

**Integrazione DocSuite con gestionali esterni – Versione semplificata**

## **Allegato Tecnico: Integrazione DocSuite con gestionali esterni**

**Fabrizio Lazzarotto**  
Product Manager

**Versione 1.5**  
**Del 15 Luglio 2020**

## Sommario

1. Introduzione .....	4
2. Requisiti di integrazione.....	4
3. Architettura dell'integrazione .....	5
Architettura complessiva di riferimento (DocSuite @BPM) .....	7
4. Scenari di integrazione.....	8
Visualizzazione External Viewer dei documenti di un protocollo/archivio/atto .....	8
Descrizione tecnica per il fornitore per integrare l'External Viewer.....	9
5. Documentazione WebApi pubbliche di integrazione .....	12
Descrizione schema generico di un comando delle WebAPI.....	12
ContentType StartWorkflowCommand: Descrizione del modello di avvio workflow .....	13
WebApi Descrizione del modello – CollaborationModel.....	15
WebApi Descrizione del modello – SignerModel.....	15
WebApi Descrizione del modello – CategoryModel .....	15
WebApi Descrizione del modello – ContainerModel .....	16
WebApi Descrizione del modello – ArchiveModel.....	16
WebApi Descrizione del modello – DocumentUnitModel .....	16
WebApi Descrizione del modello – MetadataModel.....	16
WebApi Descrizione del modello – FascicleModel .....	17
WebApi Descrizione del modello – ContactModel .....	17
WebApi Descrizione del modello – DocumentModel .....	18
WebApi Descrizione del modello – DocSuiteSectorModel.....	18
WebApi Descrizione del modello – IdentityContext .....	18
WebApi Descrizione del modello – RoleModel .....	19
WebApi Descrizione del modello – IdentityModel .....	19
WebApi Elenco delle parole chiave usato nei modelli – WorkflowParameterNames .....	20
Eventi di risposta delle attività di workflow.....	21

Tabella dei modelli degli eventi.....	24
6. Esempi dei modelli JSON di Avvio workflow ed eventi di risposta .....	25
Esempio 1 – Linguaggio C# - Avvio Workflow Firma digitale con protocollazione .....	25
Esempio 2 – Linguaggio C# - Avvio Workflow Protocollazione manuale.....	26
Esempio 3 – Linguaggio C# - Avvio Workflow Protocollazione automatica.....	27
Esempio 4 – Linguaggio C# - Avvio Workflow crea archivio .....	28
Esempio 5 – Linguaggio C# - Avvio Workflow Invia PEC automatica.....	29
Esempio 5 – Linguaggio C# - Avvio Workflow Creazione Fascicolo .....	29
Esempio 6 – Modello JSON - Evento “Protocollo creato” a fronte di un workflow “Protocolla documento” .....	30
Esempio 7 – Modello JSON - Evento “Protocollo creato” a fronte di un workflow “Collaborazione firma digitale” .....	30
Esempio 8 – Modello JSON - Evento “PEC ricevuta e associata al protocollo” a fronte di un workflow “Collaborazione firma digitale” .....	31
Esempio 9 – Modello JSON - Evento “PEC ricevuta e associata al protocollo” con esempio di indirizzo destinatario PEC specificato .....	32

## 1. Introduzione

Il seguente documento ha lo scopo di descrivere, in modalità tecnica, le interfacce e le metodologie che dovranno essere usate dai fornitori per integrarsi correttamente nella gestione esterna dei flussi documentali da e verso la DocSuite. Il modello di integrazioni deve fornire un set di Web Services che permettano l'accesso diretto al Protocollo, Atti, Archivi, Fascicoli, Dossier e Trasparenza Pubblica oltre al loro contenuto documentale in copia conforme.

Questo documento è un estratto semplificato di sola lettura architetturale, che descrivere le macro-funzioni con esempi generalisti. Per un dettaglio più specifico è necessario richiedere la versione "Manuale dello Sviluppo alle integrazioni".

