Guida di Installazione

Release 3.3.20

Link.it

09 giu 2026

Contents

CHAPTER 1

Introduzione

In questa sezione trovi una guida rapida per l'installazione della versione binaria di GovWay. Verifica e, se necessario, installa il software di base per GovWay come indicato nella Fase Preliminare. Un installer grafico ti guiderà nella personalizzazione della binary release verso la tua piattaforma.

CHAPTER 2

Fase Preliminare

Prima di procedere con l'installazione di GovWay è necessario disporre del software di base nell'ambiente di esercizio. Verificare i passi seguenti, procedendo eventualmente all'installazione dei componenti mancanti.

1. *Java Runtime Environment (JRE) 11* (è possibile scaricare JRE al seguente indirizzo: https://jdk.java.net/archive/)

   Verificare la configurazione dell'ambiente Java dell'Application Server. Si raccomanda una configurazione minima dei parametri della JVM, come segue:

   - -XX:MaxMetaspaceSize=512m -Xmx1024m

   Verificare inoltre che il charset utilizzato dalla JVM sia UTF-8:

   - -Dfile.encoding=UTF-8

2. *Application Server WildFly* (http://wildfly.org); viene supportato dalla versione 18 alla versione 26. In alternativa è possibile effettuare l'installazione su Apache Tomcat (http://tomcat.apache.org) versione 9.

   Nota

   GovWay supporta anche altri application server j2ee diversi da quelli citati, partendo dalla distribuzione sorgente.

3. Un *RDBMS* accessibile via JDBC. La release binaria è compatibile con i seguenti database:

   - *PostgreSQL 8.x o superiore*
   - *MySQL 5.7.8 o superiore*
   - *Oracle 10g o superiore*
   - *HyperSQL 2.0 o superiore*
   - *MS SQL Server 2019 o superiore*

   Il database deve essere configurato con un character encoding UTF-8 e una collation case-sensitive per garantire il corretto funzionamento dell'applicazione.

La distribuzione GovWay è stata estesamente testata prima del rilascio sulla seguente piattaforma di riferimento:

- *Openjdk 11 (version: 11.0.24+8)*
- *PostgreSQL 13 (version: 13.7)*, *PostgreSQL 16 (version: 16.2)* e *Oracle 11g ExpressEdition (version: 11.2.0.2.0)*
- *WildFly 18 (version: 18.0.1.Final)*, *WildFly 25 (version: 25.0.0.Final)*, *WildFly 26 (version: 26.1.3.Final)* e *Tomcat 9 (version: 9.0.91)*

CHAPTER 3

Esecuzione dell'Installer

1. Scarica qui la binary release di GovWay

2. Scompatta l'archivio, verifica ed eventualmente imposta la variabile d'ambiente *JAVA_HOME* in modo che riferisca la directory radice dell'installazione di Java. Lancia l'utility di installazione mandando in esecuzione il file *install.sh* su Unix/Linux, oppure *install.cmd* su Windows.

   Nota

   L'utility di installazione non installa il prodotto ma produce tutti gli elementi necessari che dovranno essere dispiegati nell'ambiente di esercizio. L'utility di installazione mostra all'avvio una pagina introduttiva.

3. Dopo la pagina introduttiva, cliccando sul pulsante *Next*, si procede con la scelta della *Modalità di Installazione*. Le scelte possibili sono:
   - *Modalità*:
     - *Nuova Installazione*: scelta da effettuare nel caso in cui si stia procedendo con una nuova installazione di GovWay.
     - *Aggiornamento*: scelta da effettuare nel caso in cui si stia procedendo con l'aggiornamento di una versione di GovWay precedentemente installata.
   - *Tipo*:
     - *Standard*: selezione del modello architetturale più semplice e adeguato nei casi più comuni
     - *Avanzata*: permette di selezionare caratteristiche architetturali avanzate nei casi con esigenze specifiche

Figure3.1: Introduzione

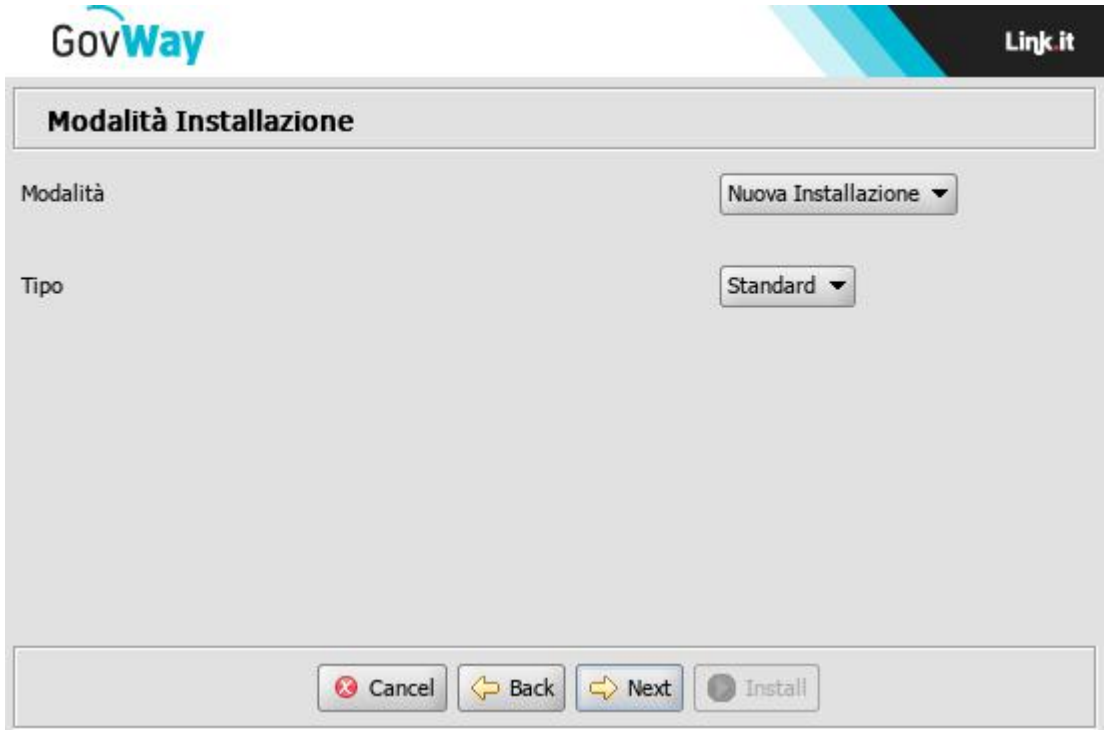

Figure3.2: Modalità di Installazione

3.1 Nuova Installazione

Supponiamo che la scelta sia quella di una nuova installazione. Vediamo come si sviluppa il processo di installazione:

1. Si procede con l'inserimento delle *Informazioni Preliminari*, che prevede i seguenti dati:

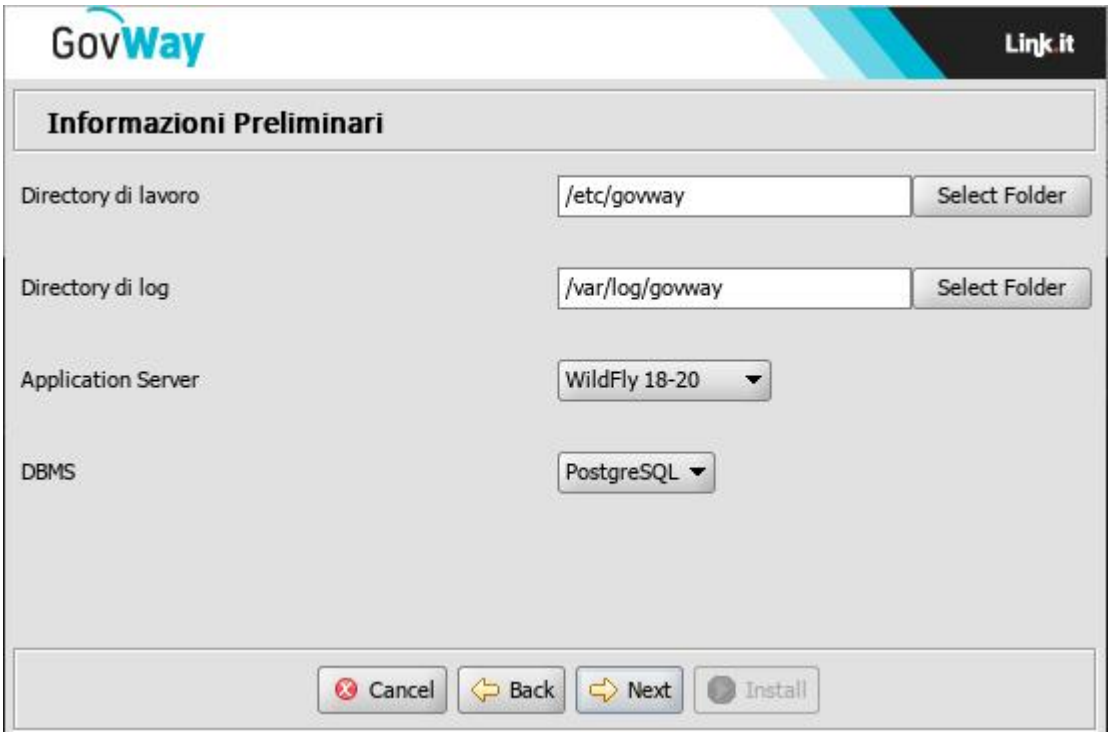

Figure3.3: Informazioni Preliminari

Operare le scelte sulla maschera di *Informazioni Preliminari* tenendo presente che:

- *Directory di lavoro*: una directory utilizzata da GovWay per inserire i diversi file di tracciamento prodotti. Non è necessario che questa directory esista sulla macchina dove si sta eseguendo l'installer; tale directory dovrà esistere nell'ambiente di esercizio dove verrà effettivamente installato il software GovWay.
- *Directory di log*: una directory utilizzata da GovWay per produrre i file di log. Non è necessario che questa directory esista sulla macchina dove si sta eseguendo l'installer; tale directory dovrà esistere nell'ambiente di esercizio dove verrà effettivamente installato il software GovWay.
- *DBMS*: il tipo di database scelto tra quelli supportati: PostgreSQL, MySQL, Oracle, HyperSQL, SQLServer.
- *Application Server*: il tipo di application server tra quelli supportati: WildFly (deve essere selezionata la voce che comprende la versione utilizzata tra: 18-21, 22-24 o 25-26) e Apache Tomcat (versione 9).

2. Al passo successivo si dovranno inserire tutti i dati per l'accesso al database ed in particolare:
   - *Hostname*: indirizzo per raggiungere il database
   - *Porta*: la porta da associare all'host per la connessione al database
   - *Nome Database*: il nome dell'istanza del database a supporto di GovWay. Non è necessario che questo database esista in questa fase. Il database di GovWay infatti potrà essere creato nella fase successiva purché il nome assegnato coincida con il valore inserito in questo campo.

Figure3.4: Informazioni Accesso Database

- *Username*: l'utente con diritti di lettura/scrittura sul database sopra indicato. Analogamente al punto precedente, l'utente potrà essere creato nella fase successiva dopo aver creato il database. Ricordarsi però di utilizzare il medesimo username indicato in questo campo.
- *Password*: la password dell'utente del database.

3. Il successivo passo richiede di stabilire le credenziali relative alle utenze di amministrazione per l'accesso ai cruscotti di gestione:

   I dati da inserire sono:

   - *Username/Password* relativi all'utente amministratore della govwayConsole.
   - *Username/Password* relativi all'utente operatore della govwayMonitor.

4. Nel successivo passo è possibile indicare se tra gli archivi generati devono essere inclusi i servizi che permettono la configurazione ed il monitoraggio di GovWay tramite API REST e i servizi di Health Check basati su API REST e/o SOAP.

5. Al passo successivo si dovranno inserire i dati relativi ai profili di interoperabilità supportati dal gateway:

   - *Profilo*: contrassegnare con un flag i profili aggiuntivi che saranno gestite da GovWay, scelti tra quelli offerti built-in dal prodotto:
     - *ModI*
     - *SPCoop*
     - *eDelivery*
     - *SdI (Fatturazione Elettronica)*

Figure3.5: Informazioni Utenza Amministratore

Figure3.6: Configurazione Servizi

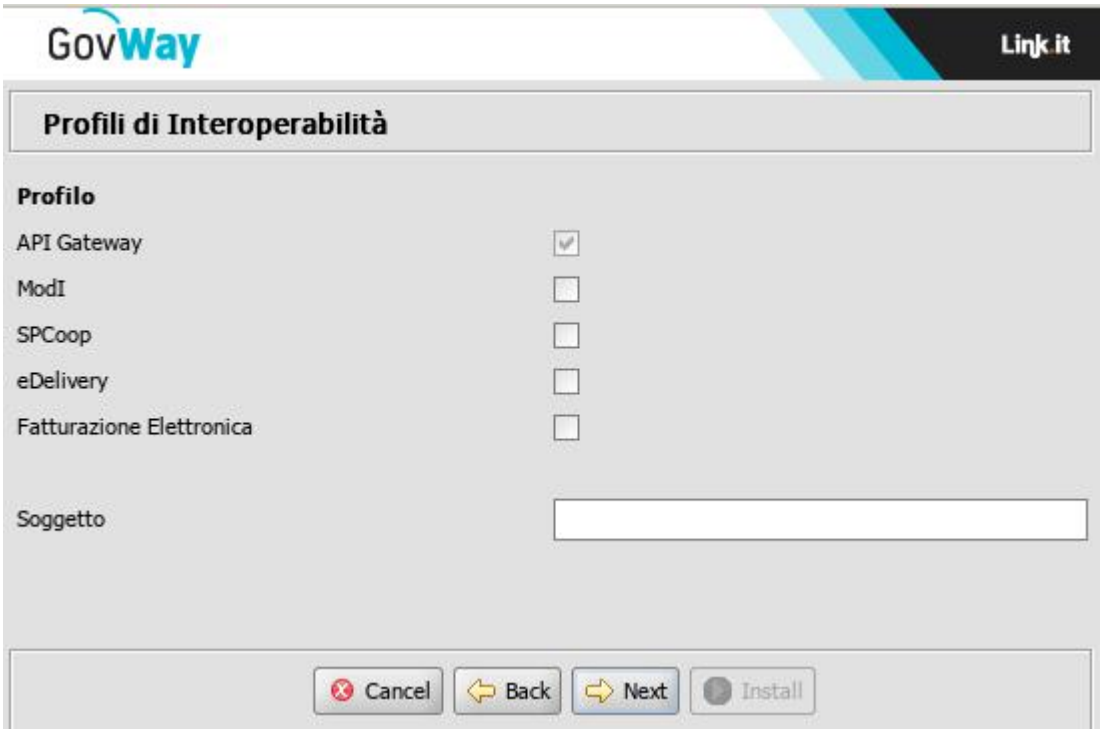

Figure3.7: Profili di Interoperabilità

Nota

Il profilo "API Gateway" viene sempre installato.

- *Soggetto*: nome del soggetto interno che verrà creato automaticamente.

6. Se si è scelto di includere il profilo eDelivery verranno presentati tre ulteriori tre passi di installazione. Nel primo passo viene richiesto di immettere la versione dell'Application Server e del Database associato alla versione di Domibus utilizzata.
7. Nel secondo passo, relativamente alla configurazione del profilo eDelivery, viene richiesto di immettere i relativi dati di configurazione.

   I dati di configurazione da immettere in questo step riguardano l'installazione di Domibus con la quale GovWay deve integrarsi per il dialogo con altri access point tramite il protocollo eDelivery. I dati richiesti sono:

   - HTTP Endpoint: gli endpoint per contattare l'access point domibus interno
     - Domibus MSH URL: endpoint pubblico per la raggiungibilità dagli altri access point
     - Domibus Backend WS URL: endpoint dei servizi di backend che saranno utilizzati da GovWay per l'integrazione a Domibus
   - Broker JMS: i dati di accesso al broker JMS utilizzato internamente da Domibus
     - Provider URL: endpoint del Broker JMS
     - Username/Password: credenziali per l'accesso ai servizi del Broker JMS
8. Nell'ultimo passo, relativamente alla configurazione del profilo eDelivery, verranno richiesti i dati di accesso al database utilizzato da Domibus:

Figure3.8: Configurazione eDelivery

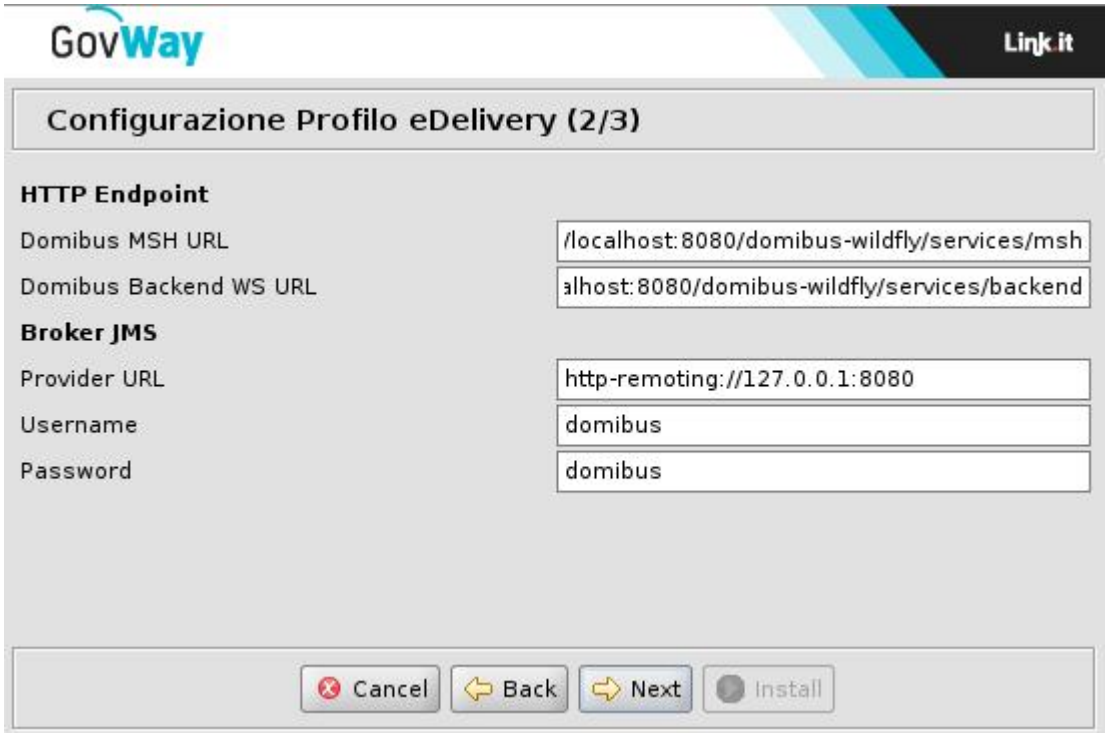

Figure3.9: Configurazione eDelivery (HTTP/JMS)

Figure3.10: Configurazione eDelivery (DBMS)

- *Hostname*: indirizzo per raggiungere il database
- *Porta*: la porta da associare all'host per la connessione al database
- *Nome Database*: il nome dell'istanza del database a supporto di Domibus.
- *Username*: l'utente con diritti di lettura/scrittura sul database sopra indicato.
- *Password*: la password dell'utente del database.

9. All'ultimo passo, premendo il pulsante *Install* il processo di configurazione si conclude con la produzione dei file necessari per l'installazione di GovWay che verranno inseriti nella nuova directory *dist* creata al termine di questo processo.

   I files presenti nella directory **dist** dovranno essere utilizzati nella fase successiva di dispiegamento di GovWay

3.2 Aggiornamento

Supponiamo che la scelta sia quella di aggiornare una installazione precedente. Vediamo come si sviluppa il processo per differenza rispetto al caso di una nuova installazione:

1. Il primo passo è quello di indicare la versione di GovWay da cui si parte per l'aggiornamento.
2. Al passo successivo, dove si indicano le informazioni preliminari, vi è il vincolo di indicare la medesima piattaforma database utilizzata per l'installazione che si vuole aggiornare.
3. Nel successivo passo è possibile indicare se tra gli archivi generati devono essere inclusi i servizi che permettono la configurazione ed il monitoraggio di GovWay tramite API REST e i servizi di Health Check basati su API REST e/o SOAP.
4. Nella maschera che permette la scelta dei profili di interoperabilità, vi è il vincolo di indicare almeno i medesimi profili utilizzati per l'installazione che si vuole aggiornare.

Figure3.11: Installazione

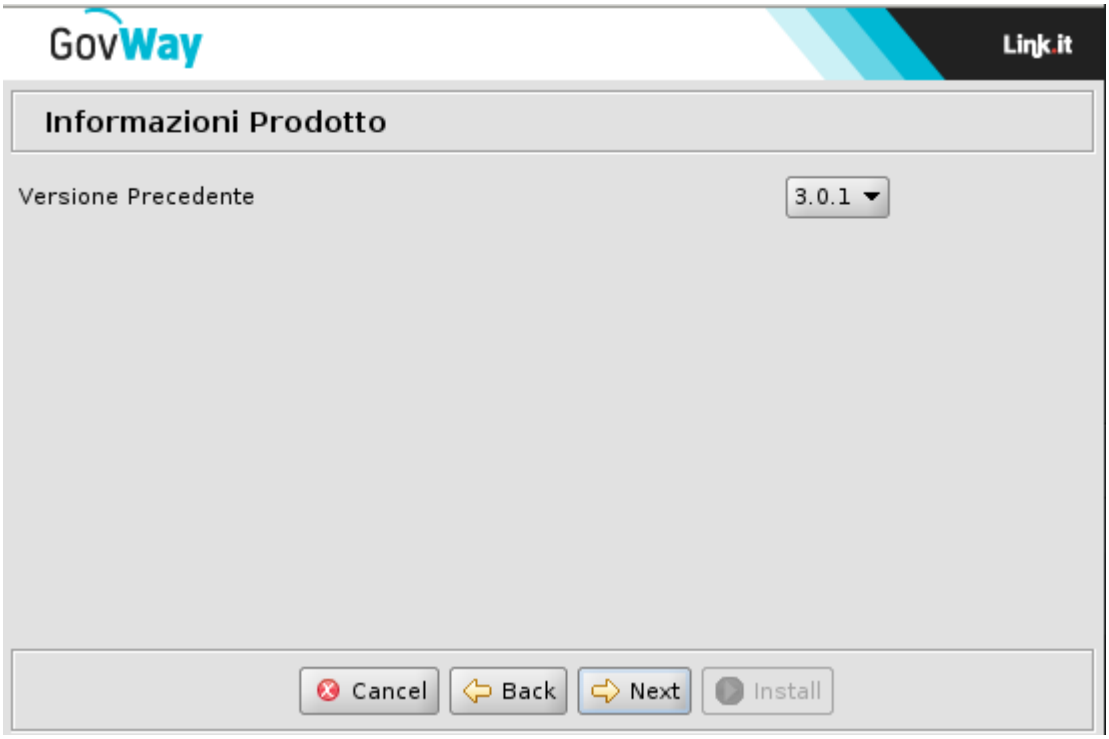

Figure3.12: Scelta versione precedente

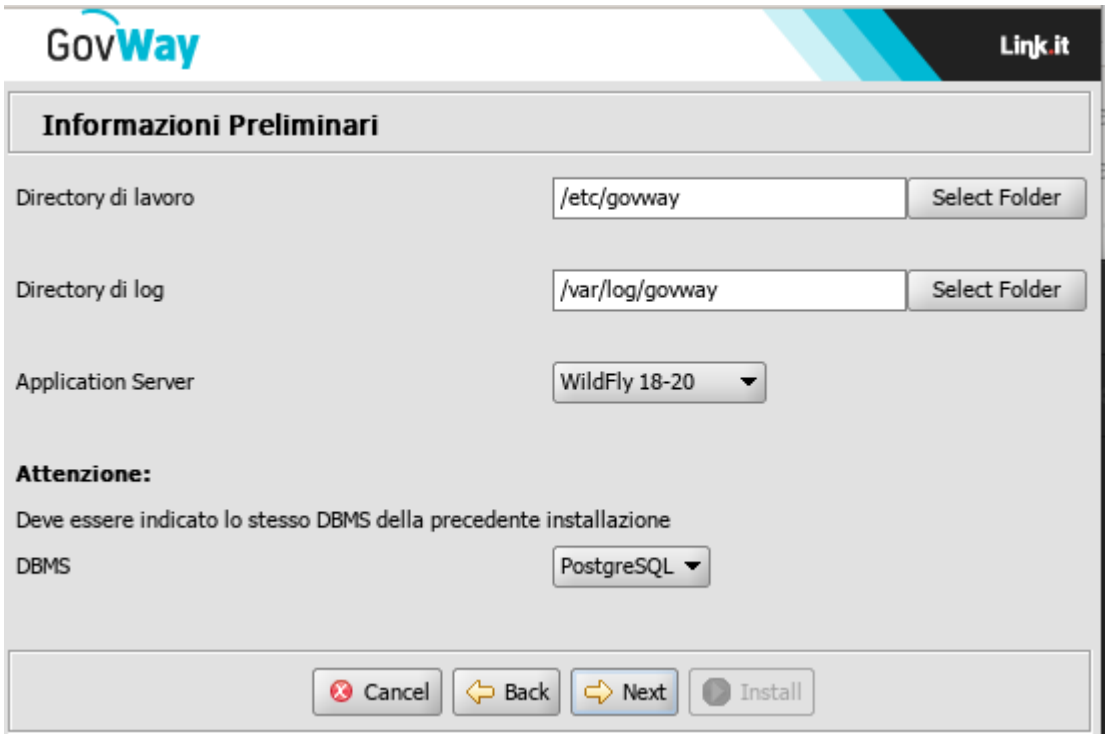

Figure3.13: Scelta piattaforma database identica all'installazione di provenienza

Figure3.14: Configurazione Servizi

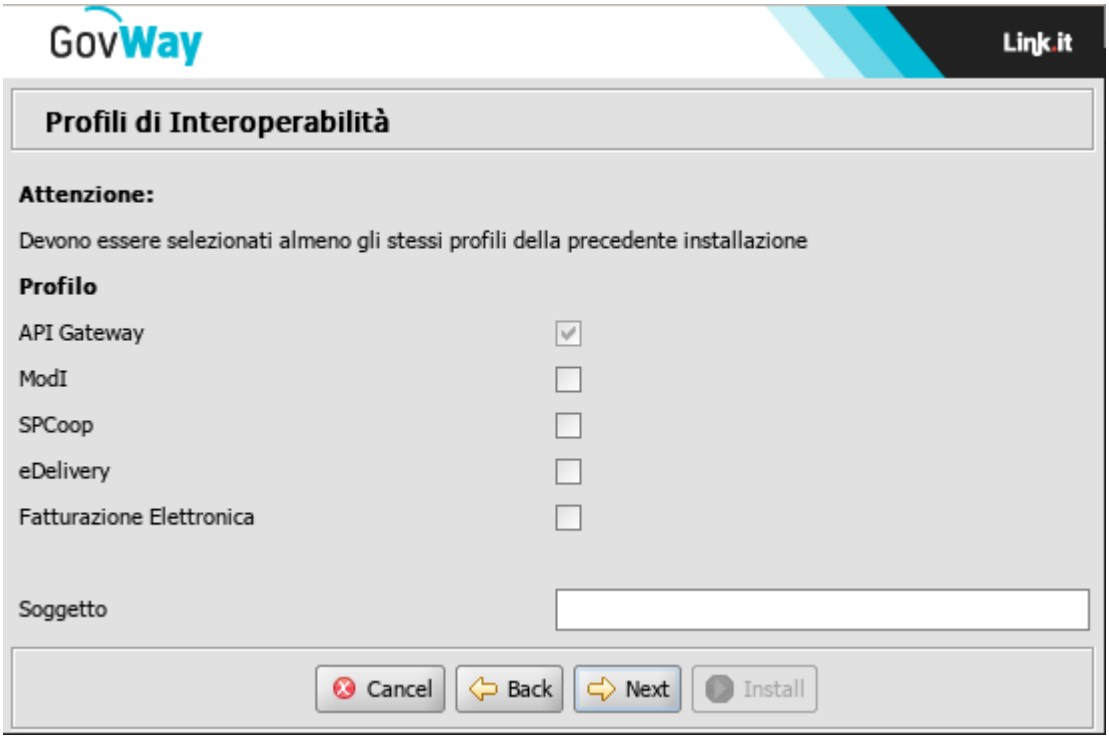

Figure3.15: Scelta Profilo di Interoperabilità

5. I rimanenti passaggi sono uguali al caso della nuova installazione con la differenza che non sarà disponibile la funzione per impostare le credenziali dei cruscotti grafici.

3.3 Modalità Avanzata

Se si è optato per la modalità di installazione «Avanzata», dopo aver visualizzato il pannello delle «Informazioni Preliminari», il processo d'installazione mostra un pannello aggiuntivo denominato «Informazioni Dispiegamento».

Le informazioni richieste da questo pannello servono a fornire le preferenze su come devono essere pacchettizzati i datasource e i singoli componenti applicativi, in accordo all'architettura di dispiegamento che si vuole adottare.

La sezione **DBMS** prevede le seguenti opzioni:

- *Configurazione*. Indica lo schema del database da utilizzare come repository delle configurazioni:
  - *Utilizza stesso database del Runtime*: le configurazioni risiedono nel medesimo schema di esercizio di Govway
  - *Database dedicato alla configurazione*: le configurazioni risiedono su uno schema separato da quello di esercizio di Govway
- *Tracce*. Indica lo schema database da utilizzare come repository delle tracce:
  - *Utilizza stesso database del Runtime*: le tracce risiedono nel medesimo schema di esercizio di Govway
  - *Database dedicato alle Tracce*: le tracce risiedono su uno schema separato da quello di esercizio di Govway
  - Opzione *Standard / Full Index*: se si sceglie l'opzione *Full Index*, il database sarà configurato con un indice contenente molteplici colonne che consente di migliorare le performance in fase di accesso alle tracce

Figure3.16: Informazioni Dispiegamento

- *Statistiche*. Indica lo schema database da utilizzare come repository delle statistiche:
  - *Utilizza stesso database del Runtime*: le statistiche risiedono nel medesimo schema di esercizio di Govway
  - *Utilizza stesso database delle Tracce*: le statistiche risiedono nel medesimo schema utilizzato per le Tracce
  - *Database dedicato alle Statistiche*: le statistiche risiedono su uno schema separato da quello di esercizio (e tracce) di Govway
  - Opzione *Standard / Full Index*: se si sceglie l'opzione *Full Index* il database sarà configurato con un indice contenente molteplici colonne che consente di migliorare le performance in fase di accesso alle statistiche

La sezione **Componenti Applicative** prevede le seguenti opzioni:

- *Configurazione e Monitoraggio*: indica l'ambiente in cui saranno dispiegate le console web:
  - *Utilizza stesso ambiente del Runtime*: le console web risiedono nel medesimo ambiente (application server) dove è in esecuzione il runtime di Govway
  - *Ambiente dedicato*: le console web risiedono in un ambiente (application server) distinto da quello del runtime di Govway
- *Generazione delle Statistiche*: scelta del meccanismo da adottare per il periodico aggiornamento delle statistiche:
  - *Utilizza stesso ambiente del Runtime*: il meccanismo di generazione delle statistiche è automatizzato con un sistema di integrazione al runtime di Govway
  - *Generazione tramite Applicazione Batch*: la generazione delle statistiche avviene tramite invocazione di un batch appositamente generato
- *Gestione dei Nodi*: Indica la modalità con cui si vuole gestire la configurazione dei diversi nodi di runtime:
  - *Statica*: modalità che prevede la registrazione manuale dei nodi presenti come descritto nella sezione *Configurazione in Load Balancing*. In questo caso la console di gestione consente

l'accesso a specifiche operazioni di manutenzione sui singoli nodi.

