fornisce una console di gestione.
- 3. Importazione del template. Il template dipende dal tipo di plugin da creare e corrisponde al GAV identificato nello script al punto 1. Il template viene creato attraverso l'utilizzo della UPP A£\_X14 e una volta prodotto deve essere copiato nella cartella gtw-plugin-template
- 4. Attraverso la UPP LO\_080, creare il plugin A37 definitivo. La UPP si interfaccia al microservice gtw-deployer e produce due oggetti:
  - Un plugin A37 in formato web application (war), posto nella cartella gtw-deployable
  - Un file specifico di configurazione, posto nella cartella gtw-plugin-configuration
- 5. A questo punto il plugin A37 può essere pubblicato ed avviato, sempre utilizzando le funzionalità offerte dalla UPP LO\_080

La procedura è quindi composta da tre fasi:

- 1. **Fase di configurazione**, che corrisponde ai passaggi 1, 2 e 3 della lista precedente
- 2. Fase di build, corrispondente al punto 4
- 3. E **fase di deploy/undeploy**, corrispondente al punto 5. Questa fase può essere eseguita più volte perché un plugin una volta prodotto può essere attivato o disattivato più volte.

La procedura precedente, descritta nell'esempio per un microservice A37, è la stessa identica per i microservices di tipo A38. Ovviamente cambieranno i file di configurazione e i template ma le fasi di deploy rimangono le stesse.

A seconda delle fasi di avanzamento della procedura di deploy, un microservice A37 o A38 si può trovare in uno dei seguenti stati:

1. **Configurato**: su AS400 è stato definito uno script LOA (37 o 38) di configurazione e nella cartella gtw-config esiste il corrispondente file yaml.



- 2. Creato: nella cartella gtw-deployable è stato creato il file war specifico per quel microservice
- 3. Pubblicato: il microservice è stato pubblicato in Payara come web service.
- 4. Attivo: il microservice pubblicato in Payara, si è correttamente registrato sul sistema e si è connesso al relativo plugin A37 o A38. Questo è lo stato di piena operatività di un plugin.

# Analisi dei log di sistema

Il framework SG durante il suo funzionamento genera una serie di file di log che consentono di tener traccia di tutto quello che accade. L'analisi dei file di log è lo strumento fondamentale per analizzare lo stato del sistema a seguito di malfunzionamenti o comportamenti inattesi.

Dove trovare i file di log

Il framework SG genera i file di log nella stessa cartella dell'application server Payara. Questa scelta è voluta, in modo da avere un unico punto in cui poter consultare tutti i file di log, sia quelli di Payara che quelli generati dal framework SG.

La cartella che contiene i file di log è pertanto diversa a seconda che il framework SG sia installato come web application o come container docker.

In particolare:

• Per installazioni come web application, i log si possono trovare nella cartella:

<payara install dir>/glassfish/domain/domain1/logs

• Per installazioni docker, i log si trovano invece nella cartella:

<smeup\_gateway\_docker\_dir>/payara\_work\_dirs/logs

Nomenclatura dei file di log

All'interno del framework SG, ogni microservice genera uno specifico file di log. Quindi il numero dei file di log generati è almeno pari al numero di microservice attivi



all'interno del framework. Alcuni microservices generano più di un file di log, al fine di tracciare con maggior precisione alcune funzionalità specifiche.

Il nome dei file di log è dato dal codice UUID del microservice.

Quindi i servizi base del framework generano i seguenti file di log:

Servizio	UUID	File di log
gtw-hub	HUB	hub.log
gtw-logger	LOGGER	logger.log
gtw-resource-manager	RESOURCE-MANAGER	resource-manager.log
gtw-config-manager	CONFIG-MANAGER	config-manager.log
gtw-a37-smeup-dispatcher	A37-SMEUP-DISPATCHER	a37-smeup-dispatcher.log
gtw-a37-influxdb-dispatcher	A37-INFLUXDB-DISPATCHE R	a37-influxdb-dispatcher.log
gtw-a37-mqtt-dispatcher	A37-MQTT-DISPATCHER	a37-mqtt-dispatcher.log
gtw-a37-http-dispatcher	A37-HTTP-DISPATCHER	a37-http-dispatcher.log
gtw-as400-listener	AS400-LISTENER	as400-listener.log
gtw-as400-connector	AS400-CONNECTOR	as400-connector.log
gtw-deployer	DEPLOYER	deployer.log
gtw-smeup-adapter	SMEUP-ADAPTER	smeup-adapter.log
gtw-smeup-ws	SMEUP-WS	smeup-ws.log