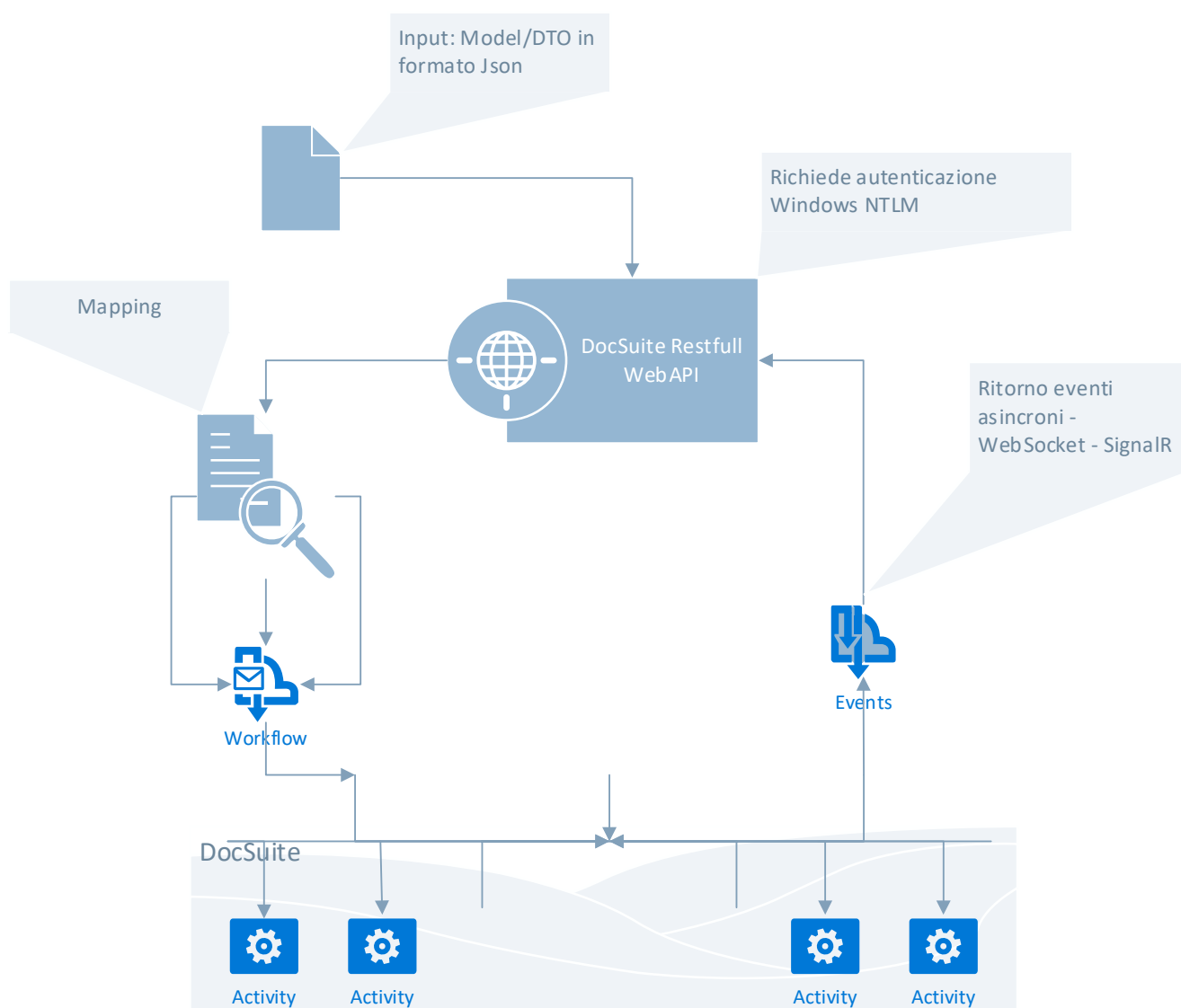
## 2. Requisiti di integrazione

I requisiti necessari a una corretta integrazione sono:

- Tutte le chiamate verso i servizi di integrazione sono autenticate tramite Windows NTLM. Nel caso di interrogazioni e/o attività con utenti anonimi vanno stabiliti i modelli di trust per il reciproco riconoscimento degli utenti.
- La tecnologia di riferimento sono le WebAPI quindi i modelli sono passati in formato JSON.
- Il prodotto DocSuite WebAPI attualmente supporta i verbi POST / GET veicolati da protocollo http/https. Gli eventuali altri verbi saranno analizzati al manifestarsi della necessità.
- Gli stream binari sono compatibili con la codifica Base64 (UTF-8)
- Le date sono richieste nel formato ISO 8601 ["yyyy-MM-dd'T'HH:mm:ss.FFFK"]
- Necessità di inserimento, modifica e lettura della Rubrica DocSuite.
- Necessità di aprire una Dossier.
- Necessità di aprire una Fascicolo.
- Necessità di inserire Unità documentali:
  - Protocolli
  - Archivi (ad esempio Fatture attive, Fatture passive, Offerte, Richieste di acquisto, Richieste di ordini, Contratti, Mancate disdette, Libro giornale, ecc...)
- Necessità di leggere specifiche Unità documentali.
- Necessità di interagire con le notifiche del motore di workflow.
- Necessità di accedere a una rappresentazione UI in sola lettura dei vari moduli della DocSuite senza entrare direttamente nella complessità funzionale e operativa. Il modulo si chiamerà "External Viewer".

### 3. Architettura dell'integrazione

Di seguito viene mostrata l'architettura di riferimento che è composta principalmente da un unico 'entry point', ovvero le WebAPI Restfull che isolano la complessità di accesso al motore della DocSuite. Nei capitoli successivi verranno fornite le definizioni di tutti i Model/DTO necessari al corretto funzionamento dei requisiti del cliente.



Il pattern di integrazione si basa quindi sia sull'utilizzo di scenari command / response via service bus sia di publish / subscribe con filtri sui topic pubblicati e sottoscritti.

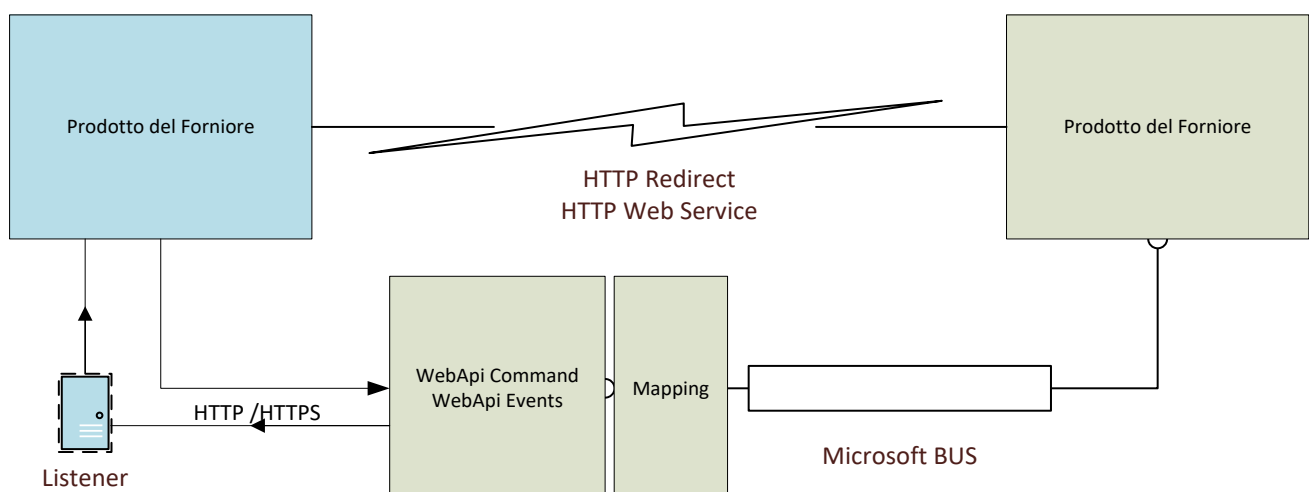
In particolare il primo caso si applica per la comunicazione di richieste (comandi) ovvero di azioni che il sistema destinatario deve intraprendere, mentre il secondo si applica alla comunicazione (pubblicazione) reciproca di eventi che accadono nei due sistemi.