– *Dinamica*: in questo caso le operazioni di manutenzione dei singoli nodi non risultano più accessibili individualmente, ma viene indirizzato complessivamente il cluster.

> **Nota**
>
> La modalità *Dinamica* è pensata per gli ambienti docker dove la visibilità del singolo nodo viene meno. Oltre alla gestione differente, anche il monitoraggio non consentirà più di conoscere il nodo su cui è stata gestita la singola transazione.

In funzione delle opzioni, fornite in questo pannello, saranno proposti dalla procedura d'installazione alcuni passaggi aggiuntivi. In particolare, riguardo l'accesso agli schemi del database, in funzione del numero di datasource previsto saranno opzionalmente richiesti in altrettanti pannelli:

- *Informazioni DataSource dedicato alle Configurazioni*
- *Informazioni DataSource dedicato alle Tracce*
- *Informazioni DataSource dedicato alle Statistiche*

In funzione della modalità di *Gestione dei Nodi* indicata, nel caso si sia scelto la modalità *Dinamica* verrà presentato l'ulteriore pannello *Dispiegamento Dinamico* che consente di indicare i seguenti dati:

- *Servizio Check* URL: servizio che consente di monitorare lo stato dei nodi; è possibile utilizzare il servizio "/govway/check" di un qualsiasi nodo (es. acceduto tramite un bilanciatore) o un servizio fornito dal container.
- *Servizio Proxy* URL: servizio che consente di inviare un comando ad ogni nodo di runtime attivo; deve essere indirizzato il servizio "/govway/proxy" di un qualsiasi nodo attivo.

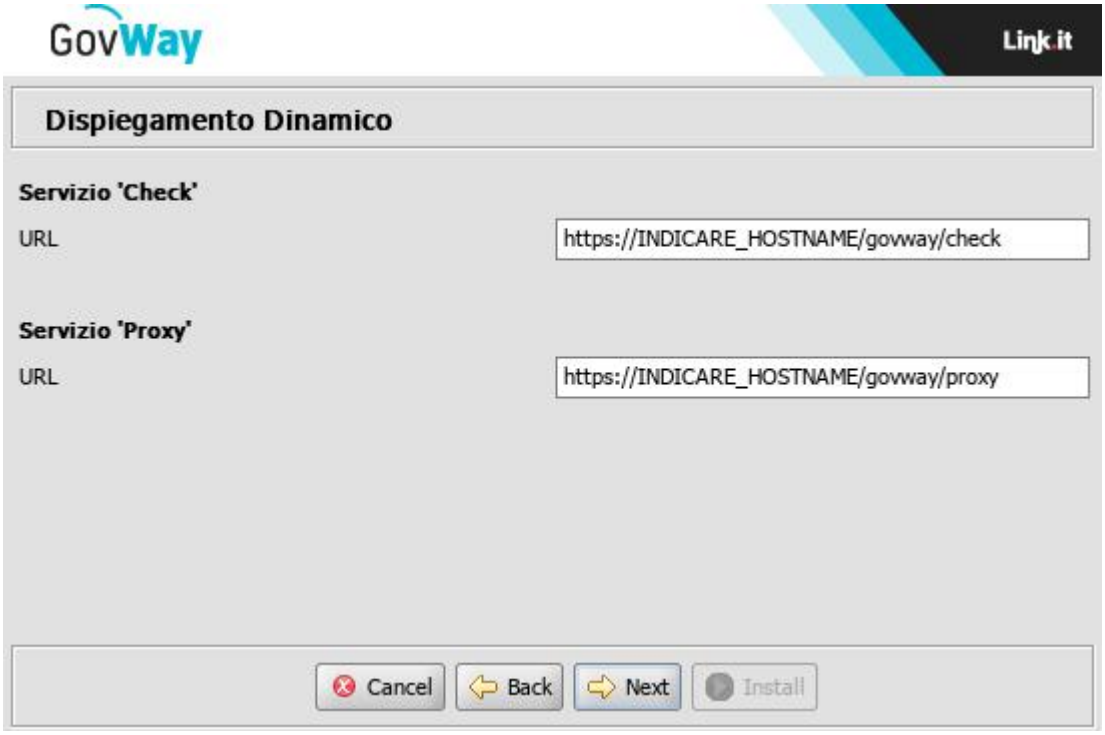

Figure3.17: Dispiegamento Dinamico

I restanti passi del processo di installazione rimangono invariati rispetto alla modalità Standard.

CHAPTER 4

Fase di Dispiegamento

Al termine dell'esecuzione dell'utility di installazione vengono prodotti i files necessari per effettuare il dispiegamento nell'ambiente di esercizio. Tali files sono disponibili nella directory *dist*.

Il processo di dispiegamento del software si distingue nei casi di nuova installazione e aggiornamento. Le sezioni seguenti illustrano i due casi.

Nota

Se in fase di esecuzione del wizard di installazione si è scelta la modalità *Avanzata*, in base alle opzioni aggiuntive fornite, potrà essere stato prodotto un numero maggiore di elementi da dispiegare negli ambienti di installazione. Le sezioni seguenti forniranno un dettaglio a parte per il caso dell'installazione avanzata. Tali indicazioni potranno essere ignorate in caso di installazione standard.

4.1 Nuova Installazione

Per completare il processo di installazione si devono effettuare i passi seguenti:

1. Creare un utente sul RDBMS avente i medesimi valori di username e password indicati in fase di setup.
2. Creare un database, per ospitare le tabelle dell'applicazione, avente il nome indicato durante la fase di setup. Il charset da utilizzare è UTF-8, la collation deve essere case sensitive.
3. Impostare i permessi di accesso in modo che l'utente creato al passo 1 abbia i diritti di lettura/scrittura sul database creato al *passo 2*. Si può consultare un esempio relativo a questi primi 3 passi, riferito alla piattaforma PostgreSQL, in sezione *Esempio di setup del database PostgreSQL*.
4. Eseguire lo script *sql/GovWay.sql* per la creazione dello schema del database.

   Successivamente eseguire lo script *sql/GovWay_init.sql* per inserire i dati di inizializzazione del database.

   Ad esempio, nel caso di PostgreSQL, si potranno eseguire i comandi:

   - *psql <hostname> <username> -f sql/GovWay.sql*
   - *psql <hostname> <username> -f sql/GovWay_init.sql*

Nota

In caso di installazione *Avanzata*:

- Gli script *sql/GovWay.sql* e *sql/GovWay_init.sql* saranno eseguiti limitatamente al database del runtime di Govway.
- Se presente lo script *sql/GovWayConfigurazione.sql*, deve essere eseguito per la creazione dello schema database dedicato alle configurazioni. Successivamente eseguire lo script *GovWayConfigurazione_init.sql* per l'inserimento dei dati di inizializzazione del database.
- Se presente lo script *sql/GovWayStatistiche.sql*, deve essere eseguito per la creazione dello schema database dedicato alle statistiche. Successivamente eseguire lo script *GovWayStatistiche_init.sql* per l'inserimento dei dati di inizializzazione del database.
- Se presente lo script *sql/GovWayTracciamento.sql*, deve essere eseguito per la creazione dello schema database dedicato alle tracce. Successivamente eseguire lo script *GovWayTracciamento_init.sql* per l'inserimento dei dati di inizializzazione del database.

5. Installare il DriverJDBC, relativo al tipo di RDBMS indicato in fase di setup, nella directory:
   - *<WILDFLY_HOME>/standalone/deployments*, nel caso di Wildfly.
   - *<TOMCAT_HOME>/lib*, nel caso di Tomcat.
6. Per le connessioni al database è necessario configurare i seguenti datasource impostati con i parametri forniti durante l'esecuzione dell'utility di installazione:
   - Il gateway necessita di un datasource con nome JNDI:
     - *org.govway.datasource*
   - Le console grafiche necessitano di un datasource con nome JNDI:
     - *org.govway.datasource.console*
   - Nel caso si sia richiesto il supporto al protocollo eDelivery, è necessario un terzo datasource con nome JNDI:
     - *org.govway.datasource.console.domibus*

   I datasource, preconfigurati per l'Application Server indicato, sono disponibili nella directory datasource e contengono le configurazioni di accesso al database indicate (ip, db_name, utenza, password). Tali files possono essere utilizzati come riferimento per la definizione dei datasource richiesti nelle modalità disponibili per l'Application Server scelto. Tali files possono anche essere utilizzati direttamente per un rapido dispiegamento copiandoli nelle seguenti posizioni nel file system:
   - *<WILDFLY_HOME>/standalone/deployments*, nel caso di Wildfly.
   - *<TOMCAT_HOME>/conf/Catalina/localhost*, nel caso di Tomcat

   Utilizzando i file preconfigurati, su WildFly è necessario sostituire al loro interno il placeholder *NOME_DRIVER_JDBC.jar* con il nome del driver JDBC installato in precedenza.

Nota

In caso di installazione *Avanzata*, in base alle scelte effettuate per differenziare gli schemi database saranno disponibili nella directory *dist/datasource* tutte le definizioni richieste. In particolare, se previsto, potranno essere presenti:

- *org.govway.datasource.tracciamento*, per l'accesso al database dedicato alle tracce
- *org.govway.datasource.statistiche*, per l'accesso al database dedicato alle statistiche

Se inoltre è stato indicato un ambiente dedicato per *Configurazione e Monitoraggio», quindi distinto dal runtime, dentro la directory *dist/datasource* saranno presenti le due directory:

- *runtime*, contenente i datasource da dispiegare nell'ambiente dedicato al runtime di Govway
- *manager*, contenente i datasource da dispiegare nell'ambiente dedicato alle console di configurazione e monitoraggio

7. Eseguire il dispiegamento delle applicazioni presenti nella directory *archivi* secondo le modalità disponibili per l'Application Server scelto. Per un rapido dispiegamento è possibile copiare gli archivi nelle seguenti posizioni nel file system:
   - *<WILDFLY_HOME>/standalone/deployments*, nel caso di Wildfly.
   - *<TOMCAT_HOME>/webapps*, nel caso di Tomcat

   Nota

   In caso di installazione *Avanzata*, se è stata indicata la scelta di un ambiente dedicato per Configurazione e Monitoraggio, la directory *dist/archivi* conterrà due subdirectory:
   - *runtime*, contenente gli archivi applicativi da dispiegare nell'ambiente dedicato al runtime di Govway
   - *manager*, contenente gli archivi applicativi da dispiegare nell'ambiente dedicato alle console di configurazione e monitoraggio

8. Verificare che la directory di lavoro di GovWay, fornita con le informazioni preliminari dell'utility di installazione, esista o altrimenti crearla con permessi tali da consentire la scrittura all'utente di esecuzione dell'application server
9. Copiare nella directory di lavoro tutti i files di configurazioni presenti nella directory *cfg*. Ad esempio con il comando:
   - *cp cfg/*.properties /etc/govway/*

   La directory di destinazione deve essere accessibile in lettura all'utente con cui si esegue l'Application Server.

   Nota

   In caso di installazione *Avanzata*, se è stata indicata la scelta di un ambiente dedicato per Configurazione e Monitoraggio, la directory *dist/cfg* conterrà due subdirectory:
   - *runtime*, contenente i file di configurazione da copiare nella directory di lavoro dell'ambiente dedicato al runtime di Govway
   - *manager*, contenente i file di configurazione da copiare nella directory di lavoro dell'ambiente dedicato alle console di configurazione e monitoraggio

10. La fase di predisposizione del software è stata completata. Si raccomanda di attivare la "*Cifratura delle Informazioni Confidenziali*".
11. Avviare l'application server con il relativo service oppure utilizzando la linea di comando:

- *<WILDFLY_HOME>/bin/standalone.sh*, nel caso di Wildfly.
- *<TOMCAT_HOME>/bin/startup.sh*, nel caso di Tomcat.

Nota

In caso di installazione *Avanzata*, se è stata indicata la scelta *Generazione tramite Applicazione Batch* relativamente all'opzione di *Generazione delle Statistiche*, sarà presente la directory *dist/batch*. Per il dispiegamento del batch fare riferimento alla sezione *Batch Generazione Statistiche*.

4.2 Aggiornamento

Sulla base delle scelte operate sulle maschere del wizard, nella directory *dist* saranno presenti i file necessari per procedere all'aggiornamento richiesto.

L'aggiornamento richiede i seguenti step:

- Fermo dell'Application Server
- *Aggiornamento del database*
- *Aggiornamento dei datasource*
- *Aggiornamento degli archivi applicativi*
- *Aggiornamento dei file di properties*
- *Aggiornamento delle Informazioni confidenziali*
- Riavvio dell'Application Server

4.2.1 Aggiornamento del database

Nella sottodirectory *sql* si trovano gli script SQL da eseguire sul database attualmente utilizzato per adeguarlo alla nuova versione:

1. Eseguire lo script *sql/GovWay_upgrade_<new-version>.sql* per aggiornare lo schema del database.
2. Se sono stati selezionati nuovi profili di interoperabilità rispetto alla precedente installazione, devono essere eseguiti anche gli script:
   - *sql/profili/GovWay_upgrade_initialize-profilo-<new-profile>.sql*
3. Se si è modificata la tipologia di Application Server rispetto a quella utilizzata nell'installazione precedente (es. da jboss a tomcat), deve anche essere eseguito lo script:
   - *sql/utilities/as/upgradeAS_to<new-type>.sql*

4.2.2 Aggiornamento dei datasource

Nella sottodirectory *datasource* si trovano le configurazioni dei datasource automaticamente generate in base ai dati di connessione al database forniti e all'Application Server indicato. Se non è stato modificato l'Application Server, rispetto a quello utilizzato nell'attuale installazione, un aggiornamento dei datasource non è necessario. Eventualmente confrontare i dati di configurazione dei datasource generati dall'installer con i dati degli attuali datasource presenti nell'Application Server per verificare che non esistano differenze. Se necessario aggiornare questi elementi, procedere come indicato per questa attività nella Sezione *Nuova Installazione*.

4.2.3 Aggiornamento degli archivi applicativi

Eseguire il dispiegamento delle applicazioni presenti nella sottodirectory *archivi* secondo le modalità disponibili per l'Application Server scelto. Per un rapido dispiegamento è possibile copiare gli archivi nelle seguenti posizioni del file system:

- wildfly: WILDFLY_HOME/standalone/deployments/
- tomcat: TOMCAT_HOME/webapps/

4.2.4 Aggiornamento dei file di properties

Nella sottodirectory *cfg* si trovano i template dei file di properties esterni. Questi file, durante l'installazione del prodotto, sono già stati copiati nella directory di lavoro di GovWay. Tali file di properties hanno lo scopo di fornire all'utente dei file pre-confezionati, con proprietà commentate, da utilizzare rapidamente secondo quanto descritto nei manuali d'utilizzo del prodotto, per modificare eventuali configurazioni built-in. Non è quindi indispensabile che tali file vengano riportati sull'installazione precedente e soprattutto occorre fare attenzione a non sovrascrivere eventualmente i precedenti, se erano stati modificati rispetto al template iniziale (generato dall'installer).

Nota

Nella sottodirectory *cfg/utilities/diff* vengono riportate solamente le modifiche attuate sui file, rispetto alla versione precedente, nel formalismo «diff» (estensione .diff) o il file intero (estensione .properties) se si tratta di un file che non esisteva nella precedente versione

4.2.5 Aggiornamento delle Informazioni confidenziali

Si raccomanda di attivare la "*Cifratura delle Informazioni Confidenziali*" qualora non fosse stata attivata nella versione precedente.

È possibile proteggere le informazioni attualmente presenti in chiaro nella base dati utilizzando il tool descritto nella sezione "*Aggiornamento delle Informazioni confidenziali*".

4.3 Batch Generazione Statistiche

Tra le opzioni dell'installazione avanzata esiste l'opzione che consente di mantenere il controllo diretto sulla generazione delle statistiche tramite una procedura batch. Selezionando l'opzione di *Generazione tramite Applicazione Batch* relativamente alla voce *Generazione delle Statistiche*, il processo di installazione crea la directory *dist/batch*. L'esecuzione della procedura deve essere schedulata ad intervalli regolari ad esempio utilizzando un cron.

Dentro la directory *dist/batch* troviamo le seguenti subdirectory:

- *generatoreStatistiche*, che comprende il batch
- *crond*, che comprende un esempio di utilizzo del batch agganciato a un cron

La directory *dist/batch/generatoreStatistiche* contiene la seguente struttura:

- directory *lib*, che comprende le librerie per l'esecuzione del batch
- directory *jdbc*, che deve essere popolata con il driver JDBC adeguato alla piattaforma database adottata
- directory *properties*, che comprende i file di configurazione. Tra questi troviamo il file *daoFactory.properties* dove sono presenti i dati per la connessione al database
- script shell per l'esecuzione dei batch che campionano le informazioni statistiche su differenti intervalli: orario o giornaliero.

CHAPTER 5

Verifica dell'Installazione

Appena conclusa la fase di dispiegamento si può procedere con l'avvio dell'application server, quindi:

1. Verificare che la *govwayConsole*, l'applicazione web per la gestione di GovWay, sia accessibile tramite browser all'indirizzo: *http://<hostname-pdd>/govwayConsole*. In caso di corretto funzionamento verrà visualizzata la schermata seguente:

Figure5.1: Verifica Installazione: govwayConsole

2. Accedere alla govwayConsole utilizzando le credenziali fornite durante l'esecuzione dell'installer.

3. Verificare che la *govwayMonitor*, l'applicazione web per il monitoraggio di GovWay, sia accessibile tramite browser all'indirizzo: *http://<hostname-pdd>/govwayMonitor*. In caso di corretto funzionamento verrà visualizzata la schermata seguente:

Figure5.2: Verifica Installazione: govwayMonitor

4. Accedere alla govwayMonitor utilizzando le credenziali fornite durante l'esecuzione dell'installer.
5. Se durante l'esecuzione dell'Installer è stato indicato di generare il servizio che consente la configurazione tramite API REST, in caso di corretto funzionamento sarà possibile scaricare l'interfaccia OpenAPI v3. L'interfaccia nel formato yaml sarà disponibile all'indirizzo:

   - *http://<hostname-pdd>/govwayAPIConfig/openapi.yaml*

   L'interfaccia nel formato json sarà disponibile all'indirizzo:

   - *http://<hostname-pdd>/govwayAPIConfig/openapi.json*

6. Se durante l'esecuzione dell'Installer è stato indicato di generare il servizio che consente il monitoraggio tramite API REST, in caso di corretto funzionamento sarà possibile scaricare l'interfaccia OpenAPI v3. L'interfaccia nel formato yaml sarà disponibile all'indirizzo:

   - *http://<hostname-pdd>/govwayAPIMonitor/openapi.yaml*

   L'interfaccia nel formato json sarà disponibile all'indirizzo:

   - *http://<hostname-pdd>/govwayAPIMonitor/openapi.json*

CHAPTER 6

Finalizzazione dell'Installazione

Terminati i passi descritti nelle precedenti sezioni, GovWay è pienamente operativo e può essere utilizzato per proteggere le proprie API. Il prodotto viene dispiegato con dei parametri di configurazione che possiedono dei valori di default relativamente alle seguenti tematiche:

1. *GovWay Config Map*

   La mappa "*govway.map.properties*" consente di definire una serie di variabili Java che possono essere riferite da tutte le proprietà presenti nei vari file di configurazione e dalle configurazioni attivate tramite la console di gestione, come descritto nella sezione *GovWay Config Map*.

2. *Cifratura delle informazioni confidenziali*

   GovWay supporta la gestione delle informazioni confidenziali salvate su database e delle chiavi/keystore presenti su filesystem attraverso la cifratura/decifratura mediante una master key, utilizzando un approccio HYOK (Hold Your Own Key) o BYOK (Bring Your Own Key).

   Con l'approccio HYOK, le operazioni di cifratura e decifratura avvengono utilizzando una master key presente in un keystore locale (risiedente su filesystem) o all'interno di un HSM come descritto nella sezione *Registrazione Token PKCS11*.

   In alternativa, è possibile adottare una gestione BYOK, dove la master key è depositata su un servizio cloud. In questo caso, le operazioni di wrap-key e unwrap-key delle informazioni confidenziali vengono gestite tramite chiamate API esposte dal servizio cloud.

   Maggiori indicazioni a riguardo sono presenti nella sezione *Cifratura delle informazioni confidenziali (HYOK/BYOK)*.

3. *GovWay Secrets*

   La mappa "*govway.secrets.properties*" consente di definire una serie di variabili Java, in maniera simile a *GovWay Config Map*, con la differenza che i valori saranno forniti cifrati e GovWay si occuperà di decifrarli prima del loro caricamento nel sistema. Maggiori dettagli vengono forniti nella sezione *GovWay Secrets*.

4. *URL di Invocazione*

   Per conoscere l'url di invocazione di una API protetta da GovWay è possibile accedere al dettaglio di una erogazione o fruizione tramite la govwayConsole. L'url fornita avrà un prefisso *http://localhost:8080/govway*.

Se il gateway è stato dispiegato in modo da essere raggiungibile tramite un host, porta o contesto differente è possibile configurare tale prefisso seguendo le indicazioni descritte nella sezione *Url di Invocazione*.

5. *Multi-Tenant*

   I processi di configurazione e monitoraggio attuabili tramite le console sono ottimizzati nell'ottica di gestire sempre un unico dominio rappresentato da un soggetto interno il cui nome è stato fornito durante l'esecuzione dell'installer (Fig. 3.7).

   Per estendere l'ambito delle configurazioni e del monitoraggio tramite console a più di un soggetto interno al dominio seguire le indicazioni presenti nella sezione *Multi-Tenant*.

6. *Modalità di Tracciamento*

   GovWay registra le informazioni relative alle comunicazioni gestite attraverso un sistema di tracciamento configurabile su database e/o su file. È importante valutare attentamente le modalità di tracciamento da attivare in base alle esigenze di monitoraggio e alle risorse disponibili.

   La configurazione di default prevede il tracciamento su database nella fase finale «*Risposta consegnata*», con attivazione automatica di un processo di failover su filesystem in caso di indisponibilità del database.

   Avvertimento

   Il processo di failover richiede adeguato spazio disco nella directory configurata (default: */var/govway/resources*) per serializzare temporaneamente le tracce in attesa di essere riversate nel database.

   Per maggiori dettagli sulle fasi di tracciamento, sul tracciamento su database con failover e sul tracciamento su file seguire le indicazioni presenti nella sezione *Modalità di Tracciamento*.

7. *Gestione CORS*

   Nella configurazione di default di GovWay è abilitata la gestione del *cross-origin HTTP request (CORS)*; è possibile modificarne la configurazione seguendo le indicazioni presenti nella sezione *Gestione CORS*.

8. *Rate Limiting*

   GovWay permette definire un rate limiting sulle singole erogazioni o fruizioni di API. Le metriche utilizzabili riguardano il numero di richieste all'interno di un intervallo temporale, l'occupazione di banda, il tempo di risposta etc.

   In presenza di una installazione con più nodi gateway attivi, GowWay per default effettua il conteggio delle metriche utilizzate dalle policy di rate limiting indipendentemente su ogni singolo nodo del cluster. Questa soluzione è certamente la più efficiente, ma presenta dei limiti evidenti se è necessaria una contabilità precisa del numero di accessi consentiti globalmente da parte dell'intero cluster. In tali situazioni è necessario modificare la configurazioni di default per attivare modalità di conteggio distribuite le quali richiedono alcune configurazioni del sistema descritte nella sezione *Configurazione di Hazelcast*.

   Oltre al rate limiting GovWay consente di fissare un numero limite complessivo, indipendente dalle APIs, riguardo alle richieste gestibili simultaneamente dal gateway, bloccando le richieste in eccesso.

   Per default GovWay è configurato per gestire simultaneamente al massimo 200 richieste. Per modificare tale configurazione seguire le indicazioni presenti nella sezione *Numero Complessivo Richieste Simultanee*.

   Sempre a livello globale, GovWay limita la dimensione massima accettata di una richiesta e di una risposta a 10MB. Per modificare i livelli di soglia della policy seguire le indicazioni presenti nella sezione *Dimensione Massima dei Messaggi*.

9. *Tempi Risposta*

GovWay è preconfigurato con dei parametri di timeout per quanto concerne la gestione delle connessioni verso gli applicativi interni (erogazioni) o esterni (fruizioni) dal dominio di gestione. Per effettuare un tuning di tali parametri seguire le indicazioni descritte nella sezione *Tempi Risposta*.

10. *Caching della Risposta - Disk Cache*

    In GovWay è possibile abilitare il salvataggio delle risposte in una cache sia globalmente, in modo che sia attivo per tutte le APIs, che singolarmente sulla singola erogazione o fruizione. Questa funzionalità permette ad un backend server di non dover riprocessare le stesse richieste più volte.

    La configurazione di default prevede di salvare in una cache, che risiede in memoria RAM, fino a 5.000 risposte (ogni risposta comporta il salvataggio di due elementi in cache). In caso venga superato il numero massimo di elementi che possano risiedere in cache, vengono eliminate le risposte meno recenti secondo una politica *LRU*.

    GovWay consente di personalizzare la configurazione della cache in modo da aggiungere una memoria secondaria dove salvare gli elementi in eccesso. Per abilitare la memoria secondaria seguire le indicazioni descritte nella sezione *Caching della Risposta - Disk Cache*.

11. *Device PKCS11*

    Il Cryptographic Token Interface Standard, PKCS#11, definisce interfacce di programmazione native per token crittografici, come acceleratori crittografici hardware e smartcard. Per consentire a GovWay di accedere ai token PKCS#11 nativi è necessario configurare correttamente il provider PKCS#11 e registrarlo tra i token conosciuti da GovWay seguendo le indicazioni descritte nella sezione *Registrazione Token PKCS11*.

12. *Online Certificate Status Protocol (OCSP)*

    Il protocollo Online Certificate Status Protocol (OCSP), descritto nel RFC 2560, consente di verificare la validità di un certificato senza ricorrere alle liste di revoca dei certificati (CRL). Le modalità di verifica possono differire per svariati motivi: dove reperire la url del servizio OCSP a cui richiedere la validità del certificato e il certificato dell'Issuer che lo ha emesso, come comportarsi se un servizio non è disponibile, eventuale validazione CRL alternativa etc. GovWay consente di definire differenti policy OCSP che saranno disponibili per la scelta in fase di configurazione di una funzionalità che richiede la validazione di un certificato; in ognuna delle policy sarà possibile configurare modalità di verifica dei certificati differenti descritte nella sezione *Online Certificate Status Protocol (OCSP)*.

13. *API di Configurazione e Monitoraggio*

    GovWay fornisce sia una console che dei servizi che espongono API REST per la sua configurazione e per il monitoraggio. L'installer genera per default le console mentre i servizi devono essere selezionati puntualmente dall'utente (Fig. 3.6).

    Gli indirizzi per accedere alle console sono già stati forniti nella fase di *Verifica dell'Installazione*.

    Nel caso invece siano stati generati i servizi, gli indirizzi base per utilizzarli sono:

    - *http://<hostname-pdd>/govway/ENTE/api-config/v1/*
    - *http://<hostname-pdd>/govway/ENTE/api-monitor/v1/*

    ma deve essere completata la configurazione del Controllo degli Accessi per poterli invocare correttamente seguendo le indicazioni descritte nella sezione *API di Configurazione e Monitoraggio*.

14. *Load Balancing*

    Il prodotto è preconfigurato per funzionare su di una singola istanza. Per realizzare un'installazione in load balancing seguire le indicazioni descritte nella sezione *Configurazione in Load Balancing*.

15. *Configurazione HTTPS*

    GovWay processa ogni richiesta in una duplice veste agendo sia da server al momento della ricezione della richiesta che da client al momento di inoltrare la richiesta verso i backend applicativi.

In entrambi i ruoli la configurazione varia a seconda dell'architettura in cui è stato dispiegato GovWay (es. presenza di un Web Server). Indicazioni sulla configurazione vengono fornite nella sezione *Configurazione HTTPS*.

16. *Integrazione delle Console con IdM esterno*

    Nella sezione *Integrazione delle Console con IdM esterno* vengono fornite indicazioni su come sia possibile delegare l'autenticazione delle utenze ad un IdM esterno da cui ottenere il principal dell'utenza.

17. *Richieste "application/x-www-form-urlencoded" su WildFly*

    Per poter gestire correttamente richieste con Content-Type "application/x-www-form-urlencoded" su application server "WildFly", è richiesto di abilitare l'attributo "allow-non-standard-wrappers" nella configurazione dell'A.S. Indicazioni sulla configurazione vengono fornite nella sezione *Richieste "application/x-www-form-urlencoded" su WildFly*.

18. *ApplicationSecurityDomain "other" su WildFly 25 o superiore*

    A partire dalla versione 25 di wildfly, nella configurazione di default è abilitato un application-security-domain "other" che rende obbligatoria la presenza di credenziali valide per invocare i contesti di GovWay. Questa configurazione deve essere disabilitata come indicato nella sezione *ApplicationSecurityDomain "other" su WildFly 25 o superiore*.

19. *Cache*

    In GovWay è possibile abilitare l'utilizzo di cache che mantengono i dati di configurazioni acceduti, i keystore e i certificati, il risultato dei processi di validazione, autenticazione, autorizzazione e altri aspetti minori. Ogni funzionalità è associata ad una cache dedicata i cui parametri sono configurabili come indicato nella sezione *Cache*.

20. *Esposizione di Informazioni*

    È possibile adottare alcune misure di sicurezza per limitare l'esposizione di informazioni relative all'architettura e alle tecnologie utilizzate. Indicazioni su come configurare l'application server e altri componenti sono fornite nella sezione *Esposizione di Informazioni*.

6.1 GovWay Config Map

All'interno del file *<directory-lavoro>/govway.map.properties* è possibile razionalizzare una serie di variabili Java che potranno essere riferite in qualsiasi file di proprietà di GovWay presente nella *<directory-lavoro>* tramite la sintassi "${nomeVar}".

Le variabili potranno inoltre essere accedute all'interno delle varie configurazioni di GovWay come descritto, ad esempio, nella sezione valoriDinamici, tramite la sintassi "java:NAME" o "envj:NAME".

La sintassi da utilizzare all'interno del file *<directory-lavoro>/govway.map.properties* viene descritta nella sezione *Configurazione delle variabili*.

6.1.1 Configurazione delle variabili

All'interno del file *<directory-lavoro>/govway.map.properties* è possibile razionalizzare una serie di variabili Java tramite la seguente sintassi:

```
# Consente di definire proprietà java tramite la sintassi:
java.<nome>=<valore>
```

È possibile verificare le variabili Java, sia quelle definite nel file *govway.map.properties* che quelle presenti originariamente nel sistema, tramite due modalità:

- tutte le variabili Java vengono registrate da GovWay nel file di log *<directory-log>/govway_configurazioneSistema.log*;
- lo stesso file è inoltre generabile dinamicamente accedendo alla sezione "*Strumenti > Runtime*" (strumenti_runtime), tramite la voce *Download*.