I microservices di servizio (A37 e A38) generano invece dei file di log multipli con un nome dipendente dal codice dello script da cui è stato generato il microservice.

Quindi nel caso di servizi A37:

Sezione	Sottosezione	Files di log generati
		a37-SE1.SB1.evt
SE1	SB1	a37-SE1.SB1.log



	a37-SE1.SB1.k10

dove:

- file \*.log: è il log generico del microservice
- file \*.evt: è il log che traccia gli eventi che il microservice ha ricevuto dal plugin A37 sottostante
- file \*.k10: è il log che traccia i comandi che il microservice ha ricevuto (sono solitamente inviati da AS400 con la funzione K10)

Una struttura analoga vale anche per i log dei servizi di tipo A38 con l'unica differenza che questo tipo di servizi, per la loro natura, non ricevono eventi e quindi non generano il file di log di tipo \*.evt.



# La console UPP LO\_080 per la gestione dei microservices

La scheda ha l'obiettivo di agevolare la gestione dei vari plugins A37 e A38 che girano sul framework SG.

### Premesse

L'oggetto associato ad una istanza di SG è un oggetto Sme.UP di tipo V3-LSE, lo stesso con cui vengono identificate le istanze dello Sme.UP Provider. La comunicazione con AS400 fa capo a code dati create nella apposita libreria SMEUPUIDQ e denominate ESTC<codice-SG> e ECTS<codice-SG> (dove codice-SG è il nome della istanza SG).

## Funzioni disponibili nella console di gestione

Una volta scelto lo Smeup-Gateway sul quale si vuole lavorare (in base al suo codice univoco di istanza) si presentano le seguenti opzioni:

≡	Sme. <mark>UP</mark>	← C ↓ Spotlight-ALT-SHIFT-I	2	📵 SMILIPPO 🏢 + 🗢 🔵 🗹 U	»
Gatew	Smeup-Gateway	0			0
ay - conso	© Conferma				=
egestion					>_
ā	Dashboard C	A37 plugins 🔿 A38 plugins 🚫 Queu	e Rabbit ု Logs		۵
×,					
	†₂ Significato	∜₄ Valore	†↓ Decodifica		
	Indirizzo	Http://localhost:8080/gtw-as400-adapter	Http://localhost:8080/gtw-as400-adapter		1
	Nome Server - coda	PRVL04	PRVL04		
	Status	8	1		
	AS400(Writer)	SRVLAB01.5MEURCOM	SRVLAB01.SMEUPCOM		
	Utente	PRVL04 @	PRVL04		
	Ambiente	0010 💠	0010		
	AS400(Dispatcher)	SRVLAB01.5MEURCOM	SRVLAB01.SMEUPCOM		
	Utente	PRVL04 @	PRVL04		
	Ambiente	0010 ф	0010		
	Servizi registrati	A37DISPATCHER	true		
		A37DISPATCHER_EVT	true		
		AS4LIS	true		
		AS4WRI	true		
		CONFIG	true		
		DEPLOYER	true		
		HUB	true		
		LOGGER	true		
		SMEUP	true		
		JHLOF NO	0.00		

Dashboard



In questa sezione si vedono una serie di informazioni generali sulla istanza SG: in particolare è possibile vedere la lista dei microservice attivi, il nome delle code di comunicazione e i collegamenti all'AS400.

### A37 Plugins

In questa sezione è possibile vedere la lista dei plugins A37 collegati allo Smeup-Gateway selezionato ed operare sul singolo plugin.