La componente di riferimento si basa sul prodotto componente Microsoft Service Bus. Si propone, dunque, di realizzare uno strato di servizi WebApi pubbliche che permettano l'inserimento di comandi destinati alla DocSuite e uno che permetta la ricezione di eventi da DocSuite verso il chiamante.

I vantaggi di inserire tale insieme di servizi rispetto all'utilizzo diretto dell'infrastruttura del bus da parte del fornitore che si integra sono:

1. Rendere agnostica al fornitore la posizione di deploy del frontend DocSuite
2. Rendere indipendente la evoluzione degli schemi dei messaggi DocSuite mantenendo stabile l'interfaccia esposta
3. Permettere il posizionamento di una componente di mappatura che permetta di porre in relazione logica, il più possibile configurabili e dinamica, i metadati legati ai tag concordati in fase di configurazione e i metadati utilizzati all'interno della DocSuite per i protocolli, archivi, pratiche e fascicoli.

Proponiamo una visione dell'architettura di alto livello, che comprende anche il prodotto che si integra con i servizi della DocSuite:

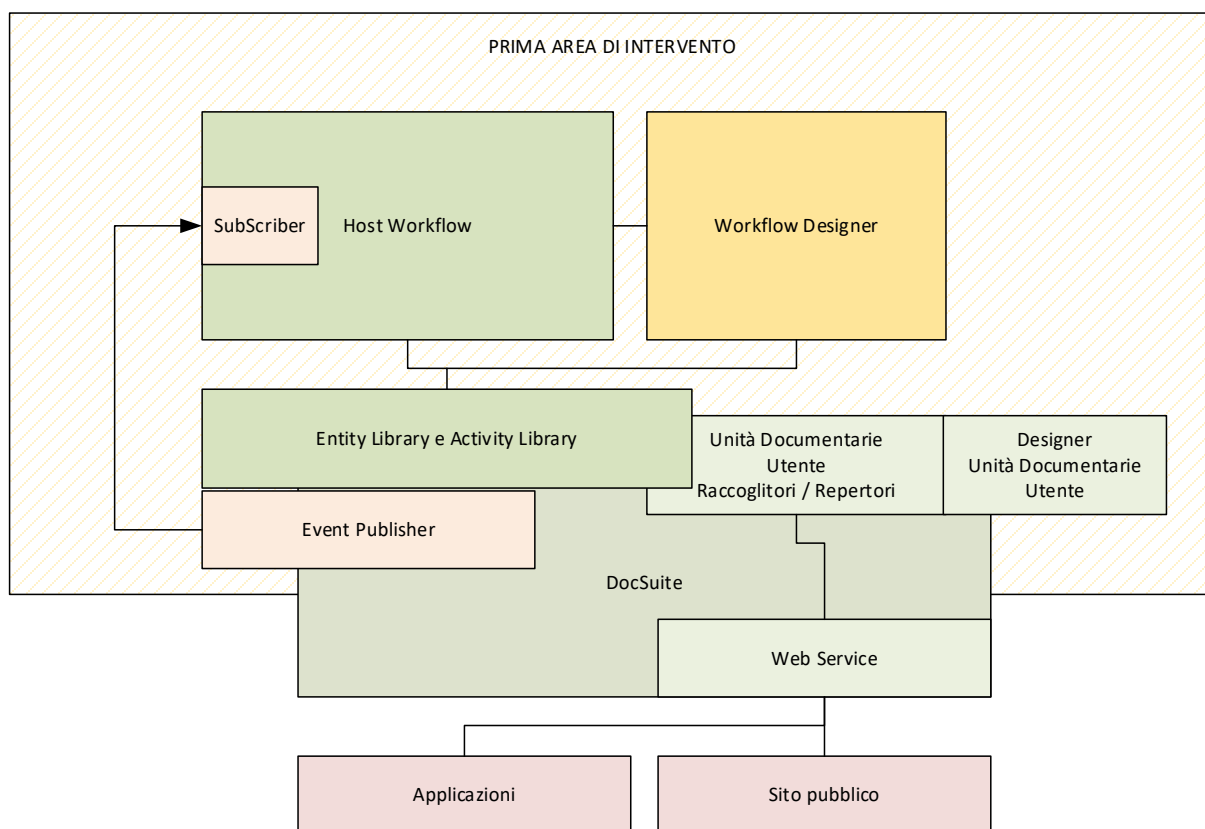


Si evidenzia il Listener lato Fornitore come componente logica o fisica che si occupa di invocare a tempo le WebApi degli eventi e gestirli all'interno della piattaforma del fornitore.

I servizi WebApi sono sottoposti a Windows Authentication e abilitati all'utente applicativo delle componenti. Eventuale filtro sull'IP di provenienza può essere applicato via configurazione del site IIS che ospita i servizi.

### Architettura complessiva di riferimento (DocSuite @BPM)

L'architettura della DocSuite complessiva della parte che abilita l'utilizzo dei workflow è la seguente:



E' compresa anche una prima versione nelle componenti che vanno a comporre il motore di BPM ovvero:

- Entity e Entity Activity Library : sarà esposta e resa disponibile sia la rappresentazione della entità delle unità documentarie specifiche (e loro istanze) e la libreria di rappresentazione ed esecuzione della attività atomiche rese disponibili al workflow.
- Una prima versione del Workflow Designer che permette di definire i primi esempi di workflow

Il motore di esecuzione dei workflow, basato su tecnologia Microsoft Windows Workflow Foundation (WF) e Microsoft Workflow Manager.