I valori delle variabili potrebbero contenere informazioni confidenziali (es. password) che non devono finire in chiaro all'interno dei log sopra indicati. Per attivarne l'offuscamento è possibile utilizzare i seguenti costrutti:

```
# Per offuscare variabili java usare la sintassi:
obfuscated.java.keys=<nome1>,<nome2>,...,<nomeN>
```

Vengono supportate diverse modalità di offuscamento, attivabili assegnando uno dei seguenti valori alla proprietà "obfuscated.mode":

- digest (default): viene calcolato il digest del valore rispetto all'algoritmo indicato nella proprietà "obfuscated.digest" (default: SHA-256);
- static: viene utilizzato staticamente il valore indicato nella proprietà "obfuscated.static" (default: **);
- none: non viene attuato alcun offuscamento.

Di seguito, la sintassi da utilizzare nel file *govway.map.properties*:

```
# Modalità utilizzata per offuscare
obfuscated.mode=digest
#obfuscated.digest=SHA-256
#obfuscated.static=******
```

6.2 Cifratura delle informazioni confidenziali (HYOK/BYOK)

GovWay supporta la gestione delle informazioni confidenziali salvate su database e delle chiavi/keystore presenti su filesystem attraverso la cifratura/decifratura mediante una master key, utilizzando un approccio HYOK (Hold Your Own Key) o BYOK (Bring Your Own Key).

Con l'approccio HYOK, le operazioni di cifratura e decifratura avvengono utilizzando una master key presente in un keystore locale (risiedente su filesystem) o all'interno di un HSM come descritto nella sezione *Registrazione Token PKCS11*.

In alternativa, è possibile adottare una gestione BYOK, dove la master key è depositata su un servizio remoto (es. in cloud). In questo caso, le operazioni di wrap-key e unwrap-key delle informazioni confidenziali vengono gestite tramite chiamate API esposte dal servizio remoto.

La gestione delle informazioni confidenziali è attivabile all'interno del file *<directory-lavoro>/byok.properties*, nel quale sono presenti le due seguenti sezioni principali:

- "Key Management Service" (KMS): ogni KMS definisce i criteri di cifratura (wrap) o di decifratura (unwrap) di un'informazione sensibile o di una chiave/keystore; la sintassi viene descritta nella sezione *Key Management Service*. Un KMS può riferirsi a una chiave gestita localmente tramite un approccio HYOK (*KMS Locale*) o gestita da remoto su un servizio cloud come previsto dall'approccio BYOK (*KMS Remoto*).
- "Security Engine": l'associazione di un KMS che consente la cifratura di un'informazione confidenziale (wrap) e un altro che ne consente la decifratura (unwrap) viene registrata sotto la definizione di un security engine, come descritto nella sezione *Security Engine*.

I KMS e i security engine vengono utilizzati da GovWay per gestire diversi aspetti:

- "Security Engine di default": l'attivazione di uno specifico security engine comporta il suo utilizzo da parte di GovWay per la cifratura e la decifratura delle informazioni confidenziali salvate su database (*Cifratura delle Informazioni Confidenziali*);

- "Secrets Map": nel file *<directory-lavoro>/govway.secrets.properties* è possibile definire delle variabili Java o delle variabili di sistema con valori cifrati tramite uno dei security engine o dei KMS definiti; tali variabili potranno essere riferite nelle varie configurazioni di GovWay e saranno decifrate al momento del loro utilizzo (per maggiori dettagli si rimanda alla sezione *GovWay Secrets*);
- "Unwrap delle chiavi/keystore": le chiavi e i keystore indicati nei connettori HTTPS e nelle funzionalità di message security possono riferirsi a path su filesystem relativi a file cifrati, la cui decodifica avviene utilizzando un KMS di "unwrap" tra quelli disponibili. La figura Fig. 6.1 mostra un esempio di utilizzo di una policy BYOK necessaria per decodificare il keystore cifrato riferito contenente la chiave e il certificato client da utilizzare in un connettore HTTPS.

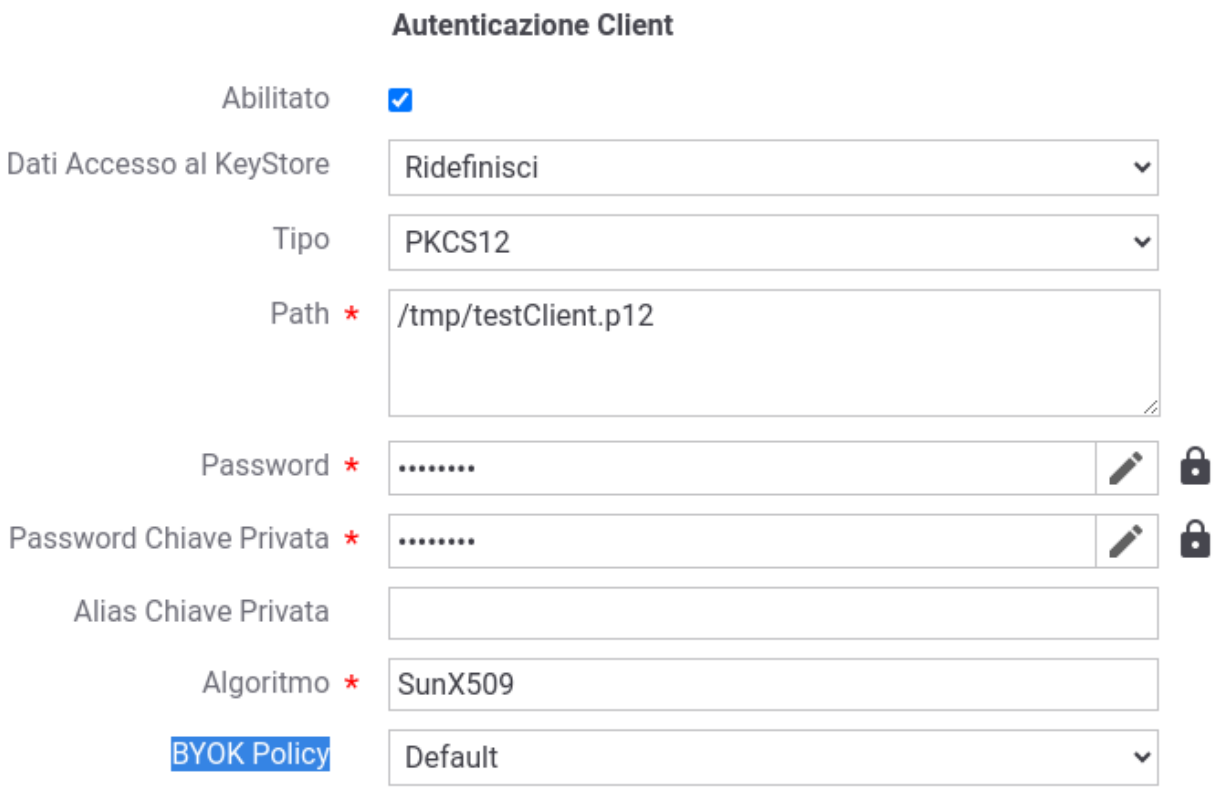

Figure6.1: Esempio di configurazione di una policy BYOK su GovWay

- "Unwrap delle chiavi/keystore per la funzionalità di FileTrace": nella sezione avanzate_fileTrace_proprietaCifrate vengono fornite le indicazioni per cifrare eventuali parti di log; i keystore o le chiavi utilizzate per la cifratura possono a loro volta essere cifrate e dovranno essere decifrate prima del loro utilizzo attraverso un KMS riferito tramite la direttiva *encrypt.encMode1.kms*.
- "Tool govway-vault-cli": al termine dell'esecuzione dell'installer (*Esecuzione dell'Installer*), nella directory *dist/tools/govway-vault-cli* è presente un tool da linea di comando che consente:
  - la cifratura o decifratura di chiavi, riferendosi a un security engine o un KMS definito nel file *<directory-lavoro>/byok.properties*;
  - l'aggiornamento di una base dati esistente, consentendo di cifrare le informazioni confidenziali precedentemente salvate in chiaro o di aggiornarle attraverso l'utilizzo di una differente master key.

  Maggiori dettagli vengono forniti nella sezione *GovWay Vault CLI*.

6.2.1 Key Management Service

In questa sezione viene descritta la sintassi da utilizzare per definire i KMS funzionali ad operazioni di cifratura (wrap) o di decifratura (unwrap) di un'informazione sensibile o di una chiave/keystore.

GovWay consente di definire molteplici policy KMS agendo sul file *<directory-lavoro>/byok.properties*.

La configurazione di ogni KMS è rappresentata dall'insieme di proprietà che presentano lo stesso prefisso "kms.<idKms>". L'identificativo <idKms> serve univocamente ad aggregare tali proprietà.

Nota

Nelle versioni precedenti alla 3.3.16.p1 veniva utilizzata la keyword "ksm" al posto di "kms". Anche se ancora supportata, l'utilizzo di "ksm" è deprecato e sarà rimosso nelle versioni future. Si raccomanda pertanto di aggiornare i file di configurazione utilizzando la nuova keyword "kms".

Un KMS viene identificato univocamente da GovWay tramite la proprietà "kms.<idKms>.type"; il suo valore deve essere univoco rispetto ai valori forniti negli altri KMS per la medesima proprietà.

Ogni KMS verrà indirizzato tramite il suo identificativo su molteplici aspetti:

- l'associazione ad un security engine, come descritto nella sezione *Security Engine*;
- la decodifica dei valori indicati nel file *<directory-lavoro>/govway.secrets.properties*, come descritto nella sezione *GovWay Secrets*;
- la cifratura o decifratura di chiavi e informazioni gestite tramite il tool "govway-vault-cli" descritto nella sezione *GovWay Vault CLI*;
- la decodifica di chiavi e keystore cifrati, riferiti nei connettori HTTPS e nelle funzionalità di message security, attraverso la selezione di un KMS tra quelli resi disponibili nella scelta della "BYOK Policy" nella console di gestione (govwayConsole) come mostrato nella figura Fig. 6.1.

Quando la selezione di un KMS avviene tramite una console di gestione, non viene visualizzato all'utente l'identificativo ma bensì una label che deve essere definita nella proprietà "kms.<idKms>.label"; di conseguenza anche questo valore deve essere univoco rispetto a quello fornito per gli altri KMS.

Di seguito un esempio di KMS basato sulla cifratura di un'informazione tramite la derivazione di una chiave conforme al comando "openssl aes-256-cbc -pbkdf2 -k encryptionPassword -a":

```
# Esempio di KMS Wrap per una cifratura basata su chiave derivata da una password␣
↪configurata all'interno del security engine 'openssl-pbkdf2'
kms.govway-openssl-pbkdf2-wrap.label=OpenSSL 'pbkdf2' Wrap Example
kms.govway-openssl-pbkdf2-wrap.type=openssl-pbkdf2-wrap
kms.govway-openssl-pbkdf2-wrap.mode=wrap
kms.govway-openssl-pbkdf2-wrap.encryptionMode=local
kms.govway-openssl-pbkdf2-wrap.input.param1.name=encryptionPassword
kms.govway-openssl-pbkdf2-wrap.input.param1.label=Encryption Password
kms.govway-openssl-pbkdf2-wrap.local.impl=openssl
kms.govway-openssl-pbkdf2-wrap.local.keystore.type=pass
kms.govway-openssl-pbkdf2-wrap.local.password=${kms:encryptionPassword}
kms.govway-openssl-pbkdf2-wrap.local.password.type=openssl-pbkdf2-aes-256-cbc
kms.govway-openssl-pbkdf2-wrap.local.encoding=base64
```

Di seguito vengono descritte tutte le opzioni di configurazione utilizzabili nella registrazione di un KMS all'interno del file *<directory-lavoro>/byok.properties*:

- kms.<idKms>.type: identificativo univoco;

- kms.<idKms>.label: etichetta associata al kms e visualizzata nelle maschere di configurazione (deve essere a sua volta univoca);
- kms.<idKms>.mode: [wrap|unwrap] indica se il KMS è utilizzabile per effettuare la cifratura (wrap) o la decifratura (unwrap);
- kms.<idKms>.encryptionMode: [optional; default:remote] una operazione di wrap/unwrap può essere realizzata invocando un kms remoto (via http) o cifrando/decifrando tramite keystore locali.

Nelle sezioni seguenti verranno fornite le opzioni di configurazione specifica di una operazione di wrap/unwrap a seconda della modalità indicata nella proprietà "kms.<idKms>.encryptionMode":

- "local" (*KMS Locale*): utilizzato per implementare un approccio "HYOK" le operazioni di cifratura e decifratura avvengono utilizzando una master key presente in un keystore locale (risiedente su filesystem) o all'interno di un HSM;
- "remote" (*KMS Remoto*): utilizzato per un approccio "BYOK" dove la master key è depositata su un servizio remoto (es. in cloud).

In entrambe le configurazioni (locale o remota) sarà possibile riferire dei parametri descritti nella sezione *Parametri di un KMS*

KMS Locale

In questa sezione viene descritta la sintassi da utilizzare per definire i KMS funzionali ad operazioni di cifratura (wrap) o di decifratura (unwrap) dove la master key è presente in un keystore locale (risiedente su filesystem) o all'interno di un HSM;

Di seguito un esempio di KMS basato sulla cifratura di un'informazione tramite una chiave pubblica

```
# Esempio di KMS Wrap per una cifratura basata su chiave pubblica
kms.govway-async-keys-wrap.label=GovWay AsyncKeys Wrap Example
kms.govway-async-keys-wrap.type=async-keys-wrap
kms.govway-async-keys-wrap.mode=wrap
kms.govway-async-keys-wrap.encryptionMode=local
kms.govway-async-keys-wrap.local.impl=java
kms.govway-async-keys-wrap.local.keystore.type=public
kms.govway-async-keys-wrap.local.key.path=/etc/govway/keys/keyPair-test.rsa.publicKey.pem
kms.govway-async-keys-wrap.local.key.algorithm=RSA/ECB/OAEPWithSHA-256AndMGF1Padding
kms.govway-async-keys-wrap.local.key.wrap=true
kms.govway-async-keys-wrap.local.algorithm=AES/CBC/PKCS5Padding
kms.govway-async-keys-wrap.local.encoding=base64
```

Il corrispettivo esempio di KMS necessario a decifrare l'informazione tramite la chiave privata associata.

```
# Esempio di KMS Unwrap per una cifratura basata su chiave asincrona
kms.govway-async-keys-unwrap.label=GovWay AsyncKeys Unwrap Example
kms.govway-async-keys-unwrap.type=async-keys-unwrap
kms.govway-async-keys-unwrap.mode=unwrap
kms.govway-async-keys-unwrap.encryptionMode=local
kms.govway-async-keys-unwrap.local.impl=java
kms.govway-async-keys-unwrap.local.keystore.type=keys
kms.govway-async-keys-unwrap.local.key.path=/etc/govway/keys/keyPair-test.rsa.pkcs8_
↪encrypted.privateKey.pem
kms.govway-async-keys-unwrap.local.key.password=${envj:read(GOVWAY_PRIVATE_KEY_PASSWORD)}
kms.govway-async-keys-unwrap.local.publicKey.path=/etc/govway/keys/keyPair-test.rsa.
↪publicKey.pem
```

(continues on next page)

(continua dalla pagina precedente)

```
kms.govway-async-keys-unwrap.local.key.algorithm=RSA/ECB/OAEPWithSHA-256AndMGF1Padding
kms.govway-async-keys-unwrap.local.key.wrap=true
kms.govway-async-keys-unwrap.local.algorithm=AES/CBC/PKCS5Padding
kms.govway-async-keys-unwrap.local.encoding=base64
```

L'operazione viene descritta da un insieme di direttive definite tramite la sintassi:

- "*kms.<idKms>.local.<direttiva>*"

Di seguito vengono fornite tutte le direttive supportate:

- *impl* [required]: indica il tipo di token cifrato prodotto o atteso per essere decifrato:
  - *jose*: un token JSON Web Encryption (JWE) conforme al RFC 7516;
  - *java*: viene utilizzata la classe Cipher fornita dal package javax.crypto per cifrare i dati che poi verranno serializzati su file di log a seconda della direttiva "*encoding*" fornita;
  - *openssl*: viene prodotto un cipher text, attraverso una chiave derivata da una password, che può essere decifrato utilizzando i comandi di encryption "openssl"; richiede un keystore di tipo "pass".
- *encoding* [required; mode=java|openssl]: indica il tipo di codifica utilizzato per la rappresentazione dei dati cifrati:
  - *base64*: rappresentazione base64;
  - *hex*: rappresentazione esadecimale;
- *keystore.type* [required]: indica il tipo di keystore utilizzato dove è presente la chiave di cifratura da utilizzare:
  - *symm*: indica l'utilizzo di una chiave simmetrica fornita attraverso tramite la direttiva:
    - *key.inline*: chiave simmetrica (es. Chiave AES dovà essere di 16, 24 o 32 byte);
    - *key.path*: [ignorata se presente "key.inline"] path su filesystem ad una chiave simmetrica (es. Chiave AES dovà essere di 16, 24 o 32 byte);
    - *key.encoding*: [optional; base64/hex] consente di indicare la codifica della chiave;
  - *pass*: indica la generazione di una chiave derivata da una password attraverso le seguenti direttive (non utilizzabile con la modalità "jose"):
    - *password*: la password utilizzata per derivare la chiave;
    - *password.type*: [opzionale; default=openssl-pbkdf2-aes-256-cbc] consente di selezionare l'algoritmo di derivazione tra le seguenti opzioni disponibili:
      - openssl-aes-256-cbc
      - openssl-pbkdf2-aes-128-cbc
      - openssl-pbkdf2-aes-192-cbc
      - openssl-pbkdf2-aes-256-cbc
    - *password.iter*: [optional] consente di indicare il numero di iterazioni durante la derivazione della chiave con l'algoritmo pbkdf2.
  - *jceks*: indica l'utilizzo di una chiave simmetrica presente in un keystore java di tipo JCEKS indirizzato tramite le seguenti direttive:
    - *keystore.path*: path su filesystem del keystore;
    - *keystore.password*: password del keystore;
    - *key.alias*: alias che identifica la chiave simmetrica nel keystore;

    - *key.password*: password della chiave simmetrica;
- *public*: indica l'utilizzo di una chiave pubblica asimmetrica fornita attraverso le seguenti direttive:
    - *key.inline*: chiave pubblica asimmetrica in formato PEM o DER (sono supportati sia i formati pkcs1 che pkcs8);
    - *key.path*: [ignorata se presente "key.inline"] path su filesystem ad una chiave pubblica asimmetrica in formato PEM o DER (sono supportati sia i formati pkcs1 che pkcs8);
    - *key.encoding*: [optional; base64/hex] consente di indicare la codifica della chiave;
    - *key.wrap* [optional; mode=java; boolean true/false]: indicazione se la chiave pubblica debba essere utilizzata per cifrare direttamente i dati (key.wrap=false) o per cifrare una chiave simmetrica AES generata dinamicamente (key.wrap=true);

> **Nota**
>
> La modalità "key.wrap=false" è utilizzabile solamente con informazioni da cifrare «sufficientemente corte» rispetto alla capacità di cifratura della chiave RSA altrimenti si avrà un errore simile al seguente: «too much data for RSA block».

- *keys*: indica l'utilizzo di chiavi asincrone fornita attraverso le seguenti direttive:
    - *key.inline*: chiave privata in formato PEM o DER (sono supportati sia i formati pkcs1 che pkcs8);
    - *key.path*: [ignorata se presente "key.inline"] path su filesystem ad una chiave privata in formato PEM o DER (sono supportati sia i formati pkcs1 che pkcs8);
    - *key.password*: password della chiave privata;
    - *key.encoding*: [optional; base64/hex] consente di indicare la codifica della chiave privata;
    - *key.wrap* [optional; mode=java; boolean true/false]: indicazione se la chiave pubblica è stata utilizzata per cifrare direttamente i dati (key.wrap=false) o per cifrare una chiave simmetrica AES generata dinamicamente (key.wrap=true);
    - *publicKey.inline*: chiave pubblica in formato PEM o DER (sono supportati sia i formati pkcs1 che pkcs8);
    - *publicKey.path*: [ignorata se presente "publicKey.inline"] path su filesystem ad una chiave pubblica in formato PEM o DER (sono supportati sia i formati pkcs1 che pkcs8);
    - *publicKey.encoding*: [optional; base64/hex] consente di indicare la codifica della chiave pubblica;
- *jwk*: indica l'utilizzo di keystore JWK che può contenere una chiave simmetrica o una chiave pubblica asimmetrica; le direttive supportate sono le seguenti:
    - *keystore.path*: path su filesystem del keystore;
    - *key.alias*: alias che identifica la chiave nel keystore;
    - *key.wrap* [optional; mode=java; boolean true/false]: indicazione se la chiave pubblica debba essere utilizzata per cifrare direttamente i dati (key.wrap=false) o per cifrare una chiave simmetrica AES generata dinamicamente (key.wrap=true);
- *jks* o *pkcs12*: indica l'utilizzo di un certificato presente in un keystore java di tipo JKS o PKCS12 indirizzato tramite le seguenti direttive:
    - *keystore.path*: path su filesystem del keystore;
    - *keystore.password*: password del keystore;

        * *key.alias*: alias che identifica il certificato nel keystore;
        * *key.wrap* [optional; mode=java; boolean true/false]: indicazione se il certificato debba essere utilizzato per cifrare direttamente i dati (key.wrap=false) o per cifrare una chiave simmetrica AES generata dinamicamente (key.wrap=true);
    - <*tipoRegistratoPKCS11*>: indica l'utilizzo di uno dei tipi di keystore PKCS11 registrati ("pkcs11") all'interno del quale è presente il certificato da utilizzare indicato tramite la direttiva:
        * *key.alias*: alias che identifica il certificato nel keystore;
        * *key.wrap* [optional; mode=java; boolean true/false]: indicazione se il certificato debba essere utilizzato per cifrare direttamente i dati (key.wrap=false) o per cifrare una chiave simmetrica AES generata dinamicamente (key.wrap=true);
- *key.algorithm* [required]: specifica l'algoritmo utilizzato per generare o gestire le chiavi crittografiche utilizzate durante il processo di cifratura;
- *algorithm* [required]: specifica l'algoritmo utilizzato per cifrare effettivamente i dati;
- *include.key.id* [optional; mode=jose; boolean true/false]: indicazione se inserire nell'header del token JWE (claim "kid") l'alias della chiave utilizzata per la cifratura;
- *key.id* [optional; mode=jose]: indica il nome della chiave che verrà inserito nel claim "kid" presente nell'header del token JWE;
- *include.cert* [optional; mode=jose; boolean true/false]: indicazione se inserire nell'header del token JWE (claim "x5c") il certificato utilizzato per la cifratura;
- *include.cert.sha1* [optional; mode=jose; boolean true/false]: indicazione se inserire nell'header del token JWE (claim "x5t") il digest SHA-1 del certificato utilizzato per la cifratura;
- *include.cert.sha256* [optional; mode=jose; boolean true/false]: indicazione se inserire nell'header del token JWE (claim "x5t#256") il digest SHA-256 del certificato utilizzato per la cifratura;
- *include.public.key* [optional; mode=jose; boolean true/false]: indicazione se inserire nell'header del token JWE (claim "jwk") la chiave pubblica utilizzata per la cifratura.

Rappresentazione dei dati cifrati con mode=java

Come descritto in precedenza indicando la modalità "java" nella direttiva "mode" viene utilizzata la classe Cipher fornita dal package javax.crypto per cifrare i dati che poi verranno serializzati su file di log a seconda della direttiva "*encoding*" fornita: base64 o hex.

In funzione del tipo di chiave (simmetrica o asimmetrica) e della direttiva key.wrap la rappresentazione dei dati cifrati conterrà più parti che devono essere considerate per poter effettuare l'operazione inversa di decifratura:

- *chiave simmetrica*: il dato cifrato è formato da due parti, separate tramite un punto, entrambe codificate in base64 o hex a seconda dell'encoding selezionato; la prima parte rappresenta il Vettore di Inizializzazione (IV) mentre la seconda sono i dati cifrati:
    - <IV>.<DatiCifrati>
- *chiave pubblica asimmetrica con direttiva key.wrap=true*: il dato cifrato è formato da tre parti, separate tramite un punto, entrambe codificate in base64 o hex a seconda dell'encoding selezionato; la prima parte rappresenta la chiave AES generata dinamicamente e cifrata con la chiave pubblica (wrap), la seconda parte il Vettore di Inizializzazione (IV) della cifratura simmetrica e la terza parte sono i dati cifrati con la chiave simmetrica:
    - <WRAP_KEY>.<IV>.<DatiCifrati>
- *chiave pubblica asimmetrica con direttiva key.wrap=false*: è presente solo una parte contenente i dati cifrati con la chiave pubblica asimmetrica:
    - <DatiCifrati>

KMS Remoto

In questa sezione viene descritta la sintassi da utilizzare per definire i KMS funzionali ad operazioni di cifratura (wrap) o di decifratura (unwrap) dove la master key è depositata su un servizio remoto (es. in cloud) e le operazioni di wrap e unwrap sono rese disponibili dal servizio remoto tramite chiamate di API.

Di seguito un esempio di KMS basato sulla chiamata dell'operazione wrap in cui l'informazione da cifrare viene fornita nel payload http come json payload e l'informazione cifrata viene ritornata all'interno di una risposta json nel claim "value".

```
# Esempio di KMS Wrap
kms.govway-remote-wrap.label=GovWay Remote Wrap Example
kms.govway-remote-wrap.type=govway-remote-wrap
kms.govway-remote-wrap.mode=wrap
kms.govway-remote-wrap.encryptionMode=remote
kms.govway-remote-wrap.http.endpoint=https://vault.example/keys/wrapkey?api-version=1.0
kms.govway-remote-wrap.http.method=POST
kms.govway-remote-wrap.http.payload.inline={"alg": "RSA1_5","value": "${kms-key}"}
kms.govway-remote-wrap.http.response.jsonPath=$.value
```

Un esempio di KMS basato sulla chiamata dell'operazione unwrap in cui l'informazione da decifrare viene fornita come parametro "key" della query url, e l'informazione decifrata viene tornata nel payload http codificata in base64. Viene inoltre attivata una gestione personalizzata del protocollo https dove viene indicato un truststore custom per effettuare l'autenticazione server.

```
# Esempio di KMS Unwrap
kms.govway-remote-unwrap.label=GovWay Remote Unwrap Example
kms.govway-remote-unwrap.type=govway-remote-unwrap
kms.govway-remote-unwrap.mode=unwrap
kms.govway-remote-unwrap.encryptionMode=remote
kms.govway-remote-unwrap.http.endpoint=https://vault.example/keys/unwrapkey?api-
↪version=1.0&key=${kms-urlencoded-key}
kms.govway-remote-unwrap.http.method=GET
kms.govway-remote-unwrap.http.username=test
kms.govway-remote-unwrap.http.password=changeme
kms.govway-remote-unwrap.http.response.base64Encoded=true
kms.govway-remote-unwrap.https=true
kms.govway-remote-unwrap.https.hostnameVerifier=true
kms.govway-remote-unwrap.https.serverAuth=true
kms.govway-remote-unwrap.https.serverAuth.trustStore.path=/tmp/test.jks
kms.govway-remote-unwrap.https.serverAuth.trustStore.type=jks
kms.govway-remote-unwrap.https.serverAuth.trustStore.password=123456
```

L'operazione viene descritta da un insieme di direttive definite tramite la sintassi:

- "*kms.<idKms>.http.<direttiva>*"

Di seguito vengono fornite tutte le direttive supportate per la gestione della richiesta http:

- *endpoint* [required]: definisce l'endpoint del kms;
- *method* [required]: definisce il metodo HTTP utilzzato per connettersi al kms;
- *header.<nome>* [optional]: consente di definire un header http con nome "<name>" valorizzato con il valore indicato nella proprietà;
- *payload.inline*: [optional] payload inviato nella richiesta http;

- *payload.path*: [optional; ignorata se presente "http.payload.inline"] path su filesystem relativo al contenuto da inviare nella richiesta http;
- *payload.username* e *http.payload.password*: [optional] definiscono la credenziale http-basic;
- *connectionTimeout*: [optional; int] tempo massimo in millisecondi di attesa per stabilire una connessione con il server;
- *readTimeout*: [optional; int] tempo massimo in millisecondi di attesa per la ricezione di una risposta dal server.

L'informazione cifrata o decifrata viene attesa per default nel payload della risposta http. È possibile configurare comportamenti differenti tramite le seguenti proprietà:

- *response.base64Encoded*: [optional; boolean] indicazione se sarà attesa una risposta codificata in base64;
- *response.hexEncoded*: [optional; boolean] indicazione se sarà attesa una risposta codificata tramite una rappresentazione esadecimale;
- *response.jsonPath*: [optional] se la risposta è un json (eventualmente dopo la decodificata base64/hex) consente di indicare un jsonPath per estrarre l'informazione da un singolo elemento.
- *response.jsonPath.base64Encoded*: [optional; boolean] indicazione se sarà atteso un valore, estratto tramite jsonPath, codificato in base64;
- *response.jsonPath.hexEncoded*: [optional; boolean] indicazione se sarà atteso un valore, estratto tramite jsonPath, codificato tramite una rappresentazione esadecimale;

Inoltre se l'endpoint contattato è su protocollo https, può essere attivata una gestione personalizzata dell'autenticazione server e/o client definendo la seguente proprietà:

- kms.<idKMS>.https=true

Tutte le configurazioni relative al protocollo https possono essere fornite utilizzando le seguenti ulteriori direttive definibili tramite la sintassi:

- "*kms.<idKms>.https.<direttivaHttps>*"

Di seguito vengono fornite tutte le direttive https supportate:

- *hostnameVerifier*: [optional; boolean] indica se deve essere verificato l'hostname rispetto al certificato server;
- *serverAuth*: [optional; boolean] indica se deve essere effettuata l'autenticazione del certificato server; nel caso venga abilitata la verifica possono essere forniti i seguenti parametri:
  - *serverAuth.trustStore.path*: path su filesystem al truststore;
  - *serverAuth.trustStore.type*: tipo del truststore;
  - *serverAuth.trustStore.password*: password del truststore;
  - *serverAuth.trustStore.crls*: path su filesystem delle CRL;
  - *serverAuth.trustStore.ocspPolicy*: identificativo di una OCSP Policy (*Online Certificate Status Protocol (OCSP)*);
- *clientAuth*: [optional; boolean] indica se deve essere inviato un certificato client; nel caso venga abilitata l'autenticazione client possono essere forniti i seguenti parametri:
  - *clientAuth.keyStore.path*: path su filesystem al keystore;
  - *clientAuth.keyStore.type*: tipo di keystore;
  - *clientAuth.keyStore.password*: password per l'accesso al keystore;
  - *clientAuth.key.alias*: alias che identifica il certificato client nel keystore;
  - *clientAuth.key.password*: password della chiave privata.

Parametri di un KMS

Nelle sezioni *KMS Locale* e *KMS Remoto* è stata descritta la sintassi da utilizzare per attuare operazioni di cifratura/decifratura utilizzando rispettivamente una master key presente in un keystore locale o remoto.

Tutte le configurazioni descritte nelle precedenti sezioni possono riferire dei parametri risolti a runtime durante l'utilizzo del KMS tramite la sintassi "${kms:<nomeparametro>}". I parametri riferiti dovranno essere definiti in un kms attraverso le seguenti proprietà:

- kms.<idKms>.input.<idParam>.name: [optional] identificativo univoco del parametro
- kms.<idKms>.input.<idParam>.label: [optional] definisce l'etichetta associata al parametro di configurazione

Tutti i parametri dovranno essere valorizzati ogni volta che viene riferito un KMS come mostrato ad esempio nella sezione *Security Engine*, durante l'associazione ad un security engine tramite le proprietà "security.<identificativo>.kms.param.<nomeParametro>", o nella sezione *Configurazione delle variabili cifrate*, attraverso le proprietà "kms.<identificativoKMS>.param.<nomeParametro>".