La matrice che mostra la lista dei plugin ha le seguenti colonne:

- Nome plugin
- Descrizione plugin
- Status Indica se il plugins è attivo o meno
- On-Line Indica se il plugins è on-line ovvero se è attivo all'interno dell'application server
- Ready Indica se il plugins è pronto per ricevere/inviare messaggi
- File War Indica la presenza o meno del file WAR/JAR del plugin
- External Indica se il plugin è un plugin che parte all'interno dell'application server, oppure se parte esternamente;

8	
37 plugins A38 plugins Oueue Rabbit Logs	
stics LOA37	
U Text	$T_{6}$ Status $T_{6}$ Orvilne $T_{6}$ Ready $T_{6}$ Flewar $T_{6}$ External
Generatore eventi generici	
rom template () Delete all plusin () Delete all A37 conf () Delete all A37 conf	
X. Step okran 17.DM1 (C. Relead olucini 17.DM1) (X. Step ali Pasajo	
	A37 plugins A38 plugins Queue Rabbit Logs Istics LOA37  To Too.  Generatore event generici  from templete  Delete all plugin Delete all A37 conf  Update all A37 conf

Ogni plugins può essere:

• Avviato (pulsante "Start plugin")



- Fermato (pulsante "Stop plugin")
- Aggiornato (pulsante "Reload plugin")

Le funzioni comuni a tutti i plugins sono:

- Stop di tutti i plugins (pulsante "Stop all plugins")
- Creazione singolo plugins A37
- Cancellazione di tutti i plugins A37
- Cancellazione di tutte le configurazioni A37
- Aggiornamento di tutte le configurazioni A37

In questa sezione è possibile anche verificare le statistiche relative all'utilizzo dei plugins.

No. 14 No. 1017							
Programs inst Programs Statistics LOA37							
Ora							
Timestamp <sup>†</sup>	Incoming $T_{\rm b}$	Consumed 1	Outgoing <sup>†</sup> L	Error 14	Codaattuale %	Tempo server T <sub>a</sub>	
2018-07-25 09:28:24.949		1	1	1	0	0	1,00
Dati Sessione							
Timestamp <sup>1</sup>	Incoming T <sub>2</sub>	Consumed <sup>1</sup> 4	Outgoing 11	Error %	Cods attuale <sup>9</sup> 1	Temposerver <sup>1</sup>	
2018-07-24 12:33:25.231		2	2	2	0	0	2,06
2018-07-24 12:38:24.961		5	5	5	0	0	0,31
2018-07-24 12:43:24.963		4	4	4	0	0	0,44
2018-07-25 09:28:24.949		2	2	1	0	0	1,00
		13	13	12	0		
Rappresentazione Sessione PRVL04							
6,0							omina
		C - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -				- Co	nsumed
							taoina
4,5							going
4,5						Err	100 H 100
3,0						Err	or

#### A38 Plugins

In questa sezione è possibile vedere la lista dei plugins A38 collegati allo Smeup-Gateway selezionato ed operare sul singolo plugin.

La matrice che mostra la lista dei plugins A38 presenta le seguenti colonne:

- Nome plugin
- Descrizione plugin



- Status Indica se il plugins è attivo o meno
- On-Line Indica se il plugins è on-line ovvero se è attivo all'interno dell'application server
- Ready Indica se il plugins è pronto per ricevere/inviare messaggi
- File War Indica la presenza o meno del file WAR/JAR del plugin

me. <mark>UP</mark>	C & Southerralt-shift-re	🕒 SALLPPO III + 🗢 🔵 🖸 U
Smeup-Gateway PRVL04	٩	
( Contentia		
O Dashboard	A37 plugins 🛛 💿 A38 plugins 📄 Queue Rabbit 📄 Logs	
Divelo list 10428		
t <sub>a</sub> Plugin	⁺å Text	T <sub>k</sub> Status T <sub>k</sub> On-line T <sub>k</sub> Ready T <sub>k</sub> Filewar
17.501-0	Video youtube	
<ul> <li>Create plugin 17.50</li> </ul>		
	1 from template Si Delete all plugin O Delete all A38 conf	
► Start plogin (7.501	Trom template     O belete all Puglin     O belete all ASS conf     Uppate all ASS conf       X     Stop plugin (7.501)     C Reload plugin (7.501)     X. Stop all Plugins	