#### 4. Scenari di integrazione

##### Visualizzazione External Viewer dei documenti di un protocollo/archivio/atto

Quando dall'applicativo del fornitore è stata eseguita la protocollazione o archiviazione di alcuni documenti l'evento di avvenuta protocollazione e archiviazione contiene gli identificativi univoci per la chiamata via URL che permette all'utente di visualizzare il sommario di protocollo.

L'evento pubblicato dalla DocSuite contiene tutte le informazioni di contesto archivistico del documento, quindi ad esempio a fronte di una semplice archiviazione potrebbero essere ritornate le informazioni relative al fascicolo, protocollo, registrazione di archivio e le diverse catene documentali (principale, allegati, annessi) che ivi sono contenute.

All'interno della DocSuite è prevista la realizzazione di una pagina di consultazione, denominata External Viewer, il cui aspetto di navigazione è simile a quello dall'attuale pagina di consultazione ma non offre opportunità di navigazione all'interno della DocSuite. Ovviamente l'accesso alle informazioni e documenti permane controllato dai diritti dell'utente windows che la utilizza (windows integrated) all'interno della DocSuite o dalle estensioni pilotate dal chiamante per mezzo del token e del certificato.

Se nell'albero di sinistra si seleziona un elemento (protocollo, fascicolo, archivio, ...) a destra viene mostrato il sommario dell'elemento che come nel caso di un protocollo è un oggetto ricco che offre le informazioni di riepilogo sia della protocollazione che dei flussi di autorizzazione (distribuzione) del protocollo stesso e le informazioni relative alle comunicazioni che ci sono state (pec, email, ...)



Se l'ambito di accesso alla riservatezza documentale, qual ora il cliente ne facesse esplicita richiesta, dovesse contenere documenti sensibili rientranti nelle casistiche descritte dall'art 25 del GDPR 2018 "General Data Protection Regulation" del nuovo codice della privacy, verrà attivata una specifica autorizzazione di accesso privacy ai documenti classificati nella DocSuite. I livelli privacy dovranno essere attribuite all'utente esterno identificato nell'integrazione.

Questa modalità necessita dunque di una specifica gestione amministrativa dentro DocSuite di identificazione e classificazione del corretto livello privacy per ogni utente esterno che dovrà avere una corretta profilazione in specifiche tabelle di configurazione DocSuite.

Se si seleziona un documento, ad esempio il documento principale del protocollo, allora viene visualizzata la copia conforme del documento.

#### Descrizione tecnica per il fornitore per integrare l'External Viewer


Il meccanismo di fiducia si basa su un servizio della Public WebAPI della DocSuite invocabile via rest http attraverso il verbo POST, che genererà un token (tramite claim authentication - OAuth) di accesso temporizzato con scadenza (mediamente 50 secondi).

L'utente dovrà utilizzare l'url preformattato secondo la documentazione, entro la scadenza del token. Ogni volta che l'utente vorrà accedere all'External Viewer, dovrà rieseguire la generazione del token. Ogni richiesta di visualizzazione contenente un token valido viene accettata dalla DocSuite e inserita nel log di audit come "Visualizzazione utente da <nome sistema del fornitore es: CRM>".

Mostriamo in dettaglio come funziona l'integrazione. A fronte dell'url contenuto nell'evento DocSuite, `https://<DNS>/ExternalViewer/#/Protocollo/anno/<ANNO>/numero/<NUMERO>` per poterlo utilizzare è necessario eseguire i seguenti passaggi per effettuare l'autenticazione (ad ogni singolo accesso):

- 1) Invocare lato server il controller "api/TokenSecurities" WebAPI REST (verbo POST) `https://<dns>/DSW.PublicWebAPI/api/TokenSecurities?authenticationId=<chiave di sicurezza dell'integrazione>&documentUnit=Protocol&year=2018&number=120,` La chiamata vi permetterà di generare un token di sicurezza temporale con scadenza impostabile in fase di integrazione (generalmente 30 secondi).

Mostriamo un esempio reale di utilizzo del controller che restituisce il token di sicurezza 08b636c6-0445-4c92-acf2-924cfc1e806e



**Request Headers** [Raw] [Header Definitions]

POST /DSW.PublicWebAPI/api/TokenSecurities?authenticationId=7AF08013-7018-4583-942C-73441C680994 HTTP/1.1

**Client**  
User-Agent: Fiddler

**Entity**  
Content-Length: 0

**Transport**  
Host: localhost

Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies

Raw JSON XML

HTTP/1.1 200 OK  
Content-Type: application/json; charset=utf-8  
Server: Microsoft-IIS/10.0  
X-Powered-By: ASP.NET  
Date: Wed, 05 Sep 2018 12:06:18 GMT  
Content-Length: 38  
"08b636c6-0445-4c92-acf2-924cfc1e806e"

La chiave di sicurezza specifica viene consegnata da Dgroove ai fornitori che ne faranno richiesta, previa autorizzazione del cliente che richiede l'integrazione.

- 2) Attraverso il TOKEN di sicurezza è possibile restituire al browser dell'utente il seguente URL così formattato (in evidenza le parti specifiche che vanno modificate)

http://<DNS>/ExternalViewer/#/Auth/appId/<chiave di sicurezza dell'integrazione>/authrule/token/kind/Protocol/param/<token=<valore del token generato dal servizio REST>|user=<nome utente>|year=>anno di protocollo>|number=<numero protocollo> tutto va codificato con encoding html>/redirectUrl/<url ricevuto nel messaggio della DocSuite> va codificato con encoding html

In dettaglio il significato dei vari punti evidenziati:

- Chiave di sicurezza dell'integrazione: valore statico GUID che verrà fornita al fornitore
- Token: valore statico che determina il tipo di autenticazione OAuth basata su Token.
- Protocol: valore statico per la visualizzazione di protocollo (oggi unico utilizzabile)
- <url ricevuto con encoding HTML applicato> : e' l'url dell'externalviewer, applicato con un encoding html in modo tale che i caratteri dell'url relativo non vengano interpretati dal browser
- <token=valore del token .....>: tale valore deve essere codificato con encoding html e contiene le informazioni di sicurezza, quali :
  - Token: da leggere mediante il controller rest TokenSecurities
  - User: nome utente dell'applicativo del fornitore
  - Year: anno di protocollo
  - Number: numero del protocollo
  - essere applicato con un encoding html

Mostriamo un esempio per comprendere al meglio il meccanismo di codifica e composizione l'url in considerazione del fatto che in alcune sezioni va applicato l'encoding HTML. L'accesso è a fronte di una richiesta per il protocollo 2018/120 da parte dell'utente 'giordano.colasanti':

<https://<DNS>/ExternalViewer/#!/Auth/appld/49DC7004-EFC1-454A-9174-CEA12D06061E/authrule/token/kind/Protocol/param/token%3d08b636C6-0445-4C92-acf2-924cfc1e806e%7cuser%3dgiordano.colasanti%7cyear%3d2018%7cnumber%3d120/redirectUrl/https%3A%2F%2F<DNS>%2FExternalViewer%2F%23%2FProtocollo%2Fanno%2F2018%2Fnumero%2F120>

## 5. Documentazione WebApi pubbliche di integrazione

### Descrizione schema generico di un comando delle WebAPI

Per esigenze di sviluppo le WebAPI ha un unico controller dei comandi che gestisce in maniera 'astratta' un'unica interfaccia ICommand. In questo modo possiamo gestire le operazioni di mapping (quindi determinare le activities del workflow necessarie alla corretta esecuzione del comando) in modo astratto. Questo approccio ci permette di evitare di dover rilasciare ad ogni implementazione una versione nuova delle WebAPI, aumentando di certo anche i costi di integrazione dei vari sistemi esterni alla DocSuite. Nella nostra visione l'evolutiva è nella interpretazione di strutture dinamiche di mapping (quindi non definite all'interno del codice) e l'installazione di motori di valutazione delle singole activity del workflow.

Un comando generico è sempre composto da un insieme minimo di proprietà che elenchiamo:

- Id: "<GUID>" [Valore univoco da inserire nel comando. Serve per agganciare gli eventi al comando. Funge da correlationId in architetture Event Driven]
- MessageName: "<nome comando>" [il nome del comando è una stringa. I valori sono documentati nei capitoli successivi]
- MessageDate: [valore stringa che rappresenta un DateTimeOffset nel formato "yyyy-MM-dd'T'HH:mm:ss.FFFK"]
- TenantName: Valore stringa statico "da concordare con Dgroove"
- TenantId: Valore stringa statico "da concordare con Dgroove"
- TenantA00Id: Valore stringa statico "da concordare con Dgroove"
- ContentType: Contiene un DTO che elenca le proprietà specifiche del workflow. Ogni comando prevede uno specifico DTO di Content che verrà descritto nel documento.
- IdentityContext: Struttura che identifica l'identità utente del chiamante. Un questo campo è necessario identificare l'utente "umano" che fisicamente sta effettuando la richiesta.
- CustomProperties: è un dizionario <stringa, object> che contiene alcune informazioni essenziali per la codifica corretta del comando.
  - "messageName": <valore della proprietà messageName>
  - "messageDate": <valore della proprietà messageDate>
  - "tenantName": <valore della proprietà TenantName>
  - "tenantId": <valore della proprietà TenantId>
  - "tenantA00Id": <valore della proprietà TenantA00Id>
  - "id": <valore univoco che identifica il correlationId>
  - "identity": <nome utente con dominio o utente del sistema chiamante che sta effettuato la richiesta>. Va sempre specificato il nome utente di contesto che sta usando l'applicazione reale. Non vanno mai indicati nomi utente di servizio.
  - "contentType": <identificativo del modello, tipicamente WorkflowModel>

### ContentType StartWorkflowCommand: Descrizione del modello di avvio workflow

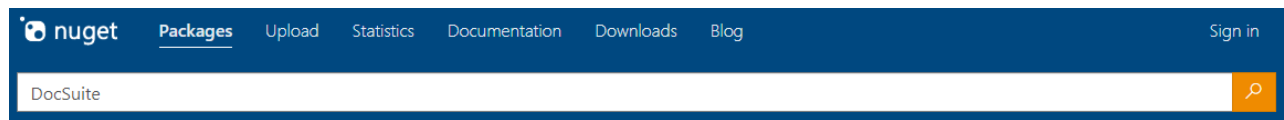
Per avviare un flusso di lavoro tramite le WebAPI, a prescindere dal tipo di Workflow, è necessario invocare un comando 'generico' definito dall'interfaccia ICommandStartWorkflow. Questa interfaccia è stata progettata per generalizzare al massimo tutti i possibili workflows definibili nella DocSuite.

Usando questo modello è necessario specificare il valore "WorkflowModel" nella proprietà "contentType".

Esponiamo con uno schema sintattico le parti più significative.







```
"contentType": {  
  "correlationId": <valore guid da usare per leggere gli eventi di completamento>,  
  "executorUser": <valore string corrisponde all'identità utente che sta effettuando la richiesta>,  
  "content": {  
    "Id": <valore guid che identifica univocamente il comando. Questo valore sarà la chiave dell'istanza  
    del workflow che fungerà anch'esso (similmente al CorrelationId) per gli eventi del Service Bus.  
    Diventerà utile per determinare quali eventi sono associati a determinate istanze attive.>.  
    "name": <valore string del nome di workflow da avviare>,  
    "workflowParameters": [<Collezione di parametri di avvio del workflow. Ogni parametro è di tipo  
WorkflowParameterModel e contiene un modello che deriva dalla interfaccia generica IModel. Le classi  
esistenti sono le seguenti:  
- ArchiveModel  
- CategoryModel  
- ContainerModel  
- ContactModel  
- DocSuiteSectorModel  
- DocumentModel  
- DocumentUnitModel  
- DossierModel  
- FascicleModel  
- MetadataModel  
- ReferenceModel  
- SignerModel  
]]  
  }  
}
```

Per semplicità di distribuzione delle componenti di integrazione, in quanto sono rivolte principalmente a sviluppatori, abbiamo pubblicato sul sito ufficiale di NUGET (<https://www.nuget.org/>) le librerie compatibili col framework Microsoft .NET [>4.6.1]>4.8] utilizzabili con lo strumento Visual Studio o compatibili.