Oltre ai parametri esistono le seguenti variabili speciali che possono essere utilizzate all'interno delle configurazioni di un KMS per riferire l'informazione confidenziale stessa per la quale si sta operando la cifratura o la decifratura:

- ${kms-key}: bytes dell'informazione confidenziale;
- ${kms-urlencoded-key}: bytes dell'informazione confidenziale codificata per essere utilizzata in una url;
- ${kms-base64-key}: bytes dell'informazione confidenziale codificata in base64;
- ${kms-hex-key}: bytes dell'informazione confidenziale codificata in una rappresentazione esadecimale;
- ${kms-base64-urlencoded-key}: bytes dell'informazione confidenziale codificata in base64 e successivamente codificata per poter essere utilizzata in una url;
- ${kms-hex-urlencoded-key}: bytes dell'informazione confidenziale codificata in una rappresentazione esadecimale e successivamente codificata per poter essere utilizzata in una url.

Di seguito un esempio di KMS che definisce dei parametri e li utilizza per definire la modalità di invocazione del kms remoto.

```
kms.govway-example.label=GovWay Example
kms.govway-example.type=govway-example
kms.govway-example.mode=unwrap
kms.govway-example.encryptionMode=remote
kms.govway-example.input.param1.name=endpointUnwrap
kms.govway-example.input.param1.label=Endpoint Unwrap Key
kms.govway-example.input.param2.name=username
kms.govway-example.input.param2.label=Username
kms.govway-example.input.param3.name=password
kms.govway-example.input.param3.label=Password
kms.govway-example.http.endpoint=${kms:endpointUnwrap}&paramValue=${kms-urlencoded-key}
kms.govway-example.http.method=GET
kms.govway-example.http.username=${kms:username}
kms.govway-example.http.password=${kms:password}
```

6.2.2 Security Engine

Nella sezione *Key Management Service* è stata documentata la sintassi necessaria a registrare KMS che consentono la cifratura di un'informazione confidenziale (wrap) o la decifratura (unwrap).

L'associazione di un KMS per l'operazione "wrap" e un altro per l'operazione "unwrap" consente di definire un *Security Engine*.

Di seguito un esempio in cui viene valorizzato il parametro richiesto da entrambi i KMS riferiti:

```
# Esempio di cifratura basata su chiave derivata da una password attesa come variabile␣
↪di sistema o della JVM
security.gw-pbkdf2.kms.wrap=openssl-pbkdf2-wrap
security.gw-pbkdf2.kms.unwrap=openssl-pbkdf2-unwrap
security.gw-pbkdf2.kms.param.encryptionPassword=esempio
```

La configurazione di ogni *Security Engine* è rappresentata dall'insieme di proprietà che presentano lo stesso prefisso "security.<idSecurityEngine>". L'identificativo <idSecurityEngine>, a differenza della configurazione descritta nella sezione *Key Management Service*, serve sia ad aggregare tali proprietà che ad identificarlo univocamente.

Di seguito vengono descritte tutte le opzioni di configurazione utilizzabili nella registrazione di un *Security Engine* all'interno del file *<directory-lavoro>/byok.properties*:

- *security.<idSecurityEngine>.kms.wrap* [required]: identificativo del KMS da utilizzare per l'operazione di cifratura (wrap);
- *security.<idSecurityEngine>.kms.unwrap* [required]: identificativo del KMS da utilizzare per l'operazione di decifratura (unwrap);
- *security.<idSecurityEngine>.kms.param.<paramName>* [optional]: come descritto nella sezione *Parametri di un KMS* ogni KMS può richiedere dei parametri di input che possono essere forniti all'interno della definizione del *Security Engine* tramite la direttiva "*kms.param.<paramName>*".

6.2.3 Cifratura delle Informazioni Confidenziali

L'attivazione di un *Security Engine* (*Security Engine*) comporta che venga utilizzato da parte di GovWay per la cifratura e la decifratura delle informazioni confidenziali salvate su database (descritte nella sezione console_informazioni_confidenziali_info) e per le console_informazioni_confidenziali_proprieta.

Per attivare la cifratura deve essere abilitata la proprietà *govway.security* nel file *<directory-lavoro>/byok.properties* valorizzandola con l'identificativo di un (*Security Engine*).

Di seguito un esempio in cui viene attivata la cifratura tramite il Security Engine "gw-pbkdf2":

```
# Security engine di default utilizzato da GovWay
govway.security=gw-pbkdf2
```

6.2.4 GovWay Vault CLI

Al termine dell'esecuzione dell'installer (*Esecuzione dell'Installer*), nella directory *dist/tools/govway-vault-cli* è presente un tool da linea di comando che consente:

- la cifratura o decifratura di informazioni o chiavi, come descritto nella sezione *Encrypt / Decrypt*;
- l'aggiornamento di una base dati esistente, consentendo di cifrare le informazioni confidenziali precedentemente salvate in chiaro o di aggiornarle attraverso l'utilizzo di una differente master key, come descritto nella sezione *Aggiornamento delle Informazioni confidenziali*.

Encrypt / Decrypt

Il tool *govway-vault-cli* consente la cifratura o decifratura di chiavi, riferendosi a un security engine o un KMS definito nel file *<directory-lavoro>/byok.properties*.

I comandi da utilizzare sono *encrypt.sh* per la cifratura e *decrypt.sh* per la decifratura.

Se invocato senza parametri, il tool visualizza nell'errore restituito gli argomenti attesi che dovranno essere forniti con l'invocazione:

```
encrypt -system_in|-file_in=text|path -system_out|-file_out=path [-sec|-kms=id]
```

Gli argomenti prevedono:

- il contenuto da elaborare, fornito tramite una delle due seguenti modalità:
  - *-system_in=TEXT*: consente di fornire il testo da elaborare direttamente a linea di comando;
  - *-file_in=PATH*: consente di indicare il path assoluto contenente il contenuto da elaborare;
- dove serializzare il contenuto elaborato (cifrato o decifrato):
  - *-system_out*: il contenuto viene visualizzato come output dell'esecuzione del tool a linea di comando;
  - *-file_out=PATH*: consente di indicare il path assoluto dove verrà salvato il contenuto elaborato;
- il riferimento a un security engine o un KMS definito nel file *<directory-lavoro>/byok.properties*:
  - *-sec=id*: identificativo di un *Security Engine*;
  - *-kms=id*: identificativo di un *Key Management Service*.

Di seguito un esempio che assume l'esistenza di un security engine "gw-pbkdf2" configurato per attuare una cifratura attraverso la derivazione di una chiave conforme al comando "openssl aes-256-cbc -pbkdf2 -k encryptionPassword -a".

```
[govway-vault-cli]$ cat Prova.txt
!!Hello World!!

[govway-vault-cli]$ ./encrypt.sh -file_in=Prova.txt -file_out=Prova.txt.enc -sec=gw-
↪pbkdf2
Encrypted content in 'Prova.txt.enc'

[govway-vault-cli]$ cat Prova.txt.enc
==gw-pbkdf2==.U2FsdGVkX1/0Qvn5Uzg8yb7xezxhSks2+buxg+/6al+Ltk+Kzo6iuP5BSiiatjFB

[govway-vault-cli]$ ./decrypt.sh -file_in=Prova.txt.enc -file_out=Prova.txt.decoded -
↪sec=gw-pbkdf2
Decrypted content in 'Prova.txt.decoded'

[govway-vault-cli]$ cat Prova.txt.decoded
!!Hello World!!
```

Aggiornamento delle Informazioni confidenziali

Il tool *govway-vault-cli* può essere utilizzato sulla base dati esistente sia per cifrare le informazioni confidenziali precedentemente salvate in chiaro, sia per aggiornarle utilizzando una differente master key.

Nota

Si consiglia di effettuare un backup della base dati prima di procedere con l'aggiornamento delle informazioni confidenziali.

Il comando *update.sh*, se invocato senza parametri, visualizza nell'errore restituito gli argomenti attesi che dovranno essere forniti con l'invocazione:

```
update -sec_in|-plain_in[=id] -sec_out|-plain_out)[=id] [-report=path]
```

Gli argomenti prevedono:

- indicazione sull'attuale cifratura utilizzata:
  - *-plain_in*: indica che le informazioni confidenziali presenti nella base dati risultano attualmente in chiaro;
  - *-sec_in=id*: identificativo del *Security Engine* con cui sono state cifrate le informazioni presenti nella base dati;
- indicazione sulla nuova cifratura che si intende apportare:
  - *-sec_out=id*: identificativo del *Security Engine* con cui si intende cifrare le informazioni presenti nella base dati;
  - *-plain_out*: indica che le informazioni confidenziali presenti nella base dati saranno salvate in chiaro;
- *-report=PATH* [optional]: argomento opzionale che consente di definire un path su file system dove verranno salvate tutte le informazioni modificate dall'operazione di aggiornamento.

6.3 GovWay Secrets

All'interno del file *<directory-lavoro>/govway.secrets.properties* è possibile razionalizzare una serie di variabili Java, in maniera simile a *GovWay Config Map*, con la differenza che i valori saranno forniti cifrati e GovWay si occuperà di decifrarli prima del loro caricamento nel sistema.

Le variabili saranno riferibili in qualsiasi file di proprietà di GovWay presente nella *<directory-lavoro>* tramite la sintassi "${nomeVar}" o all'interno delle varie configurazioni di GovWay, come descritto, ad esempio, nella sezione valoriDinamici, tramite la sintassi "java:NAME" o "envj:NAME".

La cifratura dei valori può essere attuata utilizzando il tool "govway-vault-cli", disponibile all'interno della directory *dist/tools/* prodotta dall'installer (*Esecuzione dell'Installer*), e descritto nella sezione *GovWay Vault CLI* .

La sintassi da utilizzare all'interno del file *<directory-lavoro>/govway.secrets.properties* viene descritta nella sezione *Configurazione delle variabili cifrate*

6.3.1 Configurazione delle variabili cifrate

All'interno del file *<directory-lavoro>/govway.secrets.properties* è possibile razionalizzare una serie di variabili Java tramite la seguente sintassi:

```
# Consente di definire proprietà java tramite la sintassi:
java.<nome>=<valore>
```

Il valore fornito dovrà essere decifrabile tramite uno dei security engine o KMS di "unwrap" disponibili all'interno della configurazione *<directory-lavoro>/byok.properties*, descritti rispettivamente nelle sezioni *Security Engine* e *Key Management Service*.

La modalità di decifratura di default è definibile tramite la proprietà "unwrap.default.mode", che può assumere i seguenti valori:

- security: viene utilizzato il security engine riferito dall'identificativo riportato nella proprietà "unwrap.default.id".
- kms: viene utilizzato il Key Management Service riferito dall'identificativo riportato nella proprietà "unwrap.default.id".

Se la proprietà "unwrap.default.mode" non viene definita, viene utilizzata la modalità security engine.

```
# Indica la modalità di default utilizzata per decifrare i valori forniti rispetto alle␣
↪configurazioni presenti nel file 'byok.properties'.
# security: viene utilizzato il security engine indicato nella proprietà 'unwrap.default.
```

(continues on next page)

(continua dalla pagina precedente)

```
↪id' (per default viene utilizzato il security engine caricato da GovWay)
# kms: viene utilizzato il key management service definito tramite la proprietà 'unwrap.
↪default.id'
unwrap.default.mode=
unwrap.default.id=
```

È possibile verificare che le variabili Java siano state caricate tramite due modalità:

- tutte le variabili java vengono registrate da GovWay nel file di log *<directory-log>/govway_configurazioneSistema.log*;
- lo stesso file è inoltre generabile dinamicamente accedendo alla sezione "*Strumenti > Runtime*" (strumenti_runtime), tramite la voce *Download*.

I valori delle variabili, poiché contengono secrets, non vengono registrati nei log sopra indicati ma viene attuato un offuscamento definito dal valore della proprietà "obfuscated.mode":

- digest (default): viene calcolato il digest del valore rispetto all'algoritmo indicato nella proprietà "obfuscated.digest" (default: SHA-256)
- static: viene utilizzato staticamente il valore indicato nella proprietà "obfuscated.static" (default: **)
- none: non viene attuato alcun offuscamento

Di seguito, la sintassi da utilizzare nel file *govway.secrets.properties*:

```
# Modalità utilizzata per offuscare
obfuscated.mode=digest
#obfuscated.digest=SHA-256
#obfuscated.static=******
```

È inoltre possibile definire una modalità di decifratura differente da quella di default specificando la modalità da utilizzare per la singola variabile tramite il prefisso "java.security." o il prefisso "java.kms.", a seconda che si voglia utilizzare rispettivamente un security engine o un KMS per la decodifica.

Di seguito un esempio di utilizzo di un security engine differente:

```
java.<nomeVariabile>=<valoreCifratoTramiteSecurityEngineIdX>
java.security.<nomeVariabile>=<identificativoSecurityEngineIdX>
```

Di seguito un esempio di utilizzo di un kms:

```
java.<nomeVariabile>=<valoreCifratoTramiteKMSIdX>
java.kms.<nomeVariabile>=<identificativoKMSIdX>
```

Come descritto nella sezione *Parametri di un KMS* ogni KMS può richiedere dei parametri di input. Tali parametri possono essere forniti nel file *govway.secrets.properties* tramite la seguente sintassi:

```
# Per un kms è possibile configurare i parametri richiesti tramite la seguente sintassi:
kms.<identificativoKMS>.param.<nomeParametro>=<valoreParametro>
```

È infine possibile definire all'interno del file *govway.secrets.properties* delle variabili i cui valori non sono cifrati, registrandole con la seguente modalità:

```
java.<nomeVariabile>=<valoreInChiaro>
java.wrapped.<nomeVariabile>=false
```

6.4 Url di Invocazione

Per scoprire quale sia la url di una API protetta da GovWay da fornire ai client esterni, il gestore può utilizzare la govwayConsole, la quale fornisce nella visualizzazione del dettaglio di una erogazione o fruizione di API la url di invocazione (es. Fig. 6.2).

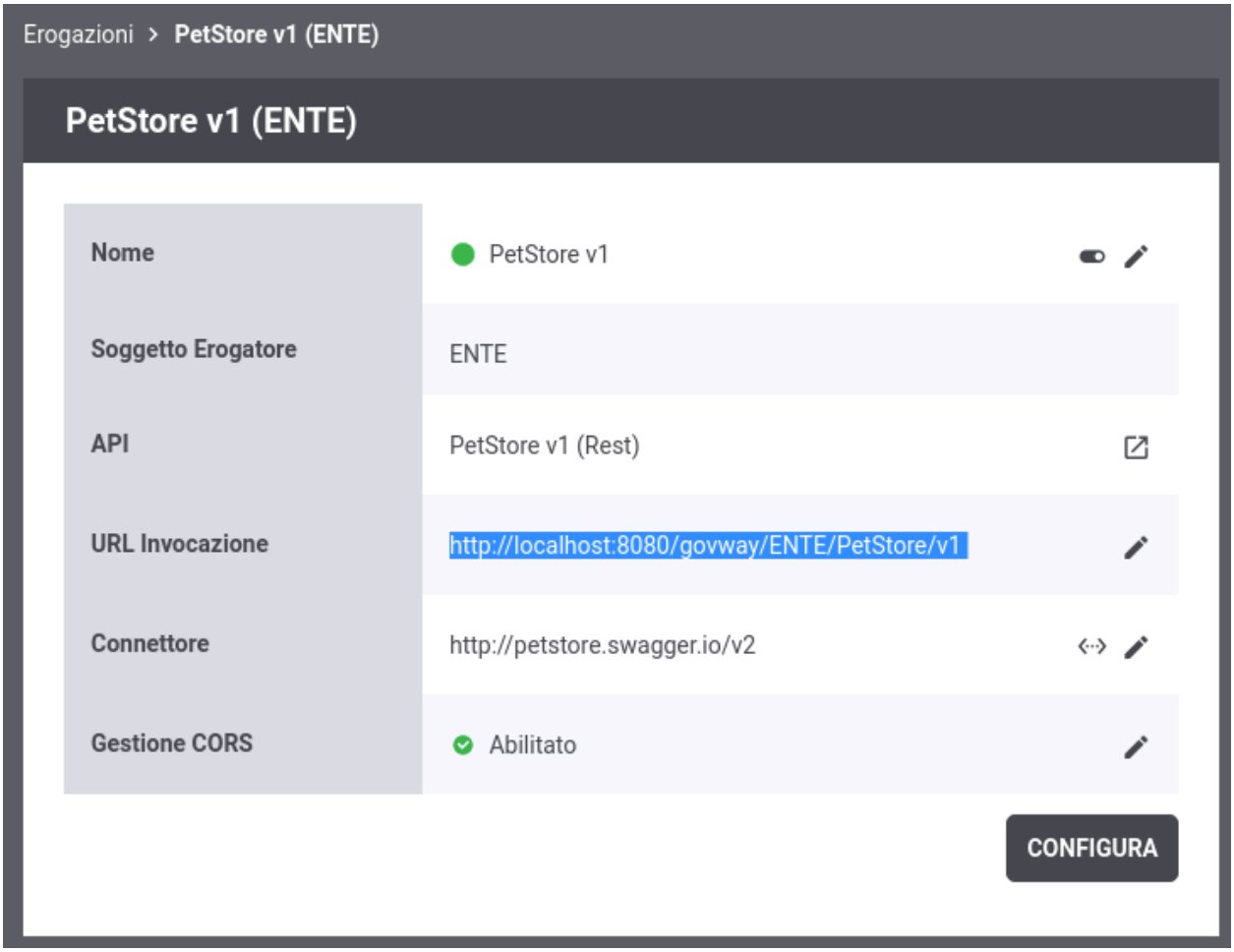

Figure6.2: Url di Invocazione di una Erogazione

L'url fornita ha per default un prefisso *http://localhost:8080/govway* che può non andar bene se il gateway è stato dispiegato in modo da essere raggiungibile tramite un host, porta o contesto differente.

Per modificare i prefissi delle url di invocazioni accedere alla voce *"Configurazione - Generale"* del menù (sezione configGenerale_urlInvocazione). Nella sezione *"URL di Invocazione API"* è possibile configurare i prefissi di una erogazione e di una fruizione. Inoltre in presenza di un reverse proxy che media le comunicazioni http con GovWay, è anche possibile configurare opportunamente le url di invocazione delle API esposte da GovWay allineandole con le eventuali configurazioni specifiche realizzate sul reverse proxy.

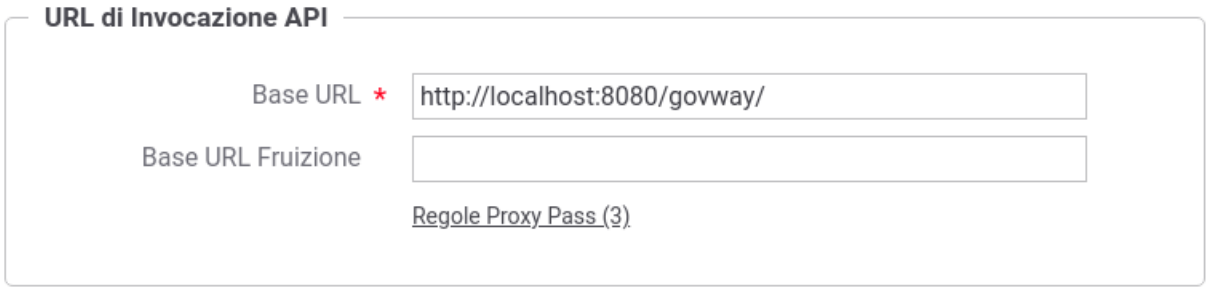

Figure6.3: Configurazione prefissi per le Url di Invocazione

6.5 Multi-Tenant

I processi di configurazione e monitoraggio attuabili tramite le console sono ottimizzati nell'ottica di gestire sempre un unico dominio rappresentato da un soggetto interno il cui nome è stato fornito durante l'esecuzione dell'installer (Fig. 3.7). In tal senso, le fruizioni e le erogazioni si intendono sempre in soggettiva riguardo un singolo soggetto interno amministrato dall'utente in sessione.

La funzionalità Multi-Tenant è un'opzione che consente di estendere l'ambito delle configurazioni prodotte dalla govwayConsole a più di un soggetto interno al dominio. Tale opzione si attiva accedendo alla voce *"Configurazione - Generale"* del menù, sezione *"Multi-Tenant"*.

Figure6.4: Abilitazione Multi-Tenant

Una volta abilitato accedere alla voce *"Soggetti"* del menù e selezionare il pulsante *"Aggiungi"* per registrare un nuovo soggetto interno (nuovo dominio).

Terminata la registrazione del nuovo soggetto sia nella console di gestione (*govwayConsole*) che nella console di monitoraggio (*govwayMonitor*) prima di procedere con qualsiasi operazione è adesso possibile selezionare il soggetto per cui si intende gestire il dominio attraverso l'apposito menù situato in alto a destra nell'intestazione delle console.

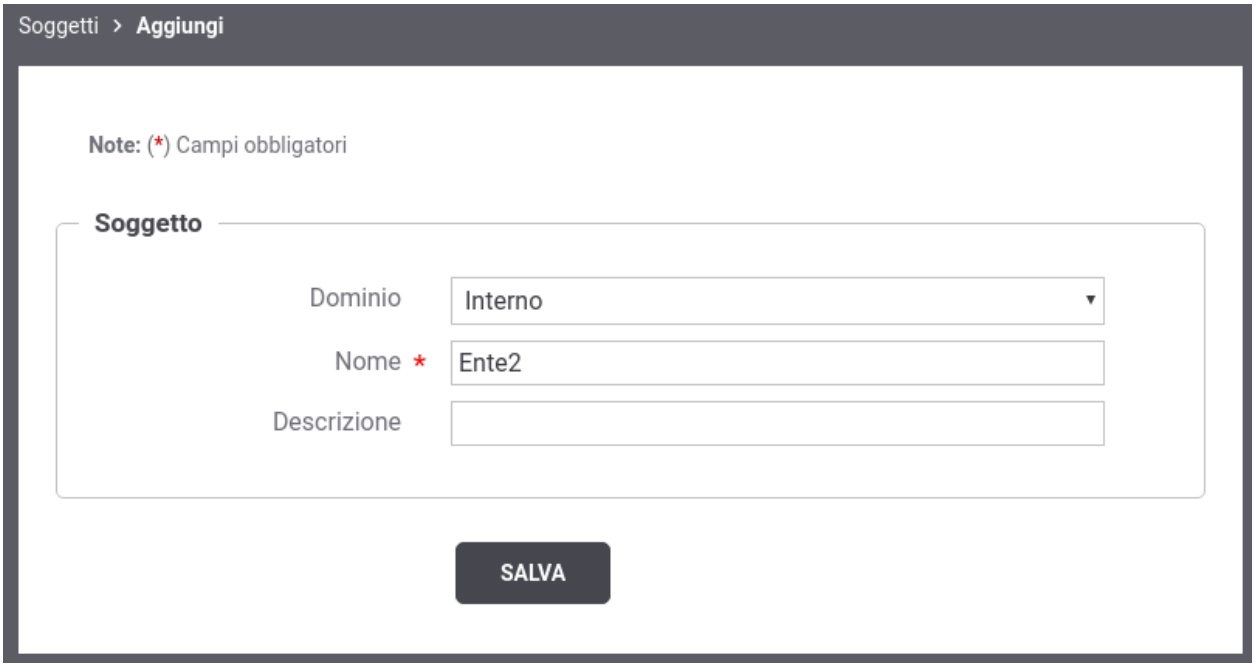

Figure6.5: Registrazione nuovo Soggetto

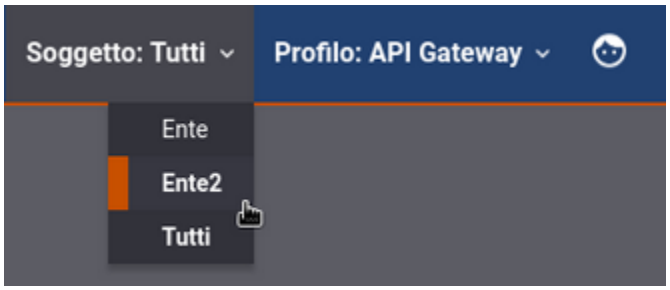

Figure6.6: Selezione del Soggetto

6.6 Modalità di Tracciamento

GovWay registra le informazioni relative alle comunicazioni gestite attraverso un sistema di tracciamento altamente configurabile. Durante la fase di finalizzazione dell'installazione è fondamentale valutare attentamente le modalità di tracciamento da attivare, in quanto impattano sia sulle prestazioni del sistema che sui requisiti di storage.

6.6.1 Configurazione del Tracciamento

GovWay registra le informazioni relative alle comunicazioni gestite attraverso un sistema di tracciamento altamente configurabile. Durante la fase di finalizzazione dell'installazione è fondamentale valutare attentamente le modalità di tracciamento da attivare, in quanto impattano sia sulle prestazioni del sistema che sui requisiti di storage.

Configurazione di Default

Il sistema viene distribuito con una configurazione di default che prevede:

- **Tracciamento su Database**: abilitato
- **Tracciamento su File**: disabilitato
- **Fase di Tracciamento Attiva**: «*Risposta consegnata*» (ultima fase, dopo l'inoltro della risposta al client)
- **Modalità**: non bloccante (gli errori di tracciamento non bloccano la transazione)

Questa configurazione rappresenta un buon compromesso tra completezza delle informazioni registrate e impatto sulle prestazioni, in quanto il tracciamento avviene dopo che la risposta è stata consegnata al client.

Fasi di Tracciamento

GovWay consente di configurare il tracciamento in diverse fasi del processamento di una richiesta, come descritto dettagliatamente nella sezione tracciamentoTransazioniFasi della documentazione della console:

1. **Richiesta ricevuta**: tracciamento prima di iniziare il processamento
2. **Richiesta in consegna**: tracciamento prima di inoltrare la richiesta al backend
3. **Risposta in consegna**: tracciamento prima di inoltrare la risposta al client
4. **Risposta consegnata**: tracciamento dopo aver inoltrato la risposta al client (**default attivo**)

Nota

L'attivazione di fasi di tracciamento anticipate (richiesta ricevuta, richiesta in consegna, risposta in consegna) aumenta la completezza delle informazioni registrate ma può impattare sulle prestazioni e sulla disponibilità del servizio se configurate in modalità bloccante.

Tracciamento su Database

Il tracciamento su database, descritto nella sezione tracciamentoTransazioniDB, è la modalità principale per registrare le transazioni gestite dal gateway. Le informazioni vengono salvate in tabelle dedicate e sono consultabili tramite la console di monitoraggio (mon_intro).

Processo di Failover

Un aspetto critico del tracciamento su database è il **processo di failover** che si attiva automaticamente quando non è possibile registrare le tracce nel database (es. database non disponibile, connessione interrotta, problemi di rete). In questo caso la traccia viene serializzata su filesystem nella directory configurata (default: */var/govway/resources*) e un timer dedicato (default: ogni 5 minuti) tenta di recuperarla e riversarla nel database fino al numero massimo di tentativi (default: 10), dopo il quale viene spostata in una directory "*dlq*" (Dead Letter Queue).

Una volta risolto il problema del database, è possibile recuperare manualmente le tracce dalla DLQ come descritto nella sezione *Recupero Tracce dalla Dead Letter Queue*.

Avvertimento

Requisiti di Spazio Disco per il Failover

Il processo di failover è una soluzione temporanea per gestire l'indisponibilità del database e **richiede adeguato spazio disco**.

Lo spazio necessario dipende da:

- **Volume di traffico**: numero di richieste al secondo gestite dal gateway
- **Dimensione delle tracce**: ogni traccia include metadati, header HTTP, contenuti dei messaggi (payload), informazioni di sicurezza
- **Tempo di indisponibilità del database**: più a lungo il database rimane indisponibile, più tracce si accumulano su disco
- **Numero di istanze**: in un cluster ogni nodo serializza le proprie tracce nella propria directory locale

È **fondamentale** monitorare lo spazio disco della directory di failover e configurare alert per prevenire l'esaurimento dello spazio.

Nota

In ambienti cloud orchestrati (Kubernetes, OpenShift, etc.) è **fortemente consigliato** utilizzare un **volume persistente condiviso** tra i pod (es. NFS, AWS EFS, Azure Files) montato sulla directory */var/log/govway/resources* per implementare efficacemente la funzionalità di failover.

Recupero dalla Dead Letter Queue

Per maggiori dettagli sul recupero manuale delle tracce dalla directory DLQ, consultare la sezione *Recupero Tracce dalla Dead Letter Queue*.

Tracciamento su File

Il tracciamento su file, descritto nella sezione tracciamentoTransazioniFileTrace, consente di registrare le informazioni delle transazioni direttamente su file di log, facilitando l'integrazione con sistemi di log management esterni (FileBeat, Logstash, Fluentd, Kafka, Splunk, ...).

Questa modalità può essere utilizzata:

- **In alternativa al tracciamento su database**: per ridurre il carico sul database o per ambienti che preferiscono soluzioni basate su file
- **In aggiunta al tracciamento su database**: per avere una doppia registrazione delle informazioni

Caratteristiche del Tracciamento su File

- **Nessun failover**: le informazioni vengono scritte direttamente sui file configurati
- **Formato personalizzabile**: è possibile definire il formato, l'ordine e quali informazioni salvare
- **Multi-topic**: le informazioni possono essere suddivise in più file di log
- **Prestazioni**: generalmente più veloce del tracciamento su database

Nota

Anche per il tracciamento su file è importante:

- Dimensionare adeguatamente lo spazio disco
- Configurare la rotazione dei log (es. logrotate)
- Monitorare lo spazio disco disponibile
- Prevedere l'archiviazione o l'invio dei log a sistemi esterni

Accesso alla Configurazione «»»»»»»»»»»»»»»»»»»»»»»»»»»»»»^^^

Per configurare le modalità di tracciamento:

1. Accedere alla console di gestione GovWay (govwayConsole)
2. Navigare in *Configurazione* > *Tracciamento* (tracciamento)
3. Configurare le sezioni:
   - **Transazioni**: per configurare fasi di tracciamento e modalità (database/file)
   - **Messaggi Diagnostici**: per configurare il livello di verbosità dei log

Nota

Le configurazioni effettuate nella console hanno valenza globale. È possibile ridefinire puntualmente il comportamento su ogni singola erogazione/fruizione tramite la voce di configurazione *Tracciamento* del dettaglio API.

Monitoraggio e Manutenzione «»»»»»»»»»»»»»»»»»»»»»»»»»»»»»^^^

Dopo aver configurato il tracciamento, è importante monitorare:

1. **Spazio disco directory failover**: verificare regolarmente l'occupazione di */var/govway/resources*
2. **Directory dlq**: controllare periodicamente se sono presenti tracce che hanno superato il numero massimo di tentativi
3. **Log di GovWay**: verificare la presenza di errori di tracciamento
4. **Prestazioni database**: monitorare il carico sul database di tracciamento
5. **Crescita database**: pianificare politiche di archiviazione e retention
6. **Rotazione log**: verificare che la rotazione dei file di log sia configurata correttamente

Il sistema fornisce metriche e informazioni attraverso i log diagnostici configurabili nella stessa sezione *Configurazione* > *Tracciamento*.

Recupero Tracce dalla Dead Letter Queue

Quando il processo di failover del tracciamento su database non riesce a riversare le tracce nel database dopo il numero massimo di tentativi configurati (default: 10), queste vengono automaticamente spostate in una directory speciale chiamata **Dead Letter Queue (DLQ)**.