Ogni plugins può essere:

- Avviato (pulsante "Start plugin")
- Fermato (pulsante "Stop plugin")
- Aggiornato (pulsante "Reload plugin")

Le funzioni comuni a tutti i plugins sono:

- Stop di tutti i plugins (pulsante "Stop all plugins")
- Creazione singolo plugins A38
- Cancellazione di tutti i plugins A38
- Cancellazione di tutte le configurazioni A38
- Aggiornamento di tutte le configurazioni A38;



#### Queue Rabbit

In questa sezione è possibile verificare quali sono le code RabbitMQ al momento attive sul sistema. Dalla console è possibile eliminare o pulire ogni singola coda purchè non sia al momento utilizzata.

Smeup-Gateway								
PRVL04	Q							
Dashboard A3	7 plugins 🚫 A3B plu	igins 🌘 Queue Ra	bbit 🔵 Logs					
Name	1) State	Ready Tu	Unacked to	Total 5	Incomings To	Delver/s	adus	
A37-17.DM1	idie Idie	0	0	0	0.00	0,00	0.00	
A37DISPATCHER EVT	idle	0	0	0	0,00	0,00	0.00	
A39-17 601	idle	0	0	0	0.00	0,00	0.00	
AS4LIS	rupping	0	0	0	0.20	0.20	0.20	
AS4WRI	idle	0	0	0	0.00	0.00	0.00	
CONFIG	running	0	0	0	0,40	0,40	0,40	
DEAD_LETTER_QUEUE	idle	0	0	0	0,00	0,00	0,00	
DEPLOYER	running	0	0	0	0,90	0,80	0,80	
HUB	running	0	0	0	0,60	0,60	0,60	
LOGGER	running	0	0	0	36,40	36,40	36,40	
& Purge queue A37-17.DM1	× Delete queue A37-17	.DM1						

Sme.UP

#### Logs

Smeup-Gateway PRVL04 © Conferma				
Dashboard A37 plugins A3	18 plugins 🔷 Queue Rabbit 🕘 Lo	gs		5 e lak
- 17 in the first sector is a sector in the sector in the sector is a sector in the sector in the sector is a sector in the sector in the sector in the sector is a sector in the sector in the sector in the sector is a sector in the sector i	Carecola C	4 Type	Size	
astronomine poten pri ostron da pa	E 24/07/18 0	ing	4 800	http://drainost.susu/gtw-logger/ap/getlog/as/-1/.dm1-s10.log ~
Server.108_2018-07-24121-17-24	24/07/18 0	10g_2018-07-24121	4.862	<sup>op</sup> <u>http://iocalhost.8080/gtw-logger/api/getiog/server.log_2018-07-24T21-17-24</u>
nuclog i	La 24/07/18 0	1	2.048	Contraction State Strate St
Server.108_2018-07-25103-42-24	G 25/0//18 ↔	108_2010-07-23103	4.003	Antropylocalnosts000/wtw-logger/ap/getlog/server.log. 2018-07-25103-42-24
17.dm Llog	La 24/07/18 0	log	0	Co http://localhost.3030/gtw-logger/apl/getlog/i/.dml.log.4/
837-17.0m1.Jog	24/07/18	log	34	op http://iocalhost.8080/stw-logger/api/getlog/a37-17.0m110g
server.log	E 25/07/18 ₽	log	4.107	G <sup>5</sup> http://iocalhost:3080/gtw-logger/apl/getlog/server.log. <sup>10</sup>
838-17.901.00g	■ 24/07/18 ©	log	2	99 http://localhost.8080/gtw-logger/api/getlog/a38-i7.s01.log.9
as4lisJog	□ 25/07/18 ₽	log	378	C http://ioralhost:8080/gbw-logger/apl/getlog/as4lis.log.