1.388 packages returned for DocSuite

[Hide prerelease](#)

-  [VecompSoftware.DocSuite.Public.Core.Models](#) by: [Fabrizio.Lazzarotto](#)  
↓ 2.503 total downloads | ⌚ last updated an hour ago | 📦 Latest version: 9.7.0 | 🔗 VecompSoftware Models DocSuite Integrazioni Standard DocSuite...  
Modelli standard per l'integrazione, tramite WebAPI, con il sistema documentale DocSuite
-  [VecompSoftware.DocSuite.Public.Helpers](#) by: [Fabrizio.Lazzarotto](#)  
↓ 1.310 total downloads | ⌚ last updated 19/12/2016 | 📦 Latest version: 8.56.0 | 🔗 VecompSoftware Models DocSuite Integrazioni Helpers DocSuite...  
Libreria coi metodi di helpers standard per l'integrazione, tramite WebAPI, con il sistema documentale DocSuite
-  [VecompSoftware.DocSuite.Public.Core.Models.Customs](#) by: [Fabrizio.Lazzarotto](#)  
↓ 1.365 total downloads | ⌚ last updated 19/12/2016 | 📦 Latest version: 8.56.0 | 🔗 VecompSoftware Models DocSuite Integrazioni Custom DocSuite...  
Modelli custom per l'integrazione, tramite WebAPI, con il sistema documentale DocSuite
-  [VecompSoftware.DocSuite.Public.Core.Models.4.5](#) by: [Fabrizio.Lazzarotto](#)  
↓ 738 total downloads | ⌚ last updated 21/12/2016 | 📦 Latest version: 8.56.0.3 | 🔗 VecompSoftware Models DocSuite Integrazioni Standard DocSuite...  
Modelli standard per l'integrazione, tramite WebAPI, con il sistema documentale DocSuite
-  [VecompSoftware.DocSuite.Public.Core.Models.Helpers](#) by: [Fabrizio.Lazzarotto](#)  
↓ 442 total downloads | ⌚ last updated an hour ago | 📦 Latest version: 9.7.0 | 🔗 VecompSoftware Models DocSuite Integrazioni Helpers DocSuite.Pu...  
Libreria coi metodi di helpers dei modelli per l'integrazione, tramite WebAPI, con il sistema documentale DocSuite
-  [VecompSoftware.DocSuite.Public.Helpers.4.5](#) by: [Fabrizio.Lazzarotto](#)  
↓ 309 total downloads | ⌚ last updated 19/12/2016 | 📦 Latest version: 8.56.0 | 🔗 VecompSoftware Models DocSuite Integrazioni Helpers DocSuite.Pu...  
Libreria coi metodi di helpers standard per l'integrazione, tramite WebAPI, con il sistema documentale DocSuite

### WebApi Descrizione del modello – CollaborationModel

Modello dell'Unità Documentaria prodotta dalla collaborazione di firma/protocollo

- Direction: Specifica se l'unità documentaria è in ingresso o in uscita
  - Inbound = -1 (Ingresso)
  - Internal = 0 (Interno / Tra uffici)
  - Outgoing = 1 (Uscita)
- DocumentUnitType: Tipologie di unità documentarie della DocSuite
  - Protocol = 1 (Protocollo)
  - Resolution = 2 (Atto)
  - DocumentSeries = 4 (Serie documentali)
  - Archive = 8 (Archivio / Unità documentaria specifica)
- Category = CategoryModel del Classificatore/Titolario dell'unità documentaria
- Container = ContainerModel del contenitore dell'unità documentaria
- Subject = Oggetto dell'unità documentaria
- Note = Note dell'unità documentaria
- Metadatas = Collezione di MetadataModel coi metadati dell'archivio
- Contacts = Collezione di ContactModel coi contatti

### WebApi Descrizione del modello – SignerModel

Modello del firmatario tipicamente usato nei workflow di firma digitale

- SignerType: Tipologia di utente di firma
  - AD = 1 (Utente di dominio)
  - DSWRole = 2 (Settore)
  - Mapping = 4 (Codice di mapping concordato con Dgroove)
- Identity: IdentityModel dell'utente (da usare in caso di SignerType = AD)
- Role: RoleModel del settore (da usare in caso di SignerType = DSWRole)
- Order = Posizione del firmatario. Deve iniziare col valore 1
- Required = Valore true/false che identifica l'obbligatorietà della firma qualificata

### WebApi Descrizione del modello – CategoryModel

Modello del Classificatore / Titolario

- Name: Voce del classificatore
- UniqueId: Codice del classificatore della DocSuite
- MappingTag: Codice di mapping concordato con Dgroove

### WebApi Descrizione del modello – ContainerModel

Modello del Contenitore dell'unità documentaria

- Name: Nome del contenitore
- UniqueId: Codice del contenitore della DocSuite
- MappingTag: Codice di mapping concordato con Dgroove

### WebApi Descrizione del modello – ArchiveModel

Modello dell'Archivio / Unità Documentaria Specifica

- ArchiveName: Nome dell'archivio documentale
- ArchiveId: Identificativo dell'archivio documentale
- Metadatas: Collezione di MetadataModel coi metadati dell'archivio

### WebApi Descrizione del modello – DocumentUnitModel

Modello Unità Documentaria Generica

- UniqueId: Identificativo univoco dell'unità documentale
- DocumentUnitType: Tipologia di unità documentale
  - Protocol = 1 (Protocollo)
  - Resolution = 2 (Atto/Delibera/Determina)
  - DocumentSeries = 4 (Serie documentale della Trasparenza Pubblica)
  - Archive = 8 (Unità documentaria specifica)

### WebApi Descrizione del modello – MetadataModel

Modello del Metadato dinamico.