Struttura Directory DLQ

La directory DLQ si trova all'interno della directory di failover configurata (default: */var/govway/resources*) con la struttura */var/govway/resources/TIPO/dlq/*, dove TIPO può essere: transazioni, diagnostici, messaggi, o altri tipi di informazioni tracciabili.

Verifica Presenza Tracce in DLQ

Per verificare se sono presenti tracce in DLQ:

```
# Conta tutti i file presenti nelle directory DLQ
find /var/govway/resources/*/dlq/ -type f ! -name "*README" | wc -l

# Lista le directory DLQ con tracce presenti
for dir in /var/govway/resources/*/dlq/; do
    count=$(find "$dir" -type f ! -name "*README" 2>/dev/null | wc -l)
    [ $count -gt 0 ] && echo "$(basename $(dirname $dir)): $count file"
done
```

Recupero Manuale delle Tracce

Una volta risolto il problema che impediva la registrazione nel database, è possibile recuperare manualmente le tracce spostate in DLQ.

Avvertimento

Prima di procedere, verificare che:

- Il database sia nuovamente disponibile e funzionante
- Il timer di failover sia attivo (se presente la proprietà "org.openspcoop2.pdd.resources.fileSystemRecovery.enabled" in govway_local.properties deve essere valorizzata a "true")

Procedura di Recupero

1. Rimuovere i file README da tutte le directory DLQ:

```
find /var/log/govway/resources/*/dlq/ -name "*README" -delete
```

2. Spostare tutte le tracce dalla DLQ alle directory di recovery:

```
for tipo_dir in /var/log/govway/resources/*/; do
    tipo=$(basename "$tipo_dir")
    [ -d "${tipo_dir}dlq" ] && find "${tipo_dir}dlq/*/" -type f -exec mv {} "${tipo_
↪dir}" \; 2>/dev/null
done
```

3. Rimuovere le directory vuote:

```
rmdir /var/log/govway/resources/*/dlq/*/ 2>/dev/null
```

Il timer di failover rileverà i file spostati e tenterà di riversarli nel database al prossimo ciclo (default: ogni 5 minuti).

6.7 Gestione CORS

In GovWay è possibile abilitare la gestione del *cross-origin HTTP request (CORS)* sia globalmente, in modo che sia valida per tutte le APIs, che singolarmente sulla singola erogazione o fruizione.

Nell'installazione di default è abilitata la gestione del CORS globalmente per tutte le API. Tale configurazione è modificabile accedendo alla voce *"Configurazione - Generale"* del menù, sezione *"Gestione CORS"*.

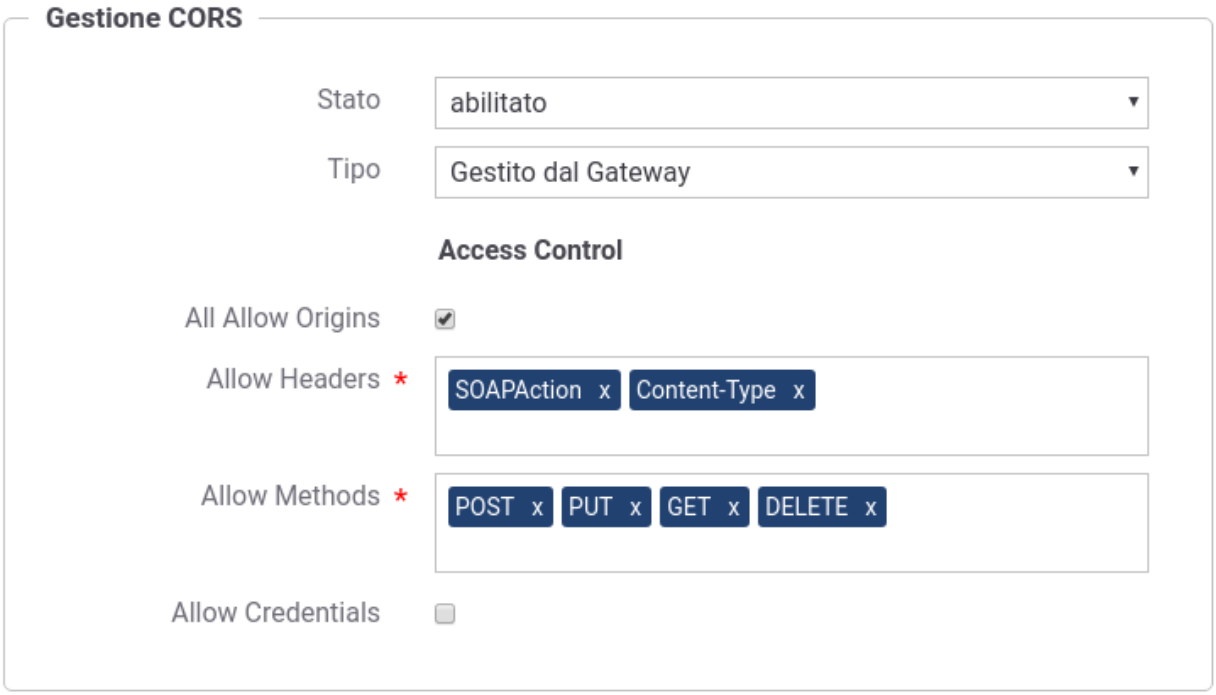

Figure6.7: Gestione CORS

6.8 RateLimiting - Policy di Default

GovWay permette definire un rate limiting sulle singole erogazioni o fruizioni di API. Per una descrizione dettagliata sulle policy di Rate Limiting supportate da GovWay si rimanda alla sezione rateLimiting della guida "Console di Gestione".

In presenza di una installazione con più nodi gateway attivi, GowWay per default effettua il conteggio delle metriche utilizzate dalle policy di rate limiting indipendentemente su ogni singolo nodo del cluster. Questa soluzione è certamente la più efficiente, ma presenta dei limiti evidenti se è necessaria una contabilità precisa del numero di accessi consentiti globalmente da parte dell'intero cluster. In tali situazioni è necessario modificare la configurazioni di default per attivare modalità di conteggio distribuite le quali richiedono alcune configurazioni del sistema descritte nella sezione *Configurazione di Hazelcast*.

Oltre al rate limiting GovWay consente di fissare un numero limite complessivo, indipendente dalle APIs, riguardo alle richieste gestibili simultaneamente dal gateway, bloccando le richieste in eccesso (*Numero Complessivo Richieste Simultanee*).

Sempre a livello globale, GovWay limita la dimensione massima accettata di una richiesta e di una risposta (*Dimensione Massima dei Messaggi*).

6.8.1 Numero Complessivo Richieste Simultanee

GovWay consente di fissare un numero limite complessivo, indipendente dalle singole APIs, riguardo alle richieste gestibili simultaneamente dal gateway, bloccando le richieste in eccesso (*Policy che limita il Numero Complessivo Richieste Simultanee*).

È inoltre possibile configurare il numero massimo di connessioni HTTP mantenute aperte (in presenza di "keep-alive") per stessa destinazione (*Numero Massimo Connessioni "Keep-Alive" per Destinazione*).

Nell'installazione di default entrambi i limiti sono fissati a "200".

Policy che limita il Numero Complessivo Richieste Simultanee

GovWay consente di fissare un numero limite complessivo, indipendente dalle singole APIs, riguardo alle richieste gestibili simultaneamente dal gateway, bloccando le richieste in eccesso. Nell'installazione di default tale limite è fissato a 200 richieste simultanee.

Per modificare la configurazione sul numero limite di richieste simultanee accedere alla voce *"Configurazione - Controllo Traffico"* del menù, sezione *"Limitazione Numero di Richieste Complessive"*.

Figure6.8: Numero Richieste Simultanee

Anche in presenza della policy, precedentemente descritta, si potrebbe rilevare un limite inferiore se a livello di application server esistono ulteriori limitazioni. Di seguito vengono fornite alcune indicazioni a riguardo.

Tomcat

L'application server, per default, limita il numero di richieste a 200. Per poter modificare il limite si deve agire sull'attributo "maxThreads" degli elementi "connector" preseneti nella configurazione di Tomcat (es. in tomcat_home/conf/server.xml).

WildFly

L'application server, per default, limita il numero di worker threads tramite la formula "cpuCount * 16" (es. https://docs.wildfly.org/26/wildscribe/subsystem/io/worker/index.html). Inoltre i worker threads sono condivisi da tutti gli http-listener che per default utilizzando il worker "default" (es. https://docs.wildfly.org/26/wildscribe/subsystem/undertow/server/http-listener/index.html).

Per modificare il livello di soglia e configurare per ogni http-listener un pool di threads dedicato si deve agire sulla configurazione di WildFly (es. in standalone/configuration/standalone.xml) prima creando un worker che definisce il pool di thread.

```
<subsystem xmlns="urn:jboss:domain:io:...">
    <worker name="default"/>
    <worker name="customWorker" task-max-threads="200"/>
```

(continues on next page)

(continua dalla pagina precedente)

```
    ...
</subsystem>
```

Si deve poi effettuare l'associazione definendo nell'elemento "http-listener" l'attributo "worker".

```
<server name="default-server">
        <http-listener name="default" socket-binding="http" worker=
↪"customWorker" ... />
        ...
</server>
```

Numero Massimo Connessioni "Keep-Alive" per Destinazione

GovWay consente di configurare il numero massimo di connessioni HTTP mantenute aperte (in presenza di "keep-alive") per stessa destinazione. Nell'installazione di default tale limite è fissato a 200 connessione per stessa destinazione.

Per modificare la configurazione sul numero di connessioni accedere alla voce *"Configurazione - Proprietà di Sistema"* del menù e modificare il valore associato alla proprietà "*http.maxConnections*".

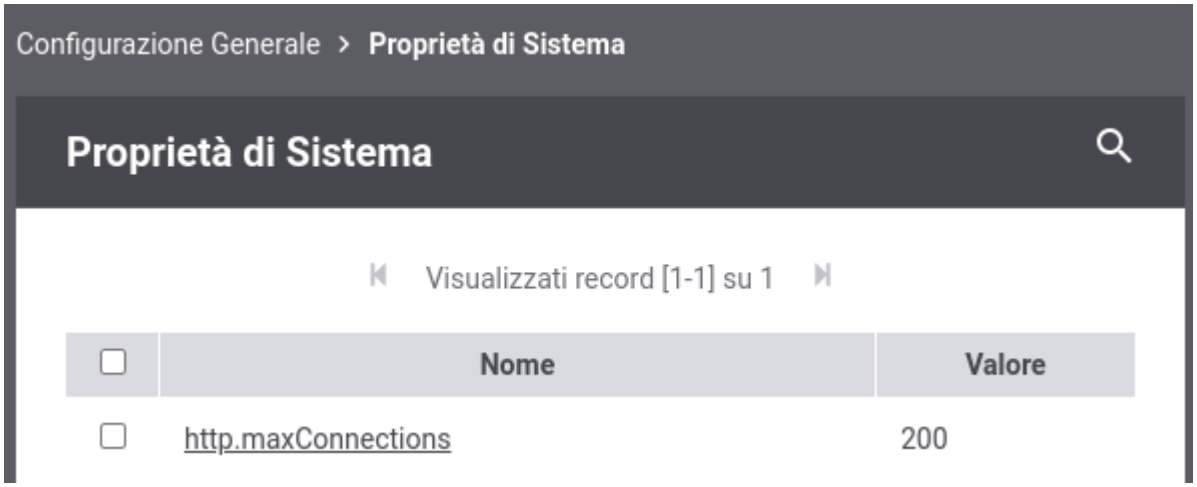

Figure6.9: Numero Massimo di Connessioni "Keep-Alive" per Destinazione

Nota

Effettuata la modifica dei files è necessario un riavvio dell'Application Server per rendere operative le modifiche.

6.8.2 Dimensione Massima dei Messaggi

L'installazione di default di GovWay possiede una policy globale di rateLimiting (Fig. 6.10), definita tramite la metrica "Dimensione Massima Messaggio", che limita la dimensione massima accettata di una richiesta e di una risposta. Il valore di default impostato per entrambe le soglie è 10240k (10MB). Per una descrizione dettagliata sulle policy di Rate Limiting supportate da GovWay si rimanda alla sezione rateLimiting della guida "Console di Gestione".

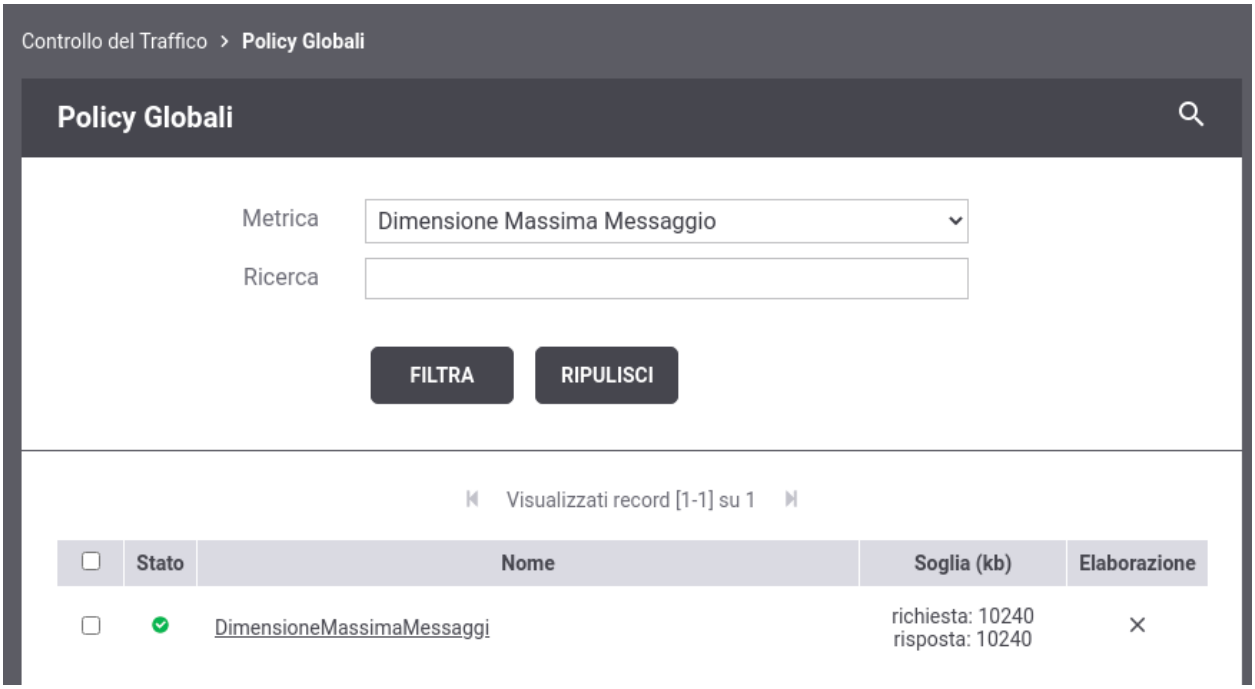

Figure6.10: Policy globale "Dimensione Massima Messaggio"

Nota

La policy "Dimensione Massima Messaggio" non è eliminabile ma è consentito modificarne i valori di soglia o disabilitarla.

Anche in presenza della policy, precedentemente descritta, si potrebbe rilevare un limite inferiore se a livello di application server esistono ulteriori limitazioni sulla dimensione dei messaggi. Di seguito vengono fornite alcune indicazioni a riguardo.

WildFly

L'application server, per default, limita la dimensione del payload delle richieste a 10MB. Per poter modificare il livello di soglia bisogna agire sull'attributo "max-post-size" nell'elemento "http-listener" della configurazione di WildFly (es. in standalone/configuration/standalone.xml): indica il numero di bytes massimo che un payload può contenere per essere processato. Se non presente l'attributo assume il valore di default 10485760 (10MB). È anche possibile disabilitare il limite impostando l'attributo al valore "0".

```
<server name="default-server">
        <http-listener name="default" socket-binding="http" max-post-size=
↪"10485760" .../>
        ...
</server>
```

L'esempio seguente riporta l'errore che si ottiene inviando una richiesta con payload superiore al limite configurato su WildFLy:

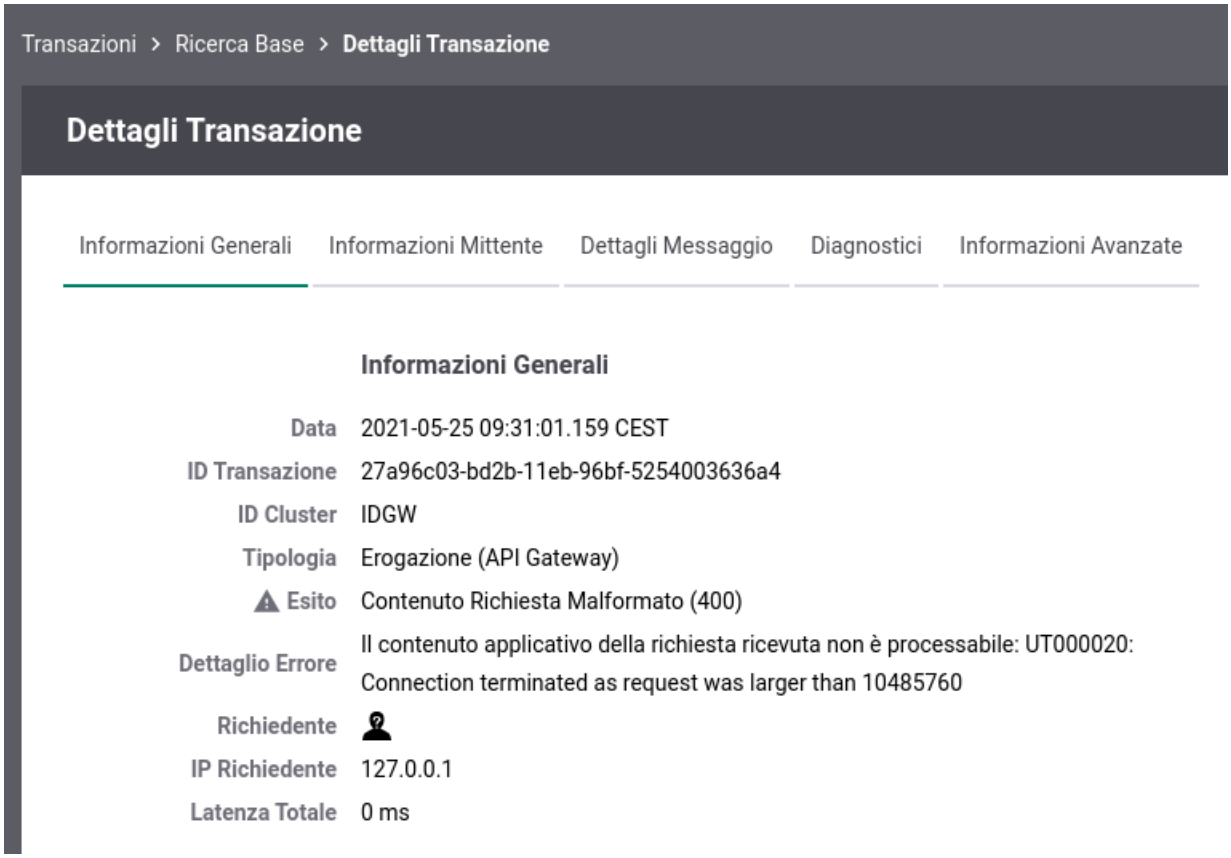

Figure6.11: Violata dimensione massima su WildFLy “UT000020: Connection terminated as request was larger than 10485760”

6.8.3 Configurazione di Hazelcast

GovWay consente di implementare qualunque politica di rate limiting in maniera effettivamente distribuita tra i nodi del cluster, indipendentemente dalle modalità previste per il bilanciamento del carico. Il conteggio delle metriche viene effettuato tramite un archivio dati distribuito implementato tramite Hazelcast (https://github.com/hazelcast/). Per una descrizione dettagliata sul Rate Limiting attuabile su un cluster di nodi si rimanda alla sezione headerGWRateLimitingCluster della guida "Console di Gestione" e nello specifico alla sezione headerGWRateLimitingCluster_distribuita.

Se viene attivata una modalità di sincronizzazione distribuita, i log emessi da Hazelcast riguardanti lo stato della sincronizzazione dei nodi del cluster sono riversati nel file di log *<directory-log>/govway_hazelcast.log*

Una volta avviata tale modalità si potrà riscontrare come nel log venga riportato il seguente warning:

```
WARN <04-06-2022 11:38:24.537> com.hazelcast.logging.AbstractLogger.
↪warning(89): Hazelcast is starting in a Java modular environment (Java 9 and␣
↪newer) but without proper access to required Java packages. Use additional␣
↪Java arguments to provide Hazelcast access to Java internal API. The␣
↪internal API access is used to get the best performance results. Arguments␣
↪to be used:
 --add-modules java.se --add-exports java.base/jdk.internal.ref=ALL-UNNAMED --
↪add-opens java.base/java.lang=ALL-UNNAMED --add-opens java.base/sun.nio.
↪ch=ALL-UNNAMED --add-opens java.management/sun.management=ALL-UNNAMED --add-
↪opens jdk.management/com.sun.management.internal=ALL-UNNAMED
```

Per fornire ad Hazelcast l'accesso alle API interne di java deve essere riportata la configurazione indicata nel warning tra le proprietà java dell'Application Server.

Nota

Ambiente di Produzione In un ambiente di produzione non è consigliato utilizzare un discovery automatico (https://docs.hazelcast.com/hazelcast/5.3/clusters/discovery-mechanisms#auto-detection). Si consiglia di disabilitare l'elemento "auto-detection" e di utilizzare un meccanismo alternativo (esempio il censimento puntuale dei nodi tramite l'elemento "tpc-ip.member-list") agendo sul file *<directory-lavoro>/govway_local.hazelcast.yaml*. Per maggiori informazioni si rimanda al manuale sulla Console di Gestione nella sezione "headerGWRateLimitingCluster_distribuita_hazelcastConfig_sharedConfig"

6.9 Tempi Risposta

GovWay è preconfigurato con dei valori di timeout riguarndati i tempi di risposta dei servizi con cui il gateway interagisce durante l'elaborazione delle richieste. Nel caso delle erogazioni, si tratta dei tempi di risposta dei servizi interni al dominio, successivamente ad una richiesta di erogazione dall'esterno. Nel caso delle fruizioni, si tratta dei tempi di risposta dei servizi esterni, successivamente ad una richiesta di fruizione da parte di un client interno al dominio. I tempi configurabili sono:

- *Connection Timeout (ms)*: Intervallo di tempo atteso, sulle comunicazioni in uscita, prima di sollevare l'errore Connection Timeout (scadenza del tempo di attesa per stabilire una connessione).
- *Read Timeout (ms)*: Intervallo di tempo atteso, dopo aver stabilito una connessione in uscita, prima di sollevare l'errore di Read Timeout (scadenza del tempo di attesa per ricevere il payload dall'interlocutore).
- *Tempo Medio di Risposta (ms)*: Valore di soglia del tempo medio di risposta al fine di valutare la situazione di *Degrado Prestazionale*, condizione per l'applicabilità di eventuali politiche restrittive di rate limiting (per ulteriori dettagli si rimanda alla guida utente).

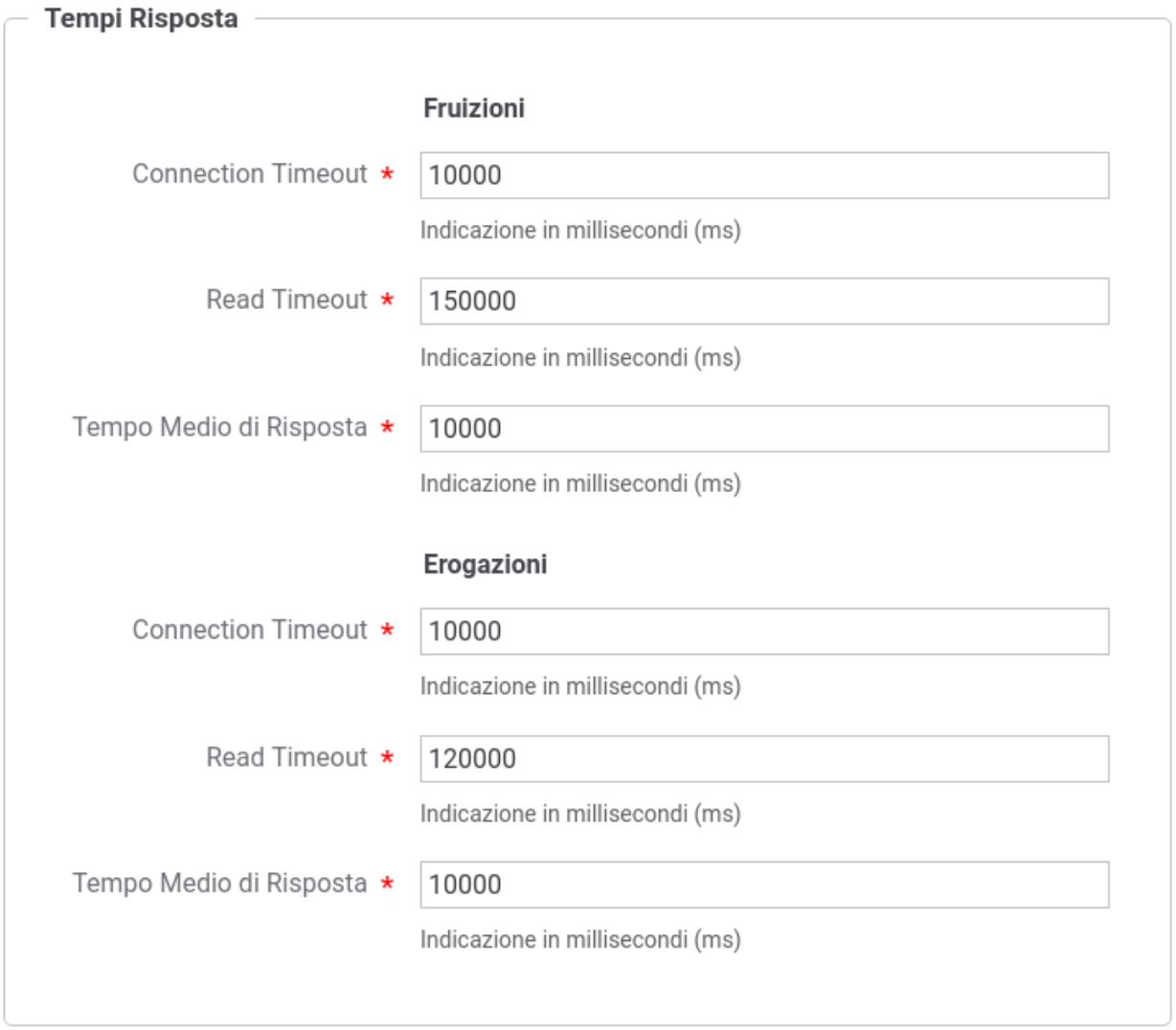

Figure6.12: Tempi Risposta

6.10 Caching della Risposta - Disk Cache

In GovWay è possibile abilitare il salvataggio delle risposte in una cache. Questa funzionalità permette ad un backend server di non dover riprocessare le stesse richieste più volte.

La configurazione di default prevede di salvare in una cache, che risiede in memoria RAM, fino a 5.000 risposte (ogni risposta comporta il salvataggio di due elementi in cache). In caso venga superato il numero massimo di elementi che possano risiedere in cache, vengono eliminate le risposte meno recenti secondo una politica *LRU*.

Per modificare la configurazione della cache in modo da aggiungere una memoria secondaria dove salvare gli elementi in eccesso è possibile agire sul file *<directory-lavoro>/govway_local.jcs.properties* scommentando le seguenti:

```
jcs.region.responseCaching=responseCachingDiskCache
jcs.region.responseCaching.elementattributes.IsSpool=true
```

Per ulteriori dettagli sui parametri di configurazione della memoria secondaria si rimanda alla documentazione della cache http://commons.apache.org/proper/commons-jcs/IndexedDiskCacheProperties.html.

La libreria di caching utilizzata da GovWay, (*JCS*: http://commons.apache.org/proper/commons-jcs/) consente di definire diversi tipi di memoria secondaria. Per ulteriori dettagli su come abilitare i vari tipi di memoria si rimanda alla documentazione: http://commons.apache.org/proper/commons-jcs/JCSPlugins.html.

Nota

Accedendo alla sezione *"Configurazione - Generale", tramite l'utilizzo della govwayConsole in modalità avanzata*, è possibile modificare i parametri di configurazione (numero di elementi e politica di svecchiamento) della cache che risiede in memoria RAM tramite la sezione *"Cache (Risposte)"*.

6.11 Registrazione Token PKCS11

Il Cryptographic Token Interface Standard, PKCS#11, definisce interfacce di programmazione native per token crittografici, come acceleratori crittografici hardware e smartcard.

Come prerequisito per consentire a GovWay di accedere ai token PKCS#11 nativi è necessario innanzitutto configurare correttamente il provider PKCS#11 predispondendo le corrette librerie di sistema (.so o .dll) necessarie.

Si può verificare la configurazione accedendo al token utilizzando il "keytool" di java. Di seguito viene fornito un esempio di accesso ad un token creato con softhsm, il simulatore pkcs11 di dnssec.

```
cat /etc/softhsm_java.conf
> name = softhsm-example
> library = /usr/lib64/libsofthsm2.so
> slotListIndex = 0

keytool -list -keystore NONE -storetype PKCS11 -providerClass sun.security.pkcs11.
↪SunPKCS11 -providerArg /etc/softhsm_java.conf
> Keystore type: PKCS11
> Keystore provider: softhsm-example
>
> Your keystore contains X entries
```

Un provider accessibile correttamente tramite keytool può essere censito tra i provider conosciuti da GovWay agendo sul file *<directory-lavoro>/hsm.properties* Di seguito un esempio di registrazione del provider softhsm di esempio mostrato in precedenza:

```
hsm.keystore.provider=SunPKCS11
hsm.keystore.provider.add=true
hsm.keystore.provider.configFile=/etc/softhsm_java.conf
hsm.keystore.pin=123456
hsm.keystore.keystoreType.label=softhsm-example
hsm.keystore.keystoreType=pkcs11
```

Sarà possibile censire più token utilizzabili da GovWay. Ogni token registrato è identificato dal valore fornito nella proprietà *hsm.keystore.keystoreType.label* e sarà utilizzabile all'interno delle configurazioni di GovWay. La figura Fig. 6.13 mostra un esempio di utilizzo del token registrato nell'esempio sopra riportato.