Questa sezione mostra la lista dei file di log disponibili; i singoli log possono essere consultati direttamente da console oppure possono essere scaricati in locale per una consultazione offline.

Per la consultazione da console basta cliccare sul nome del file, il contenuto del log verrà mostrato in una finestra di popup.



# Installazione del framework Sme.UP Gateway

GLANCER Smeup Gateway - cor		General Galivery- cons			
Smeup Gateway - cor		C 🛛 🕴 🗹			
eup Gateway - cor			- = × î		
		PRVL04 - /apt/payara41/glassfish/domains/domain1/logs/stdout.log			
unde gestione «*	* N * N * N * N * N * N * N * N	25.07.2018 0213146,479 THEO LOG From: HUB Message: Gateway receive service registration request from CONFIG 25.07.2018 0213146,478 THEO LOG From: ANARIM Message: Gateway receive service registration request from DEPLOYER 25.07.2018 0213147,028 THEO LOG From: ANARIM Message: Gateway receive service registration request from CONFIG 25.07.2018 0213147,048 THEO LOG From: ANARIM Message: Gateway receive service registration request from DEPLOYER 25.07.2018 0213147,048 THEO LOG From: ANARIM Message: Fonding message from ASARIM the CONFG in with Cid 1512 25.07.2018 0213147,018 THEO LOG From: ASARIM Message: Invoked wil: HTtp://Localhostis000/gtu-config-manager/apl/invoke with to 45.07.2018 0213147,018 THEO LOG From: ASARIM Message: Fonding message from ASARIM ther CONFG in Mich CoNFG in Mich Cid 1512 25.07.2018 02133147,107 THEO LOG From: ASARIM Message: Sending message from ASARIM thir CoNFG in Mich CoNFG in Mich Cid 1512 25.07.2018 02133147,1107 THEO LOG From: CONFIG Message: Invoked wil: HTtp://Localhost:00004/gtu-config-manager/apl/invoke with to coNFIG Message: Theode Wil: HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with action (ConFig Message): Theode Wil: HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with for (ConFig Message): Theode Wil: HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with mich Cid 1512 25.07.2018 02133147,118 THEO LOG From: CONFIG Message: Throke Wil: HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with the CONFIG Message): Throke Mich Mich Cid 1512 25.07.2018 02133147,118 THEO LOG From: CANFIG Message: Invoked with TotMessage Invoked Wil: CONFIG Message: Throke Wil: HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with the CONFIG Message: Invoke Wil: HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with the CONFIG Message: Invoked Wilt TotMessage Informaction (ConFIG Message): Throke Wilt Mich Cid 1512 25.07.2018 02133147,118 THEO LOG From: CANFIG Message: Invoke Wilt HTtp://Localhost:0004/gtu-asA00-adapter/apl/invoke with the ConFIG Message: Invoke Wilt Mich Cid 152	A86025057 message IotNessage fith IotNessage [from=CONFIG, A86027034 ti LotNessage [from=AS40kRI, t queue 1532486027034 from 486027113 message IotNessage (IotNessage [from=AS40kRI, t queue 1532486027113 message IotNessage (IotNessage [from=AS40kRI, t queue 153248027113 from 486027182 message IotNessage IotNessage IotNessage IotNessage Message IotNessage message IotNessage Message IotNessage	10	ο. 
		[From-Skamit, Lo=CheFls.II, payloadiyJpB=1AY, payloadiyZude= 25,07.2618 02:3147,248 LHP CDC PFrom: CONFIG Newspace Transcence to the Config. In PayloadiyPB=1AY, payloadiyPE=1AY, payloadiyPE=	486027250 HAG027250 Hessage TotMessage		
		[11:00-Some on Consensus of periodiciple-four physical end of the fact of t	navloadvalues		

Se i file di log dovessero essere di grandi dimensioni, per una consultazione più agevole è comunque consigliabile scaricare il file in locale.