- KeyName: Nome del metadato
- Value: Valore del metadato
- MetadataId: Identificativo interno del metadato.
- LabelName: Etichetta di visualizzazione del metadato.
- ArchiveSection: Se l'unità documentaria è un archivio, è necessario specificare la sezione del metadato.



### WebApi Descrizione del modello – FascicleModel

Modello del fascicolo.

- Subject = Oggetto del fascicolo
- Note = Note del fascicolo
- Category = CategoryModel del Classificatore/Titolario del fascicolo
- Manager = ContactModel che identifica il responsabile di procedimento
- Sectors = Collezione di DocSuiteSectorModel dei settori autorizzati al fascicolo
- DocumentUnits = Collezione di DocumentUnitModel delle document unit da fascicolare

### WebApi Descrizione del modello – ContactModel

Modello Standard del Contatto Anagrafico

- ContactType: Tipologia del contatto
  - Administration = 1 (Amministrazione)
  - AOO = 2 (Area Organizzativa Omogenea (AOO))
  - AO = 4 (Unità Organizzativa (AO))
  - AO = 8 (Unità Organizzativa (AO))
  - Role = 16 (Ruolo)
  - Role = 16 (Ruolo)
  - Group = 32 (Gruppo)
  - Sector = 64 (Settore)
  - Citizen = 128 (Persona)
  - IPA = 256 (Pubblica amministrazione da IPA)
- ContactDirectionType: Tipologia mittente/destinatario del contatto
  - Sender = 1 (Mittente)
  - Recipient = 2 (Destinatario)
- ArchiveSection: Se l'unità documentaria è un archivio, è necessario specificare la sezione del contatto
- ContactId: Codice univoco del contatto anagrafico della DocSuite
- MappingTag: Codice di mapping concordato con Dgroove
- Description: Denominazione
- BirthDate: Data di nascita nel formato "yyyy-MM-dd'T'HH:mm:ss.FFFK"
- BirthPlace: Luogo di nascita
- ExternalCode: Codice esterno del contatto
- EmailAddress: Indirizzo email
- PECAddress: Indirizzo PEC

- PhoneNumber: Numero di telefono mobile
- Address: Indirizzo
- City: Città
- ZipCode: CAP
- FiscalCode: Codice Fiscale
- FiscalTaxCode: Partita Iva o Tax Code
- TelephoneNumber: Numero di telefono
- FaxNumber: Numero di FAX
- Nationality: Nazionalità (descrizione)
- Note: Campo note

#### WebApi Descrizione del modello – DocumentModel

Modello del Documento

- DocumentName: Filename del documento
- Stream: Stream dati
- ArchiveSection: Se l'unità documentaria è un archivio, è necessario specificare anche l'identificativo o Label del documento
- DocumentType: Tipologia di documento
  - Main = 1 (Documento principale)
  - Attachment = 2 (Documento allegato)
  - Annexed = 4 (Documento annesso)

#### WebApi Descrizione del modello – DocSuiteSectorModel

Modello del Settore / Ruolo configurato nella DocSuite

- SectorRoleId: Codice univoco del settore specifico DocSuite
- Name: Nome del settore specifico DocSuite
- ArchiveSection: Se l'unità documentaria è un archivio, è necessario specificare la sezione del settore
- MappingTag: Codice di mapping concordato con Dgroove

#### WebApi Descrizione del modello – IdentityContext

Classe che contiene il contesto di sicurezza (es utente o ruoli dell'utente) del comando o dell'evento.

- Identity: Modello IdentityModel per la gestione dell'identità dell'utente.
- Roles: Collezione di modelli RoleModel che contiene i ruoli dell'utente

### WebApi Descrizione del modello – RoleModel

Classe che contiene il contesto di sicurezza "gruppo" del comando o dell'evento.

- AuthorizationType: Tipologia di gruppo
  - External = 0 (Gruppo applicativo esterno)
  - NTLM = 1 (Gruppo di Windows)
  - DocSuiteSecurity = 2 (Gruppo di sicurezza DocSuite)
  - DocSuiteToken = 4 (Token OAuth DocSuite - da non usare per il RoleModel)
  - OAuth = 8 (di sistema - non usare)
  - JWT = 16 (di sistema - non usare)
  - SPID = 32 (di sistema - non usare)
- RoleUniqueId: identificativo GUID del gruppo
- ExternalTagIdentifier : Codice di mapping concordato con Dgroove

### WebApi Descrizione del modello – IdentityModel

Modello per la gestione dell'identità dell'utente. Un buon esempio di utilizzo è passare il samAccountName nei contesti di autenticazione NTLM. Nei contesti di autenticazione OAuth si dovrebbe specificare l'accountId, o l'indirizzo email. In generale è necessario identificare univocamente un utente, quindi si deve passare un valore che permette di disambiguare le omonimie.

- Account: Riferimento all'identificativo di sicurezza. Es: samAccountName
- Authorization: Modello astratto del contesto di sicurezza
  - External = 0 (Modello di autenticazione non integrato nella DocSuite)
  - NTLM = 1 (Autenticazione integrata di Windows)
  - DocSuiteSecurity = 2 (Sicurezza integrata della DocSuite (SecurityUser))
  - DocSuiteToken = 4 (Sicurezza OAuth - Owin DocSuite WebAPI Token Bearer)
  - OAuth = 8 (Sicurezza OAuth (non DocSuite ex: Google/Live))
  - JWT = 16 (Autenticazione JSON Web Token)
  - SPID = 32 (Autenticazione SPID)

### WebApi Elenco delle parole chiave usato nei modelli – WorkflowParameterNames

I parametri dei workflow utilizzano delle chiavi identificative "statiche". Questo elenco li racchiude e li descrive.

- Parametri di protocollo:
  - PROTOCOL\_MODEL= "dsw\_p\_protocol\_model" (Modello del protocollo)
  - MANAGE = "dsw\_p\_protocol\_manage"  
(Settore competente della richiesta di protocollazione)
  - REFERENCE = "dsw\_p\_protocol" (Riferimento)
  - MAIN\_DOCUMENT = "dsw\_p\_protocol\_main\_document"  
(Documento principale)
  - ATTACHMENT\_DOCUMENT = "dsw\_p\_protocol\_attachment\_document"  
(Documento allegato)
  - ANNEXED\_DOCUMENT = "dsw\_p\_protocol\_annexed\_document"  
(Documento annesso)
  - CATEGORY\_MODEL = "dsw\_p\_protocol\_category"  
(Classificatore)
  - CONTAINER\_MODEL = "dsw\_p\_protocol\_container"  
(Contenitore)
- Parametri della collaborazione:
  - COLLABORATION\_MODEL= "dsw\_p\_collaboration\_model" (Modello della collaborazione)
  - DOCUMENT\_UNIT\_MODEL= "dsw\_p\_collaboration\_manage\_model"  
(Modello dell'unità documentale in gestione alla collaborazione)
  - MAIN\_DOCUMENT = "dsw\_p\_collaboration\_main\_document"  
(Documento principale)
  - ATTACHMENT\_DOCUMENT = "dsw\_p\_collaboration\_attachment\_document"  
(Documento allegato)
  - ANNEXED\_DOCUMENT = "dsw\_p\_collaboration\_annexed\_document"  
(Documento annesso)
  - SIGNER = "dsw\_p\_collaboration\_signer"  
(Firmatario)
  - MANAGE = "dsw\_p\_collaboration\_manage"  
(Settore della segreteria di collaborazione)

- Parametri di archivio:
  - ARCHIVE\_MODEL= "dsw\_p\_archive\_model" (Modello dell'archivio)
  - MAIN\_DOCUMENT = "dsw\_p\_archive\_main\_document"  
(Documento principale)
  - ATTACHMENT\_DOCUMENT = "dsw\_p\_archive\_attachment\_document"  
(Documento allegato)
  - ANNEXED\_DOCUMENT = "dsw\_p\_archive\_annexed\_document"  
(Documento annesso)
  - CATEGORY\_MODEL = "dsw\_p\_archive\_category"  
(Classificatore)
  - CONTACT\_MODEL = "dsw\_p\_archive\_container"  
(Contenitore)
- Parametri di fascicolo:
  - FASCICLE\_MODEL= "dsw\_p\_fascicle\_model" (Modello del fascicolo)
  - ATTACHMENT\_DOCUMENT = "dsw\_p\_fascicle\_attachment\_document"  
(Inserito del fascicolo)
  - MANAGE = "dsw\_p\_fascicle\_manage"  
(Settore responsabile del fascicolo)

### Eventi di risposta delle attività di workflow

In questa sezione sono descritti i modelli degli eventi di risposta dei workflow, rappresentati in meta linguaggio C#.

Questi eventi possono anche essere comunicati dalla DocSuite mediante chiamate a specifici controller Rest WebAPI che il Fornitore potrà offrire in fase di integrazione. In questo scenario, qual ora il fornitore sia in grado di esporre un controller REST WebAPI compatibile con lo standard RESTFull che accetti il verbo POST, la DocSuite è in grado di comunicare in real-time ogni evento generato dai workflow, codificati mediante rappresentazione JSON.

Mostriamo i vari modelli con relativa documentazione delle singole proprietà:

```
public class DocSuiteEvent
{
    /// <summary>
    /// Identificativo univoco dell'evento
    /// es: 1F4346E2-EBB5-4218-821A-4601F757BA91
    /// </summary>
    public Guid UniqueId { get; set; }

    /// <summary>
    /// Identificativo univoco del workflow che ha generato l'attività
    /// es: 5D42896D-CDB1-49E2-9655-23E362C424F7
    /// NB: questo valore potrebbe non essere passato, ad esempio nella fase 0 sarà sempre vuoto
    /// </summary>
    public Guid? WorkflowReferenceId { get; set; }

    /// <summary>
    /// Modello contenente i valori da comunicare
    /// come ad esempio la PEC, il Protocollo, il Fascicolo
    /// </summary>
    public DocSuiteModel EventModel { get; set; }

    /// <summary>
    /// Modello contenente il riferimento dell'entità che ha generato l'attività
    /// Ad esempio la gestione di una protocollazione dal flusso collaborazione conterrà in EventModel
    /// i riferimento al protocollo, mentre in ReferenceModel ci saranno i dati della collaborazione.
    /// Ragionamento analogo delle PEC. EventModel conterrà i dati della PEC mentre ReferenceModel
    /// conterrà i dati del protocollo
    /// </summary>
    public DocSuiteModel ReferenceModel { get; set; }

    /// <summary>
    /// Date afferente all'attività comunicata.
    /// es: se l'DocSuiteModel è di tipologia PEC allora sarà la data di Invio
    /// se l'DocSuiteModel è di tipologia Protocollo allora sarà la data di creazione o annullamento di protocollo
    /// se l'DocSuiteModel è di tipologia Collaboration allora sarà la data di gestione o annullamento della collaboraz.
    /// </summary>
    public DateTimeOffset EventDate { get; set; }
}
```

```
public class DocSuiteModel
{
    /// <summary>
    /// Rappresentazione testuale del modello, esempio :
    /// Protocollo 04589 del 2017
    /// </summary>
    public string Title { get; set; }
    /// <summary>
    /// Anno di creazione dell'entità passata
    /// es: 2017
    /// NB: questo valore potrebbe non essere passato, ad esempio le PEC non hanno anno.
    /// </summary>
    public short? Year { get; set; }
    /// <summary>
    /// Numero di creazione