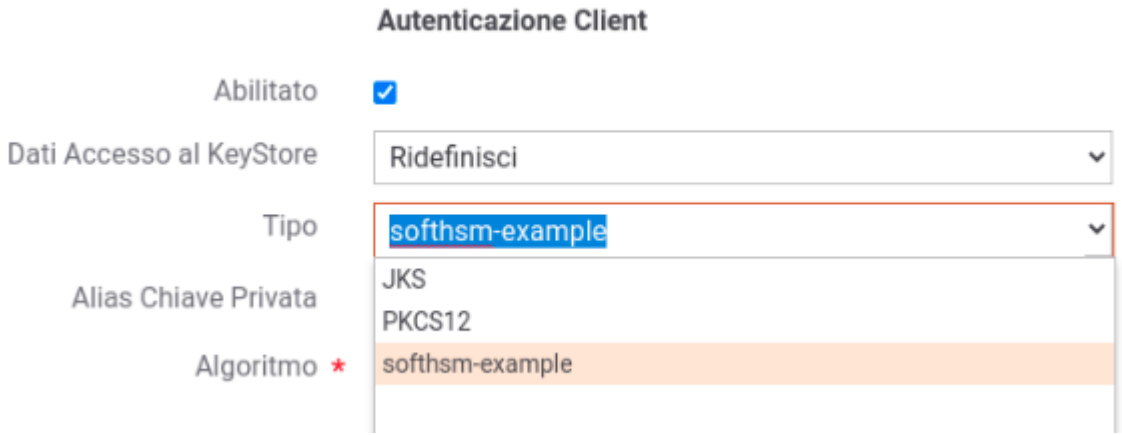

Figure6.13: Esempio di configurazione di un token PKCS11 su GovWay

Di seguito vengono descritte tutte le opzioni di configurazione utilizzabili nella registrazione di un token all'interno del file *<directory-lavoro>/hsm.properties*:

- hsm.<idKeystore>.provider: (obbligatorio su nodo run) indica la classe del provider che deve essere stata registrata in JVM/conf/security/java.security o che dovrà essere aggiunta dinamicamente tramite l'opzione successiva;
- hsm.<idKeystore>.provider.add: (opzionale, default false) indica se il provider fornito deve essere registrato dinamicamente, se non già presente;
- hsm.<idKeystore>.provider.configFile: (opzionale) se fornito verrà utilizzato per configurare il provider tramite l'istruzione "configure(configFile)";
- hsm.<idKeystore>.provider.config: (opzionale) se fornito verrà utilizzato per configurare il provider tramite l'istruzione "configure(config)";
- hsm.<idKeystore>.pin: (obbligatorio su nodo run) pin per accedere al token;
- hsm.<idKeystore>.keystoreType.label: (obbligatorio) label associata al token e visualizzata nelle console;
- hsm.<idKeystore>.keystoreType: (obbligatorio su nodo run) tipo associato al token ed utilizzato per istanziarlo tramite l'istruzione "KeyStore.getInstance(keystoreType, provider)";
- hsm.<idKeystore>.usableAsTrustStore: (opzionale, default false) indica se il token è utilizzabile anche come truststore di certificati;
- hsm.<idKeystore>.usableAsSecretKeyStore: (opzionale, default false) indica se il token è utilizzabile anche come repository di chiavi segrete.

6.12 Online Certificate Status Protocol (OCSP)

Il protocollo Online Certificate Status Protocol (OCSP), descritto nel RFC 2560, consente di verificare la validità di un certificato senza ricorrere alle liste di revoca dei certificati (CRL). Le modalità di verifica possono differire per svariati motivi: dove reperire la url del servizio OCSP a cui richiedere la validità del certificato e il certificato dell'Issuer che lo ha emesso, come comportarsi se un servizio non è disponibile, eventuale validazione CRL alternativa etc.

GovWay consente di definire differenti policy OCSP sul file *<directory-lavoro>/ocsp.properties* e una volta configurate saranno disponibili per la scelta in fase di configurazione di una funzionalità che richiede la validazione di un certificato. La sintassi delle opzioni che consentono di definire una policy OCSP viene descritta nella sezione *Configurazione Policy OCSP*. La figura Fig. 6.14 mostra un esempio di utilizzo di una policy OCSP nella validazione del certificato server in un connettore https.

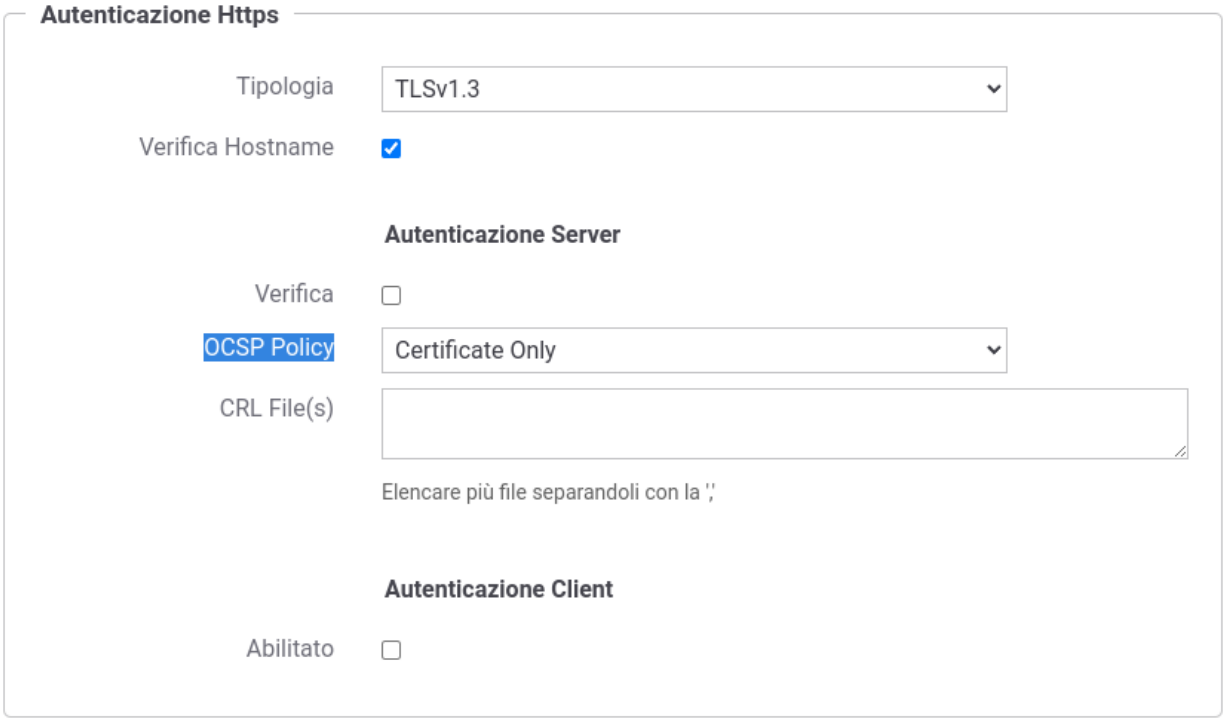

Figure6.14: Esempio di configurazione di una policy OCSP su GovWay

Con l'installazione di GovWay vengono fornite due policy di default che presentano la medesima configurazione e differiscono solamente per il fatto che una policy consente di verificare l'intera catena di certificati. Le caratteristiche di entrambe le policy sono:

- viene verificata la validità temporale del certificato prima di intraprenderne la validazione tramite servizio OCSP;
- il certificato dell'Issuer viene cercato prima nel trustStore configurato insieme alla policy e successivamente nell'extension "Authority Information Access"; se un certificato Issuer non viene trovato la verifica fallisce;
- l'endpoint del OCSP Responder viene cercato nell'extension "Authority Information Access" e se non presente la verifica fallisce;
- oltre alla validazione della risposta ottenuta dal servizio OCSP, vengono attuati ulteriori controlli che differiscono a seconda del certificato di firma utilizzato dal OCSP Responder:
  - se combacia con il certificato Issuer che ha emesso anche il certificato in fase di verifica non vengono attuati altri controlli;

- se il certificato di firma differisce ma è stato comunque emesso dall'Issuer del certificato in fase di verifica, vengono effettuate i seguenti ulteriori controlli:
  - deve possedere l'extended key usage "id_kp_OCSPSigning";
  - viene verificata la validità temporale e la verifica rispetto al certificato Issuer;
  - se nel certificato di firma è presente l'extension "CRL Distribution Points" viene attuata una verifica del certificato tramite la CRL indicata;
- se il certificato di firma differisce e non è stato emesso dallo stesso Issuer del certificato in fase di verifica il processo di validazione fallisce.

Nota

È possibile realizzare una configurazione che consente di attuare una validazione del certificato di firma anche per questo caso definendo una nuova policy OCSP che definisce, tramite le proprietà "ocsp.<policyId>.signer.trustStore" descritta nella sezione *Configurazione Policy OCSP*, il trustStore da cui reperire il certificato Issuer che ha emesso il certificato di firma della risposta OCSP.

Il risultato di una validazione OCSP di un certificato viene mantenuto in cache, in modo che successive validazioni non debbano attuare nuovamente il processo di verifica. Per ulteriori dettagli sul tempo di validità di un risultato inserito in cache si rimanda alla sezione *Cache*.

6.12.1 Configurazione Policy OCSP

GovWay consente di definire molteplici policy OCSP agendo sul file *<directory-lavoro>/ocsp.properties*.

La configurazione di ogni policy è rappresentata dall'insieme di proprietà che presentano lo stesso prefisso "ocsp.<idPolicy>". L'identificativo <idPolicy> serve univocamente ad aggregare tali proprietà.

Una policy viene identificata univocamente da GovWay tramite la proprietà "ocsp.<idPolicy>.type"; il suo valore deve essere univoco rispetto ai valori forniti nelle altre policy per la medesima proprietà.

Ogni policy OCSP configurata, viene resa disponibile per la scelta nella console di gestione (govwayConsole), tramite il valore definito nella proprietà "ocsp.<idPolicy>.label"; di conseguenza anche questo valore deve essere univoco rispetto a quello fornito per le altre policy.

Di seguito un esempio della policy di default fornita con GovWay:

```
ocsp.default.type=default
ocsp.default.label=Certificate Only
#
# Verifica di tutti i certificati della catena
ocsp.default.certificateChainVerify=false
#
# Verifica la validità del certificato prima di intraprendere la validazione tramite␣
↪OCSP/CRL
ocsp.default.checkValidity=true
ocsp.default.checkCAValidity=true
#
# Issuer
ocsp.default.ca.source=CONFIG,AUTHORITY_INFORMATION_ACCESS
#
# OCSP Responder URL
ocsp.default.url.source=AUTHORITY_INFORMATION_ACCESS
```

(continues on next page)

(continua dalla pagina precedente)

```
ocsp.default.url.breakStatus=OCSP_BUILD_REQUEST_FAILED
#
# Il certificato di firma utilizzato per la risposta OCSP può contenere indicazioni di␣
↪CRL per la sua validazione
ocsp.default.crl.signingCert.check=true
#
ocsp.default.crl.source=AUTHORITY_INFORMATION_ACCESS
#
# Il truststore utilizzato per attuare la verifica CRL viene definito tramite le␣
↪seguenti opzioni
ocsp.default.crl.trustStore.source=CONFIG,AUTHORITY_INFORMATION_ACCESS
```

Di seguito vengono descritte tutte le opzioni di configurazione utilizzabili nella registrazione di una policy all'interno del file *<directory-lavoro>/ocsp.properties*:

Aspetti generali:

- ocsp.<idPolicy>.certificateChainVerify: [opzionale, default:true] indicazione se deve essere verificata l'intera catena di certificati;
- ocsp.<idPolicy>.checkValidity: [opzionale, default:true] indicazione se deve essere effettuato un controllo di validità delle date del certificato prima di validarlo tramite OCSP;
- ocsp.<idPolicy>.checkCAValidity: [opzionale, default:true] consente di impostare un controllo identico a quello precedente, ma relativamente ai certificati che rappresentano CA.

Modalità per ottenere il certificato Issuer che ha emesso il certificato in fase di verifica:

- ocsp.<idPolicy>.ca.source: [obbligatorio] indicazione sulle modalità con cui reperire i certificati CA che corrispondono all'issuer del certificato in fase di verifica. È possibile indicare più modalità separate da virgola, e in tal caso vengono provate nell'ordine configurato fino a che non viene trovato il certificato di CA:
  - AUTHORITY_INFORMATION_ACCESS: se presente nel certificato viene acceduta l'extension "AuthorityInformationAccess" allo scopo di recuperare il certificato dichiarato come "CA Issuer";
  - CONFIG: il certificato viene cercato nel truststore indicato nella configurazione dove viene riferita la policy OCSP;
  - ALTERNATIVE_CONFIG: il certificato viene cercato nel truststore indicato nella proprietà "ocsp.<idPolicy>.ca.alternativeTrustStore".
- ocsp.<idPolicy>.ca.alternativeTrustStore: [opzionale] consente di indicare un truststore dove viene ricercato il certificato CA che corrisponde all'issuer del certificato in fase di verifica, in caso di modalità "ALTERNATIVE_CONFIG" indicata nella proprietà "ocsp.<idPolicy>.ca.source". Il tipo di truststore e la password devono essere indicate rispettivamente nelle proprietà:
  - ocsp.<idPolicy>.ca.alternativeTrustStore.type [opzionale, default:jks]
  - ocsp.<idPolicy>.ca.alternativeTrustStore.password [obbligatorio].
- ocsp.<idPolicy>.ca.notFound.rejectsCertificate: [opzionale, default:true] nel caso non sia possibile recuperare il certificato CA tramite una delle modalità indicate in "ocsp.<idPolicy>.ca.source" la validazione fallisce. Disabilitando la proprietà il processo di validazione termina correttamente senza segnalare anomalie (il servizio OCSP non verrà invocato).

Aspetti relativi alla generazione della richiesta OCSP e alla validazione della risposta OCSP:

- ocsp.<idPolicy>.nonce.enabled: [opzionale, default:true]: indicazione se nella richiesta inviata al provider OCSP deve essere aggiunta una extension contenente un identificativo univoco non predicibile (nonce: https://www.rfc-

editor.org/rfc/rfc6960#section-4.4.1). Se una extensions "nonce" viene restituita anche nella risposta viene attuato un controllo di uguaglianze rispetto a quello inviato nella richiesta.

- ocsp.<idPolicy>.secureRandomAlgorithm: [opzionale, default:SHA1PRNG] indicazione sull'algoritmo utilizzato per generare il nonce. I valori indicati devono corrispondere ad uno tra quelli definiti nell'enumeration "org.openspcoop2.utils.random.SecureRandomAlgorithm".
- ocsp.<idPolicy>.response.date.toleranceMilliseconds: [opzionale, default:600000ms=10minuti] indicazione in millisecondi della tolleranza nel controllo di validità delle date in caso in cui non venga attuata una verifica di uguaglianza tra nonce della richiesta e nonce della risposta.
- ocsp.<idPolicy>.extendedKeyUsage: [opzionale, default:OCSP_SIGNING] extendedKeyUsage richiesti al certificato di firma dell'OCSP, nel caso in cui la risposta viene firmata dal OCSP Responder utilizzando un certificato diverso dal certificato CA che ha emesso il certificato in fase di validazione. Se la proprietà viene definita vuota, non verrà attuato alcun controllo, ma se ne sconsiglia l'attuazione poiché il controllo è fondamentale per prevenire attacchi man in the middle (http), dove l'attaccante potrebbe firmare con un altro certificato in suo possesso rilasciato dalla stessa CA, certificato però non adibito a firmare risposte OCSP.

Modalità per ottenere il certificato Issuer che ha emesso il certificato di firma della risposta OCSP. Queste opzioni sono necessarie quando la risposta viene firmata dal OCSP Responder utilizzando un certificato emesso da una CA differente da quella che ha emesso il certificato in fase di validazione:

- ocsp.<idPolicy>.signer.trustStore: [opzionale] consente di indicare un truststore dove viene ricercato il certificato CA che corrisponde all'issuer del certificato di firma utilizzato dal OCSP responder per firmare la risposta. Il tipo di truststore e la password devono essere indicate rispettivamente nelle proprietà:
  - ocsp.<idPolicy>.signer.type [opzionale, default:jks]
  - ocsp.<idPolicy>.signer.password [obbligatorio].
- ocsp.<idPolicy>.signer.alias: [opzionale] insieme alla definizione della proprietà "ocsp.<idPolicy>.signer.trustStore" consente l'autorizzazione puntuale di un certificato di firma atteso nelle risposte firmate dal servizio OCSP.

Aspetti relativi all'invocazione del servizio OCSP:

- ocsp.<idPolicy>.url.source: [obbligatorio] indicazione sulle modalità con cui reperire la url del servizio OCSP. È possibile indicare più modalità separate da virgola tra le seguenti:
  - AUTHORITY_INFORMATION_ACCESS: se presente nel certificato viene acceduta l'extension "AuthorityInformationAccess" allo scopo di recuperare le url dei servizi "OCSP";
  - ALTERNATIVE_CONFIG: gli endpoint dei servizi OCSP vengono indicati nella proprietà "ocsp.<idPolicy>.url.alternative".
- ocsp.<idPolicy>.url.alternative: [opzionale] consente di indicare l'endpoint del servizio OCSP da utilizzare per la verifica del certificato; è possibile indicare più endpoint, separati da virgola.
- ocsp.<idPolicy>.url.alternative.ca: [opzionale] identico alla precedente proprietà, ma utilizzati per validare i certificati che rappresentano CA.
- ocsp.<idPolicy>.url.notFound.rejectsCertificate: [opzionale, default:true] nel caso non sia possibile recuperare l'endpoint del servizio OCSP tramite una delle modalità indicate in "ocsp.<idPolicy>.url.source" la validazione fallisce. Disabilitando la proprietà il processo di validazione termina correttamente senza segnalare anomalie (il servizio OCSP non verrà invocato).
- ocsp.<idPolicy>.url.notFound.rejectsCA: [opzionale, default:false] nel caso non sia possibile recuperare l'endpoint del servizio OCSP di un certificato di CA, la validazione termina correttamente. Abilitando la proprietà è possibile far fallire il processo di validazione.

- ocsp.<idPolicy>.url.returnCodeOk: [opzionale, default:200] consente di indicare i codici http delle risposte del servizio OCSP che devono essere considerate valide. Solamente nelle risposte valide viene poi validata e considerata la risposta ottenuta; è possibile indicare più codici separati da virgola.
- ocsp.<idPolicy>.url.breakStatus: [opzionale] nel caso di più endpoint OCSP disponibili, i servizi vengono invocati nell'ordine recuperato dalle modalità indicate nella proprietà "ocsp.<idPolicy>.url.source". Una invocazione di un servizio OCSP può fallire per svariati motivi, definiti nell'enumeration "org.openspcoop2.utils.certificate.ocsp.OCSPResponseCode". Per default qualsiasi sia il motivo del fallimento, la validazione termina con errore. La proprietà seguente consente di indicare gli stati di errore, separati da virgola, per cui il processo di validazione si interrompe e non prova ad invocare il successivo endpoint disponibile. Ad esempio, ocsp.<idPolicy>.url.breakStatus=OCSP_BUILD_REQUEST_FAILED

Le seguenti opzione vengono utilizzate sia durante l'invocazione del servizio OCSP che per il retrieve di certificati indicati in extension "AuthorityInformationAccess" o crl riferite in "CRLDistributionPoints":

- ocsp.<idPolicy>.connectTimeout: [opzionale, default:10000ms] indicazione in millisecondi sul tempo di instaurazione della connessione.
- ocsp.<idPolicy>.readTimeout: [opzionale, default:15000ms] indicazione in millisecondi sul tempo di attesa di una risposta dal servizio OCSP.
- ocsp.<idPolicy>.https.hostnameVerifier: [optional, default:true] consente, nel caso in cui le url da contattare siano in https, di disabilitare la verifica dell'hostname rispetto al CN del certificato restituito dal server.
- ocsp.<idPolicy>.https.trustAllCerts: [opzionale, default:false] consente, nel caso in cui le url da contattare siano in https, di accettare qualsiasi certificato restituito dal server.
- ocsp.<idPolicy>.https.trustStore: [opzionale] consente, nel caso in cui le url da contattare siano in https, di indicare un truststore dove viene ricercato il certificato server. Il tipo di truststore e la password devono essere indicate rispettivamente nelle proprietà
  - ocsp.<idPolicy>.https.trustStore.type [opzionale, default:jks]
  - ocsp.<idPolicy>.https.trustStore.password [obbligatorio].
- ocsp.<idPolicy>.https.keyStore: [opzionale] consente, nel caso in cui le url da contattare siano in https, di indicare un keystore dove viene ricercato il certificato client da spedire. Il tipo di keystore e la password devono essere indicate rispettivamente nelle proprietà
  - ocsp.<idPolicy>.https.trustStore.type [opzionale, default:jks]
  - ocsp.<idPolicy>.https.trustStore.password [obbligatorio].

  La password della chiave privata deve essere indicata nella proprietà:
  - ocsp.<idPolicy>.https.key.password [obbligatorio].

  Se nel keystore esistono più chiavi private deve essere indicata la chiave da utilizzare tramite la proprietà:
  - ocsp.<idPolicy>.https.key.alias [opzionale].
- ocsp.<idPolicy>.username e ocsp.<idPolicy>.password: [opzionali] consentono di specificare una credenziale basic.
- ocsp.<idPolicy>.forwardProxy.url: [opzionale] consente di indicare la url di un proxy applicativo a cui verranno inoltrate tutte le richieste; l'indirizzo remoto del servizio ocsp o della risorsa da recuperare (es. CAIssuer in AuthorityInformationAccess) viene indicata al proxy applicativo tramite un header HTTP o un parametro della url definito tramite le seguenti proprietà:
  - ocsp.<idPolicy>.forwardProxy.header: [opzionale] l'endpoint remoto, a cui il proxy applicativo dovrà inoltrare la richiesta, viene indicato nell'header http configurato nella proprietà;
  - ocsp.<idPolicy>.forwardProxy.queryParameter: [opzionale] l'endpoint remoto, a cui il proxy applicativo dovrà inoltrare la richiesta, viene indicato nel parametro della query configurato nella proprietà;

  - ocsp.<idPolicy>.forwardProxy.base64: [opzionale, default:true] indicazione se l'endpoint remoto inserito nell'header http o nel parametro della query debba essere codificato in base64 o meno.

> **Nota**
>
> L'abilitazione di un proxy applicativo richiede obbligatoriamente la definizione di una tra le due seguenti proprietà: "ocsp.<idPolicy>.forwardProxy.header" o "ocsp.<idPolicy>.forwardProxy.queryParameter".

Aspetti riguardanti l'attivazione di una validazione del certificato tramite CRL:

- ocsp.<idPolicy>.crl.signingCert.check: [opzionale, default:false] il certificato di firma utilizzato per la risposta OCSP può contenere indicazioni di CRL per la sua validazione. Se presenti verranno verificate se viene abilitata la seguente opzione.
- ocsp.<idPolicy>.crl.ca.check: [opzionale, default:true] il certificato di CA presente nella certificate chain può contenere indicazioni di CRL per la sua validazione, invece che OCSP. Se presenti verranno verificate se viene abilitata la seguente opzione.
- ocsp.<idPolicy>.crl.enabled: [opzionale, default:false] consente di attivare una validazione alternativa a OCSP che utilizza solamente CRL per la validazione del certificato.

Nei casi di attivazione di validazione tramite CRL vengono utilizzate le seguenti configurazioni.

- ocsp.<idPolicy>.crl.source: [opzionale, default:AUTHORITY_INFORMATION_ACCESS] indicazione sulle modalità con cui reperire i CRL. È possibile indicare più modalità separate da virgola tra le seguenti:
  - AUTHORITY_INFORMATION_ACCESS: se presente nel certificato viene acceduta l'extension "CRLDistributionPoints" allo scopo di recuperare l'url dove recuperare la CRL;
  - CONFIG: crl indicata nella configurazione dove viene riferita la policy OCSP;
  - ALTERNATIVE_CONFIG: la crl viene recuperata accedendo alla url indicata nella proprietà "ocsp.<idPolicy>.crl.alternative".

> **Nota**
>
> Nel caso di proprietà "ocsp.<idPolicy>.crl.signingCert.check" o "ocsp.<idPolicy>.crl.ca.check" abilitata, la modalità "AUTHORITY_INFORMATION_ACCESS" è obbligatoria.

- ocsp.<idPolicy>.crl.alternative: [opzionale] consente di indicare un indirizzo dove recuperare la CRL; è possibile indicare più endpoint, separati da virgola.
- ocsp.<idPolicy>.crl.notFound.rejectsCertificate: [opzionale, default:false] nel caso non sia possibile recuperare CRL tramite una delle modalità indicate in "ocsp.<idPolicy>.crl.source" la validazione termina correttamente. Abilitando la proprietà è possibile far fallire il processo di validazione.
- ocsp.<idPolicy>.crl.notFound.rejectsCA: [opzionale, default:false] consente di impostare un controllo identico a quello precedente, ma relativamente ai certificati che rappresentano CA.
- ocsp.<idPolicy>.crl.trustStore.source: [opzionale, default:AUTHORITY_INFORMATION_ACCESS] indicazione sulle modalità con cui costruire il truststore utilizzato per la verifica delle CRL. È possibile indicare più modalità separate da virgola, e in tal caso vengono costruito un truststore contenente tutti i certificati recuperati:
  - AUTHORITY_INFORMATION_ACCESS: se presente nel certificato viene acceduta l'extension "AuthorityInformationAccess" e l'extension "CRLDistributionPoints" allo scopo di recuperare in entrambe il certificato dichiarato come "CA Issuer".

  - CONFIG: il certificato viene cercato nel truststore indicato nella configurazione dove viene riferita la policy OCSP;
  - ALTERNATIVE_CONFIG: il certificato viene cercato nel truststore indicato nella proprietà "ocsp.<idPolicy>.crl.alternativeTrustStore".
- ocsp.<idPolicy>.crl.alternativeTrustStore: [opzionale] consente di indicare un truststore utilizzato per la verifica delle CRL. Il tipo di truststore e la password devono essere indicate rispettivamente nelle proprietà:
  - ocsp.<idPolicy>.crl.alternativeTrustStore.type [opzionale, default:jks]
  - ocsp.<idPolicy>.crl.alternativeTrustStore.password [obbligatorio].

6.13 API di Configurazione e Monitoraggio

Se nell'installer sono stati selezionati i servizi che espongono API REST per la configurazione e il monitoraggio di GovWay (Fig. 3.6) gli indirizzi base per utilizzarli sono:

- *http://<hostname-pdd>/govway/ENTE/api-config/v1/*
- *http://<hostname-pdd>/govway/ENTE/api-monitor/v1/*

Per poterli invocare deve prima essere completata la configurazione del Controllo degli Accessi accedendo alla console di gestione tramite browser all'indirizzo *http://<hostname-pdd>/govwayConsole* utilizzando le credenziali fornite durante l'esecuzione dell'installer.

Accendendo alla lista delle Erogazioni si può notare come le API relative alla configurazione ed al monitoraggio riportano uno "stato rosso" che evidenzia una configurazione incompleta.

Procedere con la configurazione del apiGwControlloAccessi di ogni API al fine di renderla invocabile dall'esterno secondo le modalità di autenticazione ed autorizzazione desiderate. Per maggiori informazioni sul *Controllo degli Accessi* si rimanda alla Guida della Console di Gestione.

6.14 Configurazione in Load Balancing

Per realizzare un'installazione in load balancing è necessario predisporre più istanze dell'Application Server, ognuna con una propria installazione del software. Sarà inoltre necessario:

1. Che tutte le istanze di GovWay siano configurate per condividere lo stesso DB.
2. Che esista un Load Balancer in grado di bilanciare il flusso di richieste in arrivo sulle varie istanze di AS ospitanti il software GovWay.
3. Che GovWay sia opportunamente configurato con un identificatore unico che contraddistingua lo specifico nodo.

In particolare per realizzare la configurazione descritta al punto 3, è necessario:

- Editare il file <directory-lavoro>/govway_local.properties aggiungendo le seguenti righe:

```
# Identificativo univoco della macchina
org.openspcoop2.pdd.cluster_id=#IDGW#
# Identificativo univoco numerico della macchina
org.openspcoop2.pdd.cluster_id.numeric=#NUMERO#
# Cifre utilizzate per l'utilizzo dell'identificativo univoco numerico come␣
↪prefisso di un numero seriale (es. identificativo eGov)
org.openspcoop2.pdd.cluster_id.numeric.dinamico.cifre=#NUMEROCIFRE#
```

  - inserendo al posto di #IDGW# l'identificatore unico associato alla specifica istanza che si sta configurando. Scegliere un identificativo con cui si possa facilmente riconoscere la macchina, ad esempio l'hostname.
  - inserendo al posto di #NUMERO# l'identificatore unico numerico associato all'istanza. Scegliere un identificativo numerico progressivo, a partire da 0, per ciascuna istanza del software GovWay nel cluster (da 0 a 99).
  - inserendo al posto di #NUMEROCIFRE# le cifre utilizzate per l'utilizzo dell'identificativo univoco numerico come prefisso di un numero seriale (es. identificativo eGov); il numero di cifre consente di aggiungere un prefisso "0" all'identificativo numerico se inferiore al numero di cifre indicate. Ad esempio un identificativo 5 verrà serializzato come "05" in caso di "org.openspcoop2.pdd.cluster_id.numeric.dinamico.cifre=2".
    - 1: permette di avere 10 macchine (da 0 a 9)
    - 2: permette di avere 100 macchine (da 00 a 99)

- Effettuata la modifica dei files è necessario un riavvio dell'Application Server per rendere operative le modifiche.

Nota

La directory "<directory-lavoro>" è la directory contenente tutti i files di configurazione. Verificare quale directory è stata indicata durante l'esecuzione del setup (vedi Esecuzione dell'Installer).

6.14.1 Configurazione delle Console

La configurazione del Load Balancing si completa fornendo ulteriori dati di configurazione alle console grafiche. Queste configurazioni consentono alle console di avere i corretti riferimenti ai nodi presenti in modo da poter dettagliare questo aspetto nelle proprie maschere ed inoltre poter propagare eventuali modifiche su ogni nodo senza attendere il timeout della cache o richiedere riavvii dell'AS.

A tale scopo sarà necessario:

1. Editare il file *<directory-lavoro>/govway_local.properties* aggiungendo le seguenti righe su ogni GovWay in Load Balancing:

```
# JMX Resources
org.openspcoop2.pdd.check.readJMXResources.enabled=true
org.openspcoop2.pdd.check.readJMXResources.username=#USERNAME#
org.openspcoop2.pdd.check.readJMXResources.password=#PASSWORD#
```

   inserendo al posto di #USERNAME# e #PASSWORD# le credenziali che dovranno essere utilizzate dalle console e che dovranno essere configurate nei punti successivi di questo paragrafo.

2. Editare il file *<directory-lavoro>/govway.nodirun.properties*

   Disabilitare la configurazione per la singola istanza commentando la proprietà "remoteAccess.checkStatus.url":

```
# Configurazione in Singola Istanza
#remoteAccess.checkStatus.url=http://127.0.0.1:8080/govway/check
```

   Abilitare la configurazione della gestione in Load Balancing scommentando le seguenti righe:

```
# Configurazione in Load Balancing
tipoAccesso=govway
aliases=#IDGW1#,..,#IDGWN#
```

   Devono essere elencati tutti gli identificativi, di ogni nodo gateway in Load Balancing, descritti in precedenza e registrati nella proprietà:

```
org.openspcoop2.pdd.cluster_id del file govway_local.properties
```

   Per ogni identificativo devono inoltre essere fornite le seguenti informazioni:

```
# Configurazione IDGW1
#IDGW1#.descrizione=#DESCRIZIONEGW1#
#IDGW1#.remoteAccess.url=http://#HOSTGW1#:#PORTGW1#/govway/check
#IDGW1#.remoteAccess.username=#USERNAMEGW1#
#IDGW1#.remoteAccess.password=#PASSWORDGW1#
...
# Configurazione IDGWN
#IDGWN#.descrizione=#DESCRIZIONEGWN#
#IDGWN#.remoteAccess.url=http://#HOSTGWN#:#PORTGWN#/govway/check
#IDGWN#.remoteAccess.username=#USERNAMEGWN#
#IDGWN#.remoteAccess.password=#PASSWORDGWN#
```

   Devono essere elencati inserendo al posto di #USERNAMEGW# e #PASSWORDGW# le credenziali utilizzate in precedenza nel file:

```
govway_local.properties, proprietà
org.openspcoop2.pdd.check.readJMXResources.username e
org.openspcoop2.pdd.check.readJMXResources.password
```

   Indicare inoltre al posto di #HOSTGW# e #PORTGW# l'hostname e la porta con cui è raggiungibile GovWay. Infine deve anche essere fornita una descrizione per ogni nodo in Load Balancing al posto di #DESCRIZIONEGW#.

Nota

Per mantenere una retrocompatibilità con le configurazioni descritte nelle precedenti versioni e attuate sui file *<directory-lavoro>/console_local.properties* e *<directory-lavoro>/monitor_local.properties*, le console utilizzeranno tali configurazioni se non riscontrano la presenza del nuovo file *<directory-lavoro>/govway.nodirun.properties*.

Configurazione HTTPS

È possibile configurare un accesso ad una url https tramite le seguenti proprietà aggiuntive, definendo il truststore da utilizzare per verificare il certificato ritornato dal server:

```
# Esempio per nodo IDGWX di un accesso tramite connettore https
#IDGWX.remoteAccess.https=true
#IDGWX.remoteAccess.https.verificaHostName=true
#IDGWX.remoteAccess.https.autenticazioneServer=true
#IDGWX.remoteAccess.https.autenticazioneServer.truststorePath=PATH
#IDGWX.remoteAccess.https.autenticazioneServer.truststoreType=jks
#IDGWX.remoteAccess.https.autenticazioneServer.truststorePassword=PASSWORD
```

Disabilitando l'autenticazione server, non sarà invece necessario definire un truststore ma verrà accettato qualsiasi certificato server (insecure):

```
# Esempio per nodo IDGWX di un accesso tramite connettore https
#IDGWX.remoteAccess.https=true
#IDGWX.remoteAccess.https.verificaHostName=true
#IDGWX.remoteAccess.https.autenticazioneServer=false
```

Le proprietà suddette, oltre a poter essere definite per ogni nodo possono anche essere configurate una volta sola eliminando il prefisso che identifica un nodo. Ad esempio:

```
# Esempio di un accesso tramite connettore https valido per tutti i nodi
remoteAccess.https=true
remoteAccess.https.verificaHostName=true
remoteAccess.https.autenticazioneServer=false
```

Configurazione Timeout

È possibile configurare i parametri di timeout (valori in millisecondi) agendo sulle seguenti proprietà:

```
#IDGW1.remoteAccess.readConnectionTimeout=5000
#IDGW1.remoteAccess.connectionTimeout=5000
```

Gruppi di Nodi (govwayConsole)

La console di gestione consente, nella sezione "Runtime", di svuotare le cache di tutti i nodi tramite un'unica operazione. Per attuare un comportamento simile ma limitato ad un gruppo di nodi è possibile configurare le seguenti proprietà classificando i nodi in gruppi:

```
# Classificazione dei nodi in gruppi
aliases.<idGruppo1>=#IDGW1,#IDGW2
aliases.<idGruppo2>=#IDGW2,#IDGWN
```

Configurazione Avanzata delle Sonde (govwayMonitor)

La console di monitoraggio invoca periodicamente un servizio "sonda" di ogni nodo registrato per verificarne il corretto funzionamento. Per default la url invocata è quella configurata nella proprietà "#IDGWN#.remoteAccess.url" descritta in precedenza. È possibile far utilizzare alla console di monitoraggio una url differente aggiungendo al file *<directory-lavoro>/govway.nodirun.properties* la seguente configurazione aggiuntiva:

```
# Configurazione IDGW1
#IDGW1#.remoteAccess.checkStatus.url=http://#HOSTGW1#:#PORTGW1#/govway/check
...
# Configurazione IDGWN
#IDGWN#.remoteAccess.checkStatus.url=http://#HOSTGWN#:#PORTGWN#/govway/check
```

È inoltre possibile elencare un numero di nodi differenti aggiungendo nel file *<directory-lavoro>/monitor_local.properties* la seguente proprietà:

```
# Configurazione in Load Balancing
statoPdD.sonde.standard.nodi=IDGW1,..,IDGWN
```

6.15 Configurazione HTTPS

GovWay processa ogni richiesta in una duplice veste agendo sia da server al momento della ricezione della richiesta che da client al momento di inoltrare la richiesta verso i backend applicativi.

In entrambi i ruoli la configurazione varia a seconda dell'architettura in cui è stato dispiegato GovWay (es. presenza di un Web Server). Nella sezioni successive vengono forniti dettagli su come è possibile attuare la configurazione sia quando GovWay agisce da server che quando agisce da client.

6.15.1 Comunicazioni in Ingresso

La configurazione varia a seconda se la terminazione ssl è gestita direttamente sull'application server (wildfly o tomcat) o viene gestita da un frontend http (Apache httpd, IIS, etc).

Wildfly

Le sezioni successive forniscono degli esempi utili ad attuare la configurazione https tramite due modalità differenti. La soluzione *Wildfly (security-realms)* non è più utilizzabile dalla versione 25 compresa in poi di WildFly.

Nota

Per conoscere maggiori dettagli e modalità di configurazioni differenti fare riferimento a quanto indicato nella documentazione ufficiale dell'Application Server Wildfly (http://wildfly.org).

Wildfly (elytron - server-ssl-contexts)

Nota

La seguente sezione fornisce degli esempi utili ad attuare la configurazione https. Per conoscere maggiori dettagli e modalità di configurazioni differenti fare riferimento a quanto indicato nella documentazione ufficiale dell'Application Server Wildfly (http://wildfly.org).

La configurazione può essere attuata nel file standalone.xml che si trova all'interno della cartella "WILDFLY_HOME/standalone/configuration/".

- Deve prima essere definito un keystore contenente il certificato che il server deve esporre, associandolo ad un nuovo "server-ssl-context" aggiunto tra i contesti esistenti.

```
<subsystem xmlns="urn:wildfly:elytron:xx" ....>
    <providers>
        ...
    </sasl>
    <tls>
        <key-stores>
            <key-store name="applicationKS">
                ...
            </key-store>
            <key-store name="govwayExampleKeyStore">
                <credential-reference clear-text="changeit"/>
                <implementation type="JKS"/>
                <file path="/etc/govway/keys/govway_server.jks"/>
            </key-store>
        </key-stores>
        <key-managers>
            <key-manager name="applicationKM" ...>
                ...
            </key-manager>
            <key-manager name="govwayExampleKeyManager" key-store=
↪"govwayExampleKeyStore" alias-filter="aliasInKeystore">
                <credential-reference clear-text="changeit"/> <!--␣
↪password chiave privata -->
            </key-manager>
        </key-managers>
        <server-ssl-contexts>
            <server-ssl-context name="applicationSSC" .../>
            <server-ssl-context name="govwayExampleSSC" need-client-auth=
↪"false" key-manager="govwayExampleKeyManager"/>
        </server-ssl-contexts>
    </tls>
</subsystem>
```

se oltre ad esporre un certificato server, si deve autenticare il certificato client del chiamante, la configurazione deve essere estesa con la definizione di un trustStore che contenga i certificati necessari a validarli e un ssl-contest configurato per richiedere il certificato client tramite l'attributo "need-client-auth" se si desidera obbligare il client a presentarsi con un certificato o tramite l'attributo "want-client-auth" se il certificato client è opzionale ma verrà comunque validato, se presente.

```
<subsystem xmlns="urn:wildfly:elytron:xx" ....>
    <providers>
        ...
    </sasl>
    <tls>
        <key-stores>
            <key-store name="applicationKS">
                ...
            </key-store>
            <key-store name="govwayExampleKeyStore">
                <credential-reference clear-text="changeit"/>
                <implementation type="JKS"/>
```

(continues on next page)

(continua dalla pagina precedente)

```
                <file path="/etc/govway/keys/govway_server.jks"/>
            </key-store>
            <key-store name="govwayExampleTrustStore">
                <credential-reference clear-text="changeit"/>
                <implementation type="JKS"/>
                <file path="/etc/govway/keys/govway_https_truststore.jks"/>
            </key-store>
        </key-stores>
        <key-managers>
            <key-manager name="applicationKM" ...>
                ...
            </key-manager>
            <key-manager name="govwayExampleKeyManager" key-store=
↪"govwayExampleKeyStore" alias-filter="aliasInKeystore">
                <credential-reference clear-text="changeit"/> <!--␣
↪password chiave privata -->
            </key-manager>
        </key-managers>
        <trust-managers>
            <trust-manager name="govwayExampleTrustManager" key-store=
↪"govwayExampleTrustStore"/>
        </trust-managers>
        <server-ssl-contexts>
            <server-ssl-context name="applicationSSC" .../>
            <server-ssl-context name="govwayExampleSSC" need-client-auth=
↪"true" key-manager="govwayExampleKeyManager" trust-manager=
↪"govwayExampleTrustManager"/>
        </server-ssl-contexts>
    </tls>
</subsystem>
```

- Il server ssl context creato deve essere associato ad un “https-listener”.

```
<https-listener name="httpsGovWay" socket-binding="httpsGovWaySB" ssl-
↪context="govwayExampleSSC"/>
```

- Si deve infine associare al socket-binding indicato nell’https listener una porta su cui l’application server gestisce le richieste https.

```
<socket-binding name="httpsGovWaySB" port="${jboss.https.port:8445}"/>
```

Nota

A partire dalla versione 25 di wildfly, nella configurazione di default è abilitato un application-security-domain “other” che rende obbligatoria la presenza di credenziali valide per invocare applicazioni web. Come indicato nella sezione *ApplicationSecurityDomain “other” su WildFly 25 o superiore*, poiché la gestione delle autorizzazioni avviene normalmente su GovWay si deve procedere a disabilitare l’application security domain commentandone la definizione all’interno della configurazione “undertow”:

```
<subsystem xmlns="urn:jboss:domain:undertow:x.0" ...>
 ...
    <application-security-domains>
       <!-- <application-security-domain name="other" security-domain=
↪"ApplicationDomain"/> -->
    </application-security-domains>
</subsystem>
```

Wildfly (security-realms)

Nota

La seguente sezione fornisce degli esempi utili ad attuare la configurazione https. Per conoscere maggiori dettagli e modalità di configurazioni differenti fare riferimento a quanto indicato nella documentazione ufficiale dell'Application Server Wildfly (http://wildfly.org).

La configurazione può essere attuata nel file standalone.xml che si trova all'interno della cartella "WILDFLY_HOME/standalone/configuration/".

- Deve prima essere definito un security realm contenente il certificato che il server deve esporre, aggiungendolo ai security realms esistenti.

```
<security-realms>
    <security-realm name="mySecurityRealm">
        <server-identities>
                <ssl>
                        <keystore path="/etc/govway/keys/govway_server.jks
↪" keystore-password="changeit"
                                alias="aliasInKeystore" key-password=
↪"changeit" />
                </ssl>
        </server-identities>
    </security-realm>
    ...
</security-realms>
```

  se oltre ad esporre un certificato server, si deve autenticare il certificato client del chiamante, la configurazione del security realm deve essere estesa con la definizione di un trustStore che contenga i certificati necessari a validarli.

```
<security-realms>
    <security-realm name="mySecurityRealmClientAuth">
        <server-identities>
                <ssl>
                        <keystore path="/etc/govway/keys/govway_https_
↪server.jks" keystore-password="changeit"
                                alias="aliasInKeystore" key-password=
↪"changeit" />
                </ssl>
        </server-identities>
        <authentication>
```

(continues on next page)

(continua dalla pagina precedente)

```
                <truststore path="/etc/govway/keys/govway_https_truststore.
↪jks" keystore-password="changeit"/>
            </authentication>
        </security-realm>
        ...
    </security-realms>
```

- Il security realm creato deve essere associato ad un "https-listener".

```
<https-listener name="httpsGovWay" socket-binding="httpsGovWay" security-
↪realm="mySecurityRealm"/>
```

Per rendere obbligatorio che il chiamante debba fornire un proprio certificato client deve essere aggiunto l'attributo "verify-client" valorizzato con il valore "REQUIRED". Se tale attributo viene valorizzato invece con il valore "REQUESTED" il certificato client non è obbligatorio ma verrà comunque validato, se presente.

```
<https-listener name="httpsGovWayClientAuth" socket-binding=
↪"httpsGovWayClientAuth" security-realm="mySecurityRealmClientAuth" verify-
↪client="REQUIRED"/>
```

- Si deve infine associare al socket-binding indicato nell'https listener una porta su cui l'application server gestisce le richieste https.

```
<socket-binding name="httpsGovWayClientAuth" port="${jboss.https.port:8445}
↪"/>
```

Tomcat

Nota

La seguente sezione fornisce degli esempi utili ad attuare la configurazione https. Per conoscere maggiori dettagli e modalità di configurazioni differenti fare riferimento a quanto indicato nella documentazione ufficiale dell'Application Server Apache Tomcat (http://tomcat.apache.org).

La configurazione può essere attuata nel file server.xml che si trova all'interno della cartella "TOMCAT_HOME/conf".

Deve essere definito un connettore contenente il certificato che il server deve esporre e la porta su cui deve gestire le richieste https.

```
<Connector port="8445" protocol="HTTP/1.1" SSLEnabled="true"
    strategy="ms" maxHttpHeaderSize="8192"
    emptySessionPath="true"
    scheme="https" secure="true" clientAuth="false" sslProtocol = "TLS"
    keyAlias="aliasInKeystore"
    keystoreFile="/etc/govway/keys/govway_https_server.jks"
    keystorePass="changeit"/>
```

Nota

Nell'esempio fornito la passowrd della chiave privata del certificato server deve coincidere con la password del keystore.

Per rendere obbligatorio che il chiamante debba fornire un proprio certificato client:

- deve essere abilitato l'attributo "clientAuth".

```
<Connector port="8445" ... clientAuth="true"  .../>
```

- deve essere fornito un trustStore che contenga i certificati necessari a validarle i certificati client ricevuti. Il trustStore deve essere fornito attraverso le proprietà java "javax.net.ssl.trustStore" e "javax.net.ssl.trustStorePassword". Per farlo è possibile ad esempio aggiungere la seguente riga al file "TOMCAT_HOME/bin/setenv.sh" (creare il file se non esiste):

```
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=/etc/govway/keys/govway_
↪https_truststore.jks -Djavax.net.ssl.trustStorePassword=changeit"
```

Frontend HTTP

Nel caso in cui la terminazione ssl viene gestita su un frontend http (Apache httpd, IIS, etc) GovWay necessita di ricevere i certificati client per attuare il processo di autenticazione https.

Nel caso di utilizzo di una integrazione "mod_jk" tra frontend e application server, GovWay riceve i certificati gestiti sul frontend http in maniera trasparente e non sono richieste ulteriori configurazioni.

Negli altri casi invece deve essere configurato opportunamente il frontend http per inoltrare i certificati client o il DN attraverso header HTTP a GovWay. Si rimanda alla documentazione ufficiale del frontend utilizzato su come attivare tale funzionalità. Di seguito invece vengono fornite indicazioni su come configurare GovWay per recepire le informazioni dagli header inoltrati dal frontend.

Integrazione Frontend - GovWay

Nota

Gli esempi forniti descrivono una configurazione valida per le erogazioni. È sufficiente utilizzare il prefisso "org.openspcoop2.pdd.services.pd." invece di "org.openspcoop2.pdd.services.pa." per adeguare la configurazione alle fruizioni.

Per abilitare il processamento degli header inoltrati dal frontend è necessario editare il file <directory-lavoro>/govway_local.properties .

1. Abilitare la proprietà "org.openspcoop2.pdd.services.pa.gestoreCredenziali.enabled"

```
# Mediazione tramite WebServer (Erogazioni)
org.openspcoop2.pdd.services.pa.gestoreCredenziali.enabled=true
# Nome del WebServer che media le comunicazioni https con GovWay
org.openspcoop2.pdd.services.pa.gestoreCredenziali.nome=#FRONTEND-NAME#
```

   inserendo al posto di #FRONTEND-NAME# il nome associato al frontend che verrà utilizzato nella diagnostica di GovWay.

2. Se il frontend inserisce in un header http il DN del Subject e/o dell'Issuer relativo ai certificati client autenticati, deve essere indicato il nome di tali header tramite la seguente configurazione:

```
# DN del Subject e dell'Issuer tramite gli header:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.subject=#SUBJECT_
↪HEADER-NAME#
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.issuer=#ISSUER_HEADER-
↪NAME#
```

inserendo al posto di #SUBJECT_HEADER-NAME# il nome dell'header http utilizzato per propagare il DN del Subject (es. "SSL_CLIENT_S_DN") e al posto di #ISSUER_HEADER-NAME# il nome dell'header http utilizzato per propagare il DN dell'Issuer (es. SSL_CLIENT_I_DN). È possibile anche attuare una configurazione dove viene processato solamente il Subject, lasciando commentata la proprietà relativa all'Isssuer.

3. Nel caso il frontend inserisca in un header http il certificato x.509 del client autenticato (es. "SSL_CLIENT_CERT"), deve essere indicato il nome dell'header, inserendolo al posto di #CLIENT-CERT_HEADER-NAME#, nella seguente configurazione:

```
# Certificato tramite l'header:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate=#CLIENT-
↪CERT_HEADER-NAME#
```

Il certificato inserito nell'header http dal frontend può essere stato codificato in base64/hex e/o tramite url encoding. È possibile effettuare la corretta decodifica configurando le seguenti proprietà:

```
# Indicazione se l'header valorizzato con il certificato è url encoded:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.url_
↪decode=false
# Indicazione se l'header valorizzato con il certificato è base64 encoded:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.base64_
↪decode=false
# Indicazione se l'header valorizzato con il certificato è hex encoded:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.hex_
↪decode=false
# Abilitando la seguente opzione, l'header valorizzato con il certificato può essere␣
↪url encoded o base64 encoded o hex encoded (verranno provate tutte le␣
↪decodifiche):
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.url_
↪decode_or_base64_decode_or_hex_decode=false
```

Nota

La proprietà "url_decode_or_base64_decode_or_hex_decode" è abilitata per default: deve essere definita con il valore "false" per poterla disabilitare.

Nota

Se viene abilitata la proprietà "url_decode_or_base64_decode_or_hex_decode", viene provata la decodificata del certificato ricevuto prima tramite urlDecode e successivamente, solamente se la decodifica non va a buon fine, viene provata la modalità base64 e infine la modalità hex. Le tre modalità vengono quindi utilizzate in alternativa. Se si desidera effettuare entrambe le decodifiche sul certificato (prima urlDecode e a seguire base64Decode o hexDecode), deve essere disabilita la proprietà "url_decode_or_base64_decode_or_hex_decode" e devono essere abilitate le due modalità singole "url_decode" e "base64_decode" o "hex_decode".

Sono inoltre disponibili ulteriori opzioni che consentono di gestire eventuali codifiche personalizzate attuate dal web server http come ad es. avviene tramite "apache" utilizzando la regola "RequestHeader set SSL_CLIENT_CERT «%{SSL_CLIENT_CERT}s» «expr=-n %{SSL_CLIENT_CERT}»" che comporta l'inoltro di un certificato PEM su unica linea, in cui i ritorni a capo vengono sostituiti tramite spazi. Per supportare questa modalità è possibile utilizzare la seguente configurazione:

```
# Disabilito tutte le precedenti decodifiche:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.url_
↪decode=false
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.base64_
↪decode=false
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.hex_
↪decode=false
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.url_
↪decode_or_base64_decode_or_hex_decode=false
# Abilito la conversione dei caratteri:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪replaceCharacters=true
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪replaceCharacters.source=\\s
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪replaceCharacters.dest=\\n
```

Nota

Si suggerisce comunque di cercare di configurare il frontend per inoltrare il certificato tramite una delle modalità di codifica standard sopra indicate: base64, hex o urlEncoded. Ad esempio su "apache" la regola sopra indicata può essere rivista in questa forma: "RequestHeader set SSL_CLIENT_CERT «expr=%{base64:%{SSL_CLIENT_CERT}s}» «expr=-n %{SSL_CLIENT_CERT}»"

La modalità di ricezione di un certificato x.509 in un header HTTP consente anche di definire un truststore per verificare nuovamente i certificati ricevuti dal FrontEnd tramite la seguente configurazione:

```
# TrustStore per verificare i certificati ricevuti
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪truststore.path=PATH_ASSOLUTO_FILE_SYSTEM
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪truststore.type=jks
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪truststore.password=changeme
```

Se è stata abilitata la verifica viene controllato che il certificato ricevuto sia stato emesso da CA presenti nel trustStore o sia lui stesso direttamente presente nel trustStore. Oltre alla verifica è possibile abilitare la validazione del certificato rispetto alla scadenza.

```
# Indicazione se deve essere verificata la scadenza dei certificati verificati␣
↪(default:true)
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪truststore.validityCheck=false
```

È inoltre possibile indicare delle CRLs per verificare se un certificato risultato revocato. Con la presenza di CRLs la validazione rispetto alla scadenza viene effettuata per default e non serve abilitarla puntualmente.

```
# Elenco delle CRL per verificare i certificati ricevuti
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪truststore.crls=PATH1.crl,PATH2.crl...
```

In alternativa alla validazione tramite CRL è possibile associare una policy OCSP indicando uno dei tipi registrati nel file *<directory-lavoro>/ocsp.properties* come proprietà "ocsp.<idPolicy>.type"; per ulteriori dettagli si

rimanda alle sezioni *Online Certificate Status Protocol (OCSP)* e *Configurazione Policy OCSP*.

```
# Policy OCSP utilizzata per verificare i certificati ricevuti
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪truststore.ocspPolicy=INDICARE_TIPO_POLICY
```

Le seguenti proprietà consentono di gestire il caso in cui il web server frontend, anche quando il chiamante non presenta un certificato client, inoltri comunque l'header specificato nella proprietà "org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate".

- org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.none: permette di definire la keyword utilizzata dal web server frontend per indicare che il fruitore non ha presentato alcun certificato;
- org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.ignoreEmpty: (boolean, default:true) se abilitata, un header valorizzato con una stringa vuota sarà interpretato come un'indicazione che il fruitore non ha presentato alcun certificato client.

```
# Le seguenti proprietà consentono di gestire il caso in cui il web server frontend,
↪ anche quando il chiamante non presenta un certificato client,
# inoltri comunque l'header specificato nella proprietà 'org.openspcoop2.pdd.services.
↪pa.gestoreCredenziali.header.ssl.certificate'.
# La seguente proprietà permette di definire la keyword utilizzata dal web server␣
↪frontend per indicare che il fruitore non ha presentato alcun certificato:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪none=COSTANTE_UTILIZZATA_INDICARE_NESSUN_CERTIFICATO_CLIENT
# Se la seguente proprietà è abilitata, un header valorizzato con una stringa vuota␣
↪sarà interpretato come un'indicazione che il fruitore non ha presentato alcun␣
↪certificato client.
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.ssl.certificate.
↪ignoreEmpty=true
```

4. Se il frontend inserisce in un header http il principal dell'identità relativa al chiamante, deve essere indicato il nome di tale header tramite la seguente configurazione:

```
# L'identità del chiamante può essere fornita dal WebServer anche come informazione
↪'principal' tramite il seguente header:
org.openspcoop2.pdd.services.pa.gestoreCredenziali.header.principal=#PRINCIPAL_
↪HEADER-NAME#
```

inserendo al posto di #PRINCIPAL_HEADER-NAME# il nome dell'header http utilizzato dal frontend.

5. Le credenziali, raccolte negli header precedentemente dichiarati, verranno utilizzate da GovWay per attuare i processi di autenticazione abilitati su ogni erogazione. La presenza obbligatoria o meno di credenziali veicolate tramite header http può essere abilitata tramite la seguente proprietà:

```
# - none: le richieste in arrivo possono non presentare alcun header che veicola␣
↪credenziali.
# - atLeastOne: le richieste in arrivo devono presentare almeno un header che␣
↪veicola credenziali.
# - ssl/principal: le richieste in arrivo devono presentare gli header richiesti␣
↪dalla modalità scelta, che è di fatto l'unica modalità di autenticazione poi␣
↪configurabile sulle erogazioni.
# Con la modalità 'none' o 'atLeastOne' è possibile usare il gestore davanti a␣
↪erogazioni con tipi di autenticazione differenti,
# delegando quindi alla singola erogazione il controllo che le credenziali attese␣
↪siano effettivamente presenti.
```

(continues on next page)

(continua dalla pagina precedente)

```
org.openspcoop2.pdd.services.pa.gestoreCredenziali.modalita=none/atLeastOne/ssl/
↪principal
```

6. È possibile abilitare l'autenticazione del frontend in modo da accettare gli header http contenenti le credenziali solamente da un frontend autenticato tramite la seguente configurazione:

```
# Modalità di autenticazione da parte di GovWay del webServer (none/ssl/basic/
↪principal)
org.openspcoop2.pdd.services.pd.gestoreCredenziali.autenticazioneCanale=none
# Credenziali attese da GovWay (a seconda della modalità di autenticazione␣
↪indicata) che identificano il webServer
#org.openspcoop2.pdd.services.pa.gestoreCredenziali.autenticazioneCanale.basic.
↪username=Username
#org.openspcoop2.pdd.services.pa.gestoreCredenziali.autenticazioneCanale.basic.
↪password=Password
#org.openspcoop2.pdd.services.pa.gestoreCredenziali.autenticazioneCanale.ssl.
↪subject=Subject
#org.openspcoop2.pdd.services.pa.gestoreCredenziali.autenticazioneCanale.
↪principal=Principal
```

Ogni parametro di configurazione descritto nei precedenti punti è personalizzabile in funzione del profilo di interoperabilità e del soggetto associato ad ogni dominio gestito. Di seguito vengono definite le varie modalità di ridefinizione nell'ordine dalla più generica alla più specifica, agendo dopo il prefisso "org.openspcoop2.pdd.services.pa.gestoreCredenziali." e prima del nome della proprietà:

- *org.openspcoop2.pdd.services.pa.gestoreCredenziali.<profilo>.PROPRIETA*

  consente di restringere la configurazione ad un determinato Profilo di Interoperabilità; "<profilo>" può assumere i valori "trasparente" (Profilo API Gateway), "modipa" (Profilo ModI), "spcoop" (Profilo SPCoop), "as4" (Profilo eDelivery), "sdi" (Profilo Fatturazione Elettronica). Esempio:

  ```
  org.openspcoop2.pdd.services.pa.gestoreCredenziali.spcoop.
  ↪nome=WebServerAutenticazioneSPCoop
  ```

- *org.openspcoop2.pdd.services.pa.gestoreCredenziali.<nomeSoggetto>.PROPRIETA*

  la configurazione indicata verrà utilizzata solamente per il soggetto interno indicato in "<nomeSoggetto>". Esempio:

  ```
  org.openspcoop2.pdd.services.pa.gestoreCredenziali.
  ↪EnteDominioInternoEsempio.nome=WebServerAutenticazioneSPCoop
  ```

- *org.openspcoop2.pdd.services.pa.gestoreCredenziali.<profilo>-<nomeSoggetto>.PROPRIETA*

  configurazione che consente di indicare il profilo di interoperabilità a cui appartiene il soggetto indicato, visto che un soggetto con lo stesso nome può essere registrato su profili differenti. Esempio:

  ```
  org.openspcoop2.pdd.services.pa.gestoreCredenziali.spcoop-
  ↪EnteDominioInternoEsempio.nome=WebServerAutenticazioneSPCoop
  ```

- *org.openspcoop2.pdd.services.pa.gestoreCredenziali.<tipoSoggetto>-<nomeSoggetto>.PROPRIETA*

  rispetto alle precedenti due proprietà è possibile indicare per il soggetto interno, indicato in "<nomeSoggetto>", anche il tipo (tipoSoggetto>. Questa opzione è utile nei profili di interoperabilità dove ai soggetti è possibile associare più tipi, come ad es. in SPCoop dove sono utilizzabili i tipi "spc", "aoo", "test". Esempio:

```
org.openspcoop2.pdd.services.pa.gestoreCredenziali.aoo-
↪EnteDominioInternoEsempio.nome=WebServerAutenticazioneSPCoop
```

- *org.openspcoop2.pdd.services.pa.gestoreCredenziali.<profilo>-<tipoSoggetto>-<nomeSoggetto>.PROPRIETA*

  rappresenta la configurazione più specifica possibile dove viene combinato sia il profilo di interoperabilità che il tipo e il nome del soggetto interno. Esempio:

  ```
  org.openspcoop2.pdd.services.pa.gestoreCredenziali.spcoop-aoo-
  ↪EnteDominioInternoEsempio.nome=WebServerAutenticazioneSPCoop
  ```

6.15.2 Comunicazioni in Uscita

La configurazione varia a seconda se la terminazione ssl è gestita direttamente sull'application server (wildfly o tomcat) o viene gestita da un reverse proxy.

Wildfly / Tomcat

Le comunicazioni in uscita utilizzano una configurazione ssl differente a seconda dell'impostazione utilizzata nei connettori configurati per ogni API.

GovWay consente di indicare esplicitamente, nella configurazione di un connettore, i keystore e truststore da utilizzare. Per questa modalità seguire le indicazioni riportate nella Guida alla Console di Gestione, nella sezione "Funzionalità Avanzate - Connettori" al paragrafo avanzate_connettori_https.

In alternativa, se viene solamente indicato un endpoint https senza fornire keystore specifici per l'API, GovWay eredita la configurazione https impostata nella JVM dell'Application Server per la quale viene fornito un esempio di configurazione.

Configurazione HTTPS della JVM

Nota

La seguente sezione fornisce degli esempi utili ad attuare la configurazione https. Per conoscere maggiori dettagli e modalità di configurazioni differenti fare riferimento a quanto indicato nella documentazione ufficiale della JVM e dell'Application Server utilizzato.

- Deve essere fornito un trustStore che contenga i certificati necessari a validare i certificati server ricevuti. Il trustStore deve essere fornito attraverso le proprietà java "javax.net.ssl.trustStore", "javax.net.ssl.trustStorePassword" e "javax.net.ssl.trustStoreType". Per farlo è possibile ad esempio aggiungere la seguente riga al file "TOMCAT_HOME/bin/setenv.sh" per Tomcat o al file "WILDFLY_HOME/bin/standalone.conf" per Wildfly:

  ```
  JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=/etc/govway/keys/govway_
  ↪https_truststore.jks -Djavax.net.ssl.trustStorePassword=changeit -Djavax.
  ↪net.ssl.trustStoreType=jks"
  ```

- Deve essere fornito un keyStore che contenga il certificato client utilizzato da GovWay. Il keyStore deve essere fornito attraverso le proprietà java "javax.net.ssl.keyStore", "javax.net.ssl.keyStorePassword" e "javax.net.ssl.keyStoreType". Per farlo è possibile ad esempio aggiungere la seguente riga al file "TOMCAT_HOME/bin/setenv.sh" per Tomcat o al file "WILDFLY_HOME/bin/standalone.conf" per Wildfly:

```
JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=/etc/govway/keys/govway_
↪https_keystore.p12 -Djavax.net.ssl.keyStorePassword=changeit -Djavax.net.
↪ssl.keyStoreType=pkcs12"
```

Nota

La passowrd della chiave privata del certificato client deve coincidere con la password del keystore.

Reverse Proxy

GovWay consente di gestire correttamente le situazioni in cui le comunicazioni tra il gateway e l'endpoint destinatario siano mediate dalla presenza di un proxy.

Nei casi più comuni si tratta di un «forward proxy». In questi casi l'indirizzo del proxy può essere censito sul connettore dell'Erogazione. Per questa modalità seguire le indicazioni riportate nella Guida alla Console di Gestione, nella sezione "Funzionalità Avanzate - Connettori - Proxy" al paragrafo avanzate_connettori_proxy.

In scenari più complessi possono essere presenti reverse proxy che intervengono nella gestione delle connessioni https, utilizzando certificati client e/o trustStore differenti per diversi contesti applicativi. In queste situazioni l'endpoint indicato nella configurazione del connettore su GovWay non è l'indirizzo remoto dell'applicativo ma bensì l'indirizzo del reverse proxy il quale a sua volta si occuperà di inoltrare la richiesta agli indirizzi a lui noti.

In questa situazione, è necessario configurare gli endpoint delle API sia su GovWay (indirizzo del reverse proxy), che sul reverse proxy (indirizzo dell'Erogatore finale).

Per semplificare la gestione di questo scenario architetturale è possibile passare l'indirizzo remoto dell'applicativo al proxy tramite un header HTTP o un parametro della url. In questo modo il censimento degli applicativi viene effettuato esclusivamente su GovWay. Per questa modalità seguire le indicazioni riportate nella Guida alla Console di Gestione, nella sezione "Funzionalità Avanzate - Gestione Proxy" al paragrafo avanzate_govway_proxy.

6.16 Integrazione delle Console con IdM esterno

Dopo aver completato l'installazione e il dispiegamento del software, come mostrato nella sezione *Verifica dell'Installazione*, le console di gestione e monitoraggio sono accessibili utilizzando le credenziali fornite durante l'esecuzione dell'installer.

La gestione delle utenze applicative, descritta nella sezione utenti della Guida alla Console di Gestione, può essere effettuata tramite l'utenza "amministratore" associata alla "govwayConsole", così come definita durante l'esecuzione dell'installer. Tale utenza permette di creare nuove utenze e gestire i permessi e le password associate alle utenze esistenti.

Per default, l'autenticazione da parte delle console di gestione e monitoraggio avviene localmente utilizzando le password assegnate in tal modo alle utenze tramite la "govwayConsole".

In alternativa, è possibile delegare l'autenticazione ad un IdM esterno, intervenendo sulle configurazioni delle Console come documentato di seguito.

Nota

Abilitando l'autenticazione tramite IdM esterno, la gestione delle utenze applicative avviene sempre tramite la "govwayConsole", associando ad ogni utenza i "principal" ottenuti dall'IdM. Non saranno invece più gestite le password associate alle utenze.

Per attivare l'autenticazione delle utenze tramite un IdM esterno è necessario intervenire sui seguenti file di configurazione:

- per la "govwayConsole": *<directory-lavoro>/console_local.properties*;
- per la "govwayMonitor": *<directory-lavoro>/monitor_local.properties*

attuando le seguenti modifiche:

1. Disabilitare l'autenticazione locale delle utenze:

```
login.application=false
```

2. Disabilitare la possibilità di effettuare logout:

```
logout.mostraButton.enabled=false
```

> **Nota**
>
> Su contesti OIDC/OAuth la proprietà può essere lasciata abilitata per effettuare l'invocazione dell'operazione di logout esposta dall'authorization server.

3. Definire la modalità di accesso all'identità dell'utenza autenticata tramite l'IdM esterno, utilizzando una delle seguenti modalità supportate:

   - Utilizzando il tipo "header" l'identità viene acquisita tramite un header http il cui nome è definito nella proprietà "login.props.header", come indicato di seguito:

   ```
   login.tipo=header
   login.props.header=<HTTP-HEADER-NAME>
   ```

   - Utilizzando il tipo "principal" l'identità viene acquisita tramite la api "javax.servlet.http.HttpServletRequest.getUserPrincipal()".

   ```
   login.tipo=principal
   ```

   La modalità "principal" richiede che l'autenticazione degli utenti sia stata preliminarmente configurata a livello del container dell'application server che ospita le console.

   - Utilizzando il tipo "oauth2" l'identità viene acquisita tramite tramite il token di accesso (access token) rilasciato dal provider OAuth2, che contiene o riferisce le informazioni sull'utente. Maggiori informazioni vengono fornite nella sezione *Integrazione delle Console con un Authorization Server tramite OIDC/OAuth*.

   > **Nota**
   >
   > L'autenticazione con modalità OAuth2 può essere utilizzata contemporaneamente alla modalità di autenticazione interna (login.application=true). In questo caso, gli utenti possono scegliere di autenticarsi tramite OAuth2 o utilizzando le credenziali locali gestite dalla console.

   ```
   login.tipo=oauth2
   # Url dell'authorization server
   login.props.oauth2.authorization.endpoint=
   # Url dove richiedere il token
   login.props.oauth2.token.endpoint=
   ```

(continues on next page)

(continua dalla pagina precedente)

```
# Url dove scaricare le indormazioni utente
login.props.oauth2.userInfo.endpoint=
# URL di validazione dei certificati JWT
login.props.oauth2.jwks.endpoint=
# URL del servizio logout federato
login.props.oauth2.logout.endpoint=
# Client ID rilasciato dal server OAuth2
login.props.oauth2.clientId=
# URL dove viene rediretto l'utente dopo l'autenticazione
login.props.oauth2.redirectUri=
# Scope dell'autenticazione
login.props.oauth2.scope=
# Nome del claim da dove leggere il principal
login.props.oauth2.principalClaim=
# Sorgente da cui leggere il principal: userinfo (default), id_token, access_
↪token
#login.props.oauth2.userInfo.source=userinfo

# Parametri timeout connessione verso il server OAuth2
#login.props.oauth2.readTimeout=15000
#login.props.oauth2.connectTimeout=10000

# Validazione dei claim del token
# Inserire una riga per ogni claim da validare nella forma: login.props.oauth2.
↪claims.validation.claimName=claimValues (lista di valori separati da virgola)
#login.props.oauth2.claims.validation.claimName=claimValue1,claimValue2,...

# Truststore https
#login.props.oauth2.https.hostnameVerifier=true
#login.props.oauth2.https.trustAllCerts=false
#login.props.oauth2.https.trustStore=PATH
#login.props.oauth2.https.trustStore.password=changeme
#login.props.oauth2.https.trustStore.type=jks
#login.props.oauth2.https.trustStore.crl=PATH

# Keystore https
#login.props.oauth2.https.keyStore=PATH
#login.props.oauth2.https.keyStore.password=changeme
#login.props.oauth2.https.keyStore.type=jks
#login.props.oauth2.https.key.alias=mykey
#login.props.oauth2.https.key.password=changeme

# Log di debug della libreria OAuth2/OIDC (default: false). Non abilitare in␣
↪produzione:
# il log puo' contenere token di sessione e claim con informazioni personali␣
↪sensibili.
#login.props.oauth2.debug=false
```

- È infine possibile configurare una modalità custom indicando una classe che implementi l'interfaccia "org.openspcoop2.utils.credential.IPrincipalReader". Eventuali proprietà di configurazione da fornire alla classe possono essere indicate nella forma "login.props.<NOME_PROP>=<VALORE_PROP>".

```
login.tipo=org.example.packageCustom.CustomPrincipalReader
login.props.nomeProp1=val1
...
login.props.nomePropN=valN
```

4. Nel caso le console siano integrate all'interno di altre applicazioni o portali, è possibile ridefinire le url alle quali vada rediretto l'utente nei casi di autorizzazione di accesso negata. Lasciando le proprietà non valorizzate verranno utilizzati le pagine di default previste dall'applicazione.

```
# Errore interno durante il login
login.erroreInterno.redirectUrl=
# Autorizzazione negata
login.utenteNonAutorizzato.redirectUrl=
# Utenza non valida
login.utenteNonValido.redirectUrl=
# Sessione scaduta
login.sessioneScaduta.redirectUrl=
# Pagina successiva all'operazione di logout
logout.urlDestinazione=
```

6.16.1 Integrazione delle Console con un Authorization Server tramite OIDC/OAuth

La presente sezione fornisce i dettagli necessari alla configurazione delle console per l'integrazione con un authorization server conforme agli standard OpenID Connect (OIDC) e OAuth 2.0.

Come riferimento viene utilizzato l'authorization server Keycloak.

Configurazione di esempio (scenario demo)

Per un rapido scenario dimostrativo, Keycloak è stato configurato come segue:

- Creazione di un realm denominato "GovWay".
- Creazione di un client con client_id e name impostati a "govway_console", associato all'Authentication Standard Flow, per implementare l'Authorization Code Flow previsto dallo standard OpenID Connect. Sono state definite come valid redirect URIs le seguenti URL:
  - http://127.0.0.1:8080/govwayConsole/oauth2/callback
  - http://127.0.0.1:8080/govwayConsole/login.do*
- Creazione di un secondo client con client_id e name impostati a "govway_monitor", anch'esso configurato con lo Standard Flow. Sono state definite come valid redirect URIs le seguenti URL:
  - http://127.0.0.1:8080/govwayMonitor/oauth2/callback
  - http://127.0.0.1:8080/govwayMonitor/public/*
- Creazione di due utenze applicative utilizzate come utenze di default per le console:
  - amministratore
  - operatore
- Creazione delle ulteriori utenze applicative i cui username devono corrispondere a quelli registrati nella console di gestione.

Configurazione lato Console

Per abilitare l'autenticazione tramite Keycloak è necessario intervenire sui seguenti file di configurazione:

- Console di gestione (govwayConsole): *<directory-lavoro>/console_local.properties*;

- Console di monitoraggio (govwayMonitor): *<directory-lavoro>/monitor_local.properties*

1. Disabilitazione dell'autenticazione locale:

```
login.application=false
```

2. Definizione della modalità di accesso all'identità autenticata tramite Keycloak:

```
login.tipo=oauth2
```

3. Configurazione delle URL di Keycloak per la negoziazione dei token:

```
# Url di autenticazione
login.props.oauth2.authorization.endpoint=https://localhost:8543/realms/
↪GovWay/protocol/openid-connect/auth
# Url dove richiedere il token
login.props.oauth2.token.endpoint=https://localhost:8543/realms/GovWay/
↪protocol/openid-connect/token
# Url dove scaricare le indormazioni utente
login.props.oauth2.userInfo.endpoint=https://localhost:8543/realms/GovWay/
↪protocol/openid-connect/userinfo
# URL di validazione dei certificati JWT
login.props.oauth2.jwks.endpoint=https://localhost:8543/realms/GovWay/
↪protocol/openid-connect/certs
# URL del servizio logout federato
login.props.oauth2.logout.endpoint=https://localhost:8543/realms/GovWay/
↪protocol/openid-connect/logout
```

Nota

L'endpoint "login.props.oauth2.userInfo.endpoint" è richiesto solo se il principal viene letto dall'endpoint */userinfo* (impostazione di default "login.props.oauth2.userInfo.source=userinfo"). Qualora il principal venga invece estratto direttamente dal token JWT (modalità "id_token" o "access_token", vedi punto 6), tale endpoint non viene utilizzato e può essere omesso.

4. Impostazione del Client ID e della Redirect URI (/oauth2/callback). Per la console di gestione (govway_console):

```
# Client ID rilasciato dal server OAuth2
login.props.oauth2.clientId=govway_console
# URL dove viene rediretto l'utente dopo l'autenticazione
login.props.oauth2.redirectUri=http://127.0.0.1:8080/govwayConsole/oauth2/
↪callback
```

Per la console di monitoraggio (govway_monitor):

```
# Client ID rilasciato dal server OAuth2
login.props.oauth2.clientId=govway_monitor
# URL dove viene rediretto l'utente dopo l'autenticazione
login.props.oauth2.redirectUri=http://127.0.0.1:8080/govwayMonitor/oauth2/
↪callback
```

5. Definizione dello Scope e del Claim contenente lo username:

```
# Scope dell'autenticazione
login.props.oauth2.scope=openid
# Nome del claim da dove leggere il principal
login.props.oauth2.principalClaim=preferred_username
```

6. Sorgente da cui leggere il principal (opzionale). Per default il claim contenente il principal (definito in "login.props.oauth2.principalClaim") viene ricercato nella risposta dell'endpoint */userinfo* dell'authorization server (proprietà "login.props.oauth2.userInfo.endpoint"). È possibile modificare tale comportamento tramite la proprietà "login.props.oauth2.userInfo.source", che accetta i seguenti valori:

   - **userinfo** (default): il principal viene ricercato nella risposta dell'endpoint */userinfo*, recuperata tramite chiamata HTTP;
   - **id_token**: il principal viene ricercato nel payload del JWT "id_token" restituito dall'authorization server, senza alcuna chiamata HTTP aggiuntiva;
   - **access_token**: il principal viene ricercato nel payload del JWT "access_token", senza alcuna chiamata HTTP aggiuntiva.

   ```
   # Sorgente da cui leggere il principal dell'utente loggato.
   # Valori ammessi: userinfo (default), id_token, access_token
   login.props.oauth2.userInfo.source=userinfo
   ```

   Nota

   Le modalità **id_token** e **access_token** sono utili nei casi in cui l'endpoint */userinfo* non sia disponibile oppure non restituisca il claim da utilizzare come principal dell'utente.

7. Validazione opzionale dei claim nel token. È possibile abilitare controlli aggiuntivi sui claim presenti nel token:

   ```
   # Validazione dei claim del token
   # Inserire una riga per ogni claim da validare nella forma: login.props.
   ↪oauth2.claims.validation.claimName=claimValues (lista di valori separati␣
   ↪da virgola)
   #login.props.oauth2.claims.validation.claimName=claimValue1,claimValue2,...
   login.props.oauth2.claims.validation.iss=https://localhost:8543/realms/
   ↪GovWay
   login.props.oauth2.claims.validation.aud=account
   ```

   È possibile validare anche claim annidati all'interno di oggetti JSON, utilizzando la notazione con il punto per navigare la struttura. Ad esempio, dato un token contenente:

   ```
   {
     ...
     "realm_access": {
       "roles": ["offline_access", "default-roles-govway", "uma_authorization"]
     },
     "resource_access": {
       "account": {
         "roles": ["manage-account", "manage-account-links", "view-profile"]
       }
     }
   }
   ```

Per i claim il cui valore è un array, il confronto esatto potrebbe non essere sufficiente. È possibile utilizzare un prefisso speciale nel valore della proprietà per attivare una logica di confronto basata sugli elementi dell'array:

- **[or]** oppure **[]**: almeno uno dei valori indicati deve essere presente nell'array del claim (logica OR);
- **[and]**: tutti i valori indicati devono essere presenti nell'array del claim (logica AND).

Esempio:

```
# Verifica che il claim 'realm_access.roles' contenga almeno uno tra i␣
↪valori indicati (OR)
login.props.oauth2.claims.validation.realm_access.roles=[]offline_access,
↪default-roles-govway

# Verifica che il claim 'resource_access.account.roles' contenga tutti i␣
↪valori indicati (AND)
login.props.oauth2.claims.validation.resource_access.account.
↪roles=[and]manage-account,manage-account-links
```

8. Parametri di connessione verso Keycloak:

```
# Parametri timeout connessione verso il server OAuth2
#login.props.oauth2.readTimeout=15000
#login.props.oauth2.connectTimeout=10000

# Truststore https
#login.props.oauth2.https.hostnameVerifier=true
#login.props.oauth2.https.trustAllCerts=false
#login.props.oauth2.https.trustStore=PATH
#login.props.oauth2.https.trustStore.password=changeme
#login.props.oauth2.https.trustStore.type=jks
#login.props.oauth2.https.trustStore.crl=PATH

# Keystore https
#login.props.oauth2.https.keyStore=PATH
#login.props.oauth2.https.keyStore.password=changeme
#login.props.oauth2.https.keyStore.type=jks
#login.props.oauth2.https.key.alias=mykey
#login.props.oauth2.https.key.password=changeme
```

9. Abilitazione di PKCE (Proof Key for Code Exchange) per maggiore sicurezza. PKCE è un'estensione di OAuth 2.0 (RFC 7636) che migliora la sicurezza del flusso Authorization Code, particolarmente raccomandata per client pubblici e applicazioni web:

```
# Abilita PKCE (Proof Key for Code Exchange) - RFC 7636
login.props.oauth2.pkce.enabled=true

# Metodo PKCE da utilizzare (valori: S256, plain)
# - S256: usa SHA-256 hash del code_verifier (RACCOMANDATO, default)
# - plain: usa il code_verifier direttamente (solo se il server non␣
↪supporta S256)
login.props.oauth2.pkce.method=S256
```

Nota

PKCE protegge contro attacchi di intercettazione del codice di autorizzazione. Il metodo **S256** (default) è raccomandato in quanto più sicuro. Il metodo **plain** dovrebbe essere usato solo se l'authorization server non supporta SHA-256.

10. Abilitazione del log di debug (opzionale). Per facilitare la diagnosi della configurazione in fase di set-up è possibile abilitare la produzione di log di debug aggiuntivi nei vari punti del flusso OAuth2/OIDC (token ricevuto dall'authorization server, informazioni utente prelevate dalla sorgente configurata, ecc.):

```
# Abilita i log di debug della libreria OAuth2/OIDC (default: false)
login.props.oauth2.debug=true
```

Avvertimento

Il log di debug può contenere token di sessione e claim con informazioni personali sensibili. Si raccomanda di non abilitarlo in ambienti di produzione.

6.17 Richieste "application/x-www-form-urlencoded" su WildFly

Per poter gestire correttamente richieste con Content-Type "application/x-www-form-urlencoded" su application server "WildFly", è richiesto di abilitare l'attributo "allow-non-standard-wrappers" nell'elemento "servlet-container" della configurazione di WildFly (es. in standalone/configuration/standalone.xml).

```
<servlet-container name="default" allow-non-standard-wrappers="true">
        ...
</servlet-container>
```

In assenza della configurazione sopra indicata, una richiesta "form-urlencoded" provoca un errore inatteso.

L'esempio seguente riporta l'errore che si ottiene non abilitando l'attributo "allow-non-standard-wrappers":

```
curl -v -d "param1=value1&param2=value2" -X POST http://127.0.0.1:8080/govway/
↪EnteTest/api-config/v1

HTTP/1.1 500 Internal Server Error
Connection: keep-alive
Content-Type: text/html;charset=UTF-8
Content-Length: 11543
Date: Tue, 07 Jul 2020 16:13:25 GMT

<html><head><title>ERROR</title><style>
body {
    font-family: "Lucida Grande", "Lucida Sans Unicode", "Trebuchet MS",␣
↪Helvetica, Arial, Verdana, sans-serif;
    margin: 5px
    ...
</head><body><div class="header"><div class="error-div"></div><div class=
↪"error-text-div">Error processing request</div>
</div><div class="label">Context Path:</div><div class="value">/govway</div>
↪<br/><div class="label">Servlet Path:</div>
<div class="value"></div><br/><div class="label">Path Info:</div><div class=
↪"value">/EnteTest/api-config/v1</div><br/>
```

(continues on next page)

(continua dalla pagina precedente)

```
<div class="label">Query String:</div><div class="value">null</div><br/><div␣
↪class="label">Stack Trace:</div>
<div class="value"></div><br/><pre>java.lang.IllegalArgumentException:␣
↪UT010023: Request org.openspcoop2.pdd.services.connector.
↪FormUrlEncodedHttpServletRequest@76dd2491 was not original or a wrapper
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.handlers.FilterHandler
↪$FilterChainImpl.doFilter(FilterHandler.java:116)
 at deployment.govway.ear//org.openspcoop2.pdd.services.connector.
↪FormUrlEncodedFilter.doFilter(FormUrlEncodedFilter.java:75)
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.core.ManagedFilter.
↪doFilter(ManagedFilter.java:61)
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.handlers.FilterHandler
↪$FilterChainImpl.doFilter(FilterHandler.java:131)
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.handlers.
↪FilterHandler.handleRequest(FilterHandler.java:84)
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.handlers.security.
↪ServletSecurityRoleHandler.handleRequest(ServletSecurityRoleHandler.java:62)
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.handlers.ServletChain
↪$1.handleRequest(ServletChain.java:68)
 at io.undertow.servlet@2.1.0.Final//io.undertow.servlet.handlers.
↪ServletDispatchingHandler.handleRequest(ServletDispatchingHandler.java:36)
 ...
 at org.jboss.threads@2.3.3.Final//org.jboss.threads.EnhancedQueueExecutor
↪$ThreadBody.doRunTask(EnhancedQueueExecutor.java:1486)
 at org.jboss.threads@2.3.3.Final//org.jboss.threads.EnhancedQueueExecutor
↪$ThreadBody.run(EnhancedQueueExecutor.java:1377)
 at java.base/java.lang.Thread.run(Thread.java:834)
</pre></body></html>
```

6.18 ApplicationSecurityDomain “other” su WildFly 25 o superiore

A partire dalla versione 25 di wildfly, nella configurazione di default è abilitato un application-security-domain “other” che rende obbligatoria la presenza di credenziali valide per invocare applicazioni web in generale e quindi anche i contesti “govway”.

Questo comporta che qualsiasi invocazione effettuata verso GovWay provoca un errore inatteso:

```
curl -u test:123456 -v -k http://127.0.0.1:8080/govway/APITest/v1

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="127.0.0.1:8080"
Content-Type: text/html;charset=UTF-8
Content-Length: 71
Date: Thu, 14 Oct 2021 10:03:51 GMT

<html><head><title>Error</title></head><body>Unauthorized</body></html>
```

Poichè la gestione delle autorizzazioni deve invece avvenire su GovWay (tramite il Controllo degli Accessi), si deve procedere a disabilitare l’application security domain commentandone la definizione all’interno della configurazione “undertow”:

```
<subsystem xmlns="urn:jboss:domain:undertow:x.0" ...>
 ...
     <application-security-domains>
        <!-- <application-security-domain name="other" security-domain=
↪"ApplicationDomain"/> -->
     </application-security-domains>
</subsystem>
```

6.19 Cache

GovWay utilizza cache che mantengono i dati di configurazioni acceduti, i keystore e i certificati, il risultato dei processi di validazione, autenticazione, autorizzazione e altri aspetti minori.

Ogni funzionalità è associata ad una cache dedicata per la quale i parametri configurabili sono:

- Stato (abilitato/disabilitato): consente di disabilitare l'utilizzo della cache;
- Dimensione (Elementi): numero massimo di elementi che possono risiedere in cache;
- Algoritmo (LRU/MRU): politica di eliminazione degli elementi quando si raggiunge la massima dimensione consentita;
- Item Life Time (Secondi): indica l'intervallo temporale massimo in cui un elemento può risiedere in cache;
- Item Idle Time (Secondi): indica l'intervallo temporale massimo in cui un elemento può risiedere in cache senza mai essere acceduto;

I parametri di ogni cache sono configurabili accedendo alla console di gestione (govwayConsole) in modalità avanzata (modalitaAvanzata) nella sezione "Configurazione -> Cache" (Fig. 6.15).

Le cache utilizzate da govway e i valori di default associati sono i seguenti:

- *Registro API*: mantiene le configurazioni delle API accedute.
  - Stato: abilitato
  - Dimensione: 10000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 7200
  - Item Idle Time (Secondi): infinito
- *Configurazione del Gateway*: mantiene le configurazioni generali di GovWay.
  - Stato: abilitato
  - Dimensione: 10000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 7200
  - Item Idle Time (Secondi): infinito
- *Dati di Autorizzazione*: contiene i risultati dei processi di autorizzazione.
  - Stato: abilitato
  - Dimensione: 5000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 7200

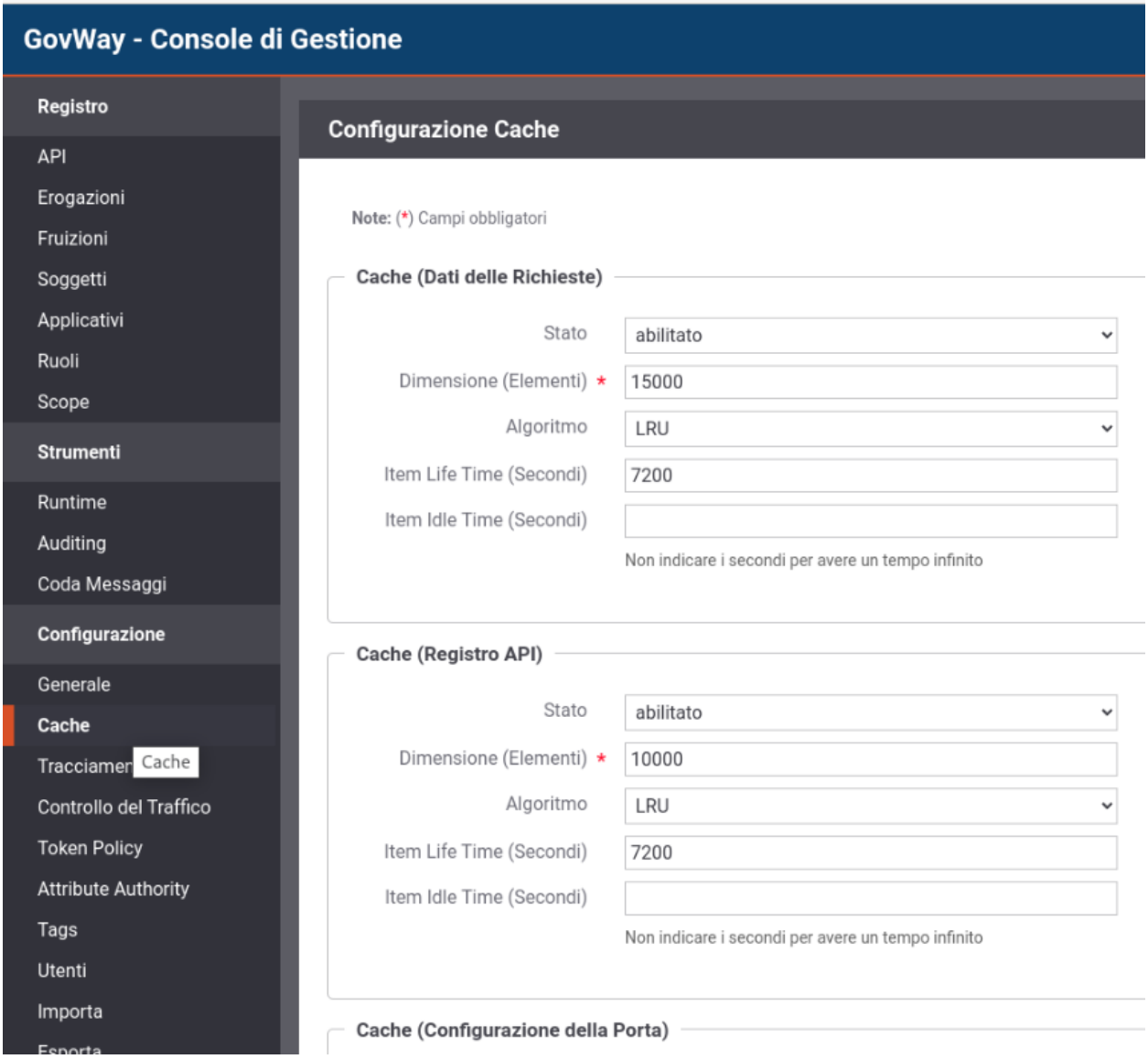

Figure6.15: GovWay Cache

  - Item Idle Time (Secondi): infinito

- *Dati di Autenticazione*: contiene i risultati dei processi di autenticazione.
  - Stato: abilitato
  - Dimensione: 5000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 7200
  - Item Idle Time (Secondi): infinito
- *Gestione dei Token*: mantiene i risultati dei processi di validazione dei token e i token negoziati.
  - Stato: abilitato
  - Dimensione: 5000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 600
  - Item Idle Time (Secondi): infinito
- *Attribute Authority*: mantiene i dai ottenuti consultando Attribute Authority.
  - Stato: abilitato
  - Dimensione: 5000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 7200
  - Item Idle Time (Secondi): infinito
- *Keystore*: contiene i keystore e i certificati acceduti, le CRL e i risultati delle validazioni tramite OCSP. A differenza delle altre cache, il parametro che indica l'intervallo temporale di validità dell'elemento differisce per le CRL e per il risultato della validazione tramite OCSP per poter consentire un intervallo minore.
  - Stato: abilitato
  - Dimensione: 10000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 7200
  - Item Idle Time (Secondi): infinito
  - CRL/OCSP Life Time (Secondi): 1800
- *Controllo Traffico - Dati Statistici*: mantiene i dati statistici utilizzati dalle politiche di Rate Limiting.
  - Stato: abilitato
  - Dimensione: 10000
  - Algoritmo: LRU
  - Item Life Time (Secondi): 300
  - Item Idle Time (Secondi): infinito
- *Cache Risposte*: cache dedicata alla funzionalità descritta nella sezione *Caching della Risposta - Disk Cache*. Non è possibile disabilitarla e nemmeno definire un intervallo di validità degli elementi in cache.
  - Stato: N.D.

  - Dimensione: 10000
  - Algoritmo: LRU
  - Item Life Time (Secondi): N.D.
  - Item Idle Time (Secondi): infinito

- *Cache (Load Balancer)*: cache dedicata alla funzionalità descritta nella sezione loadBalancerConnettore. Non è possibile disabilitarla e nemmeno definire un intervallo di validità degli elementi in cache.
  - Stato: N.D.
  - Dimensione: 10000
  - Algoritmo: LRU
  - Item Life Time (Secondi): N.D.
  - Item Idle Time (Secondi): N.D.

Cache di secondo livello

Per minimizzare gli accessi concorrenti alle varie cache e ottenere miglioramenti prestazionali viene utilizzata un'ulteriore cache di secondo livello denominata "Dati delle Richieste" che contiene tutti i dati principali raccolti durante la gestione della prima richiesta (API, configurazioni, keystore, politiche di rate limiting . . .).

- Stato: abilitato
- Dimensione: 10000
- Algoritmo: LRU
- Item Life Time (Secondi): 1800
- Item Idle Time (Secondi): infinito

Nota

Poichè la cache "Dati delle Richieste" consente di recuperare i dati normalmente presenti in altre cache di primo livello, si consiglia di impostare un intervallo temporale di validità della cache non superiore all'intervallo minimo configurato sulle seguenti cache di primo livello: Registro API, Configurazione del Gateway e Keystore.

6.20 Esposizione di Informazioni

Per limitare l'esposizione di informazioni relative all'architettura e alle tecnologie utilizzate, è possibile adottare le seguenti misure di sicurezza:

Application Server Tomcat

Se viene utilizzato Apache Tomcat come application server, è consigliabile modificare la configurazione per evitare che vengano esposte informazioni tecniche nei messaggi di errore.

Modificare il file **conf/server.xml** di Tomcat inserendo, all'interno del tag <Host>, il seguente valve:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve"
       showReport="false"
       showServerInfo="false"/>
```

Questa configurazione impedisce che vengano visualizzati:

- **showReport=»false»**: dettagli tecnici degli errori (stack trace, descrizioni dettagliate)

- **showServerInfo=»false»**: informazioni sulla versione del server Tomcat

CHAPTER 7

Esempio di setup del database PostgreSQL

Procedura indicativa, applicabile alla piattaforma RDBMS PostgreSQL, per la redisposizione del database di GovWay:

1. Creazione Utente

```
[user@localhost]$ su
Parola d'ordine: XXX
[root@localhost]# su - postgres
-bash-3.1$ createuser -P
Enter name of role to add: govway
Enter password for new role: govway
Conferma password: govway
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
CREATE ROLE
```

2. Creazione Database

```
[user@localhost]$ su
Parola d'ordine: XXX
[root@localhost]# su - postgres
-bash-3.1$createdb -E UTF8 -O govway govway
CREATE DATABASE
```

3. Abilitazione accesso dell'utente al Database, è possibile abilitare l'accesso editando il file */var/lib/pgsql/data/pg_hba.conf* (come super utente). Abilitiamo quindi l'utente govway ad accedere al db govway, aggiungendo le seguenti righe al file:

```
local govway govway md5
host govway govway 127.0.0.1 255.255.255.255 md5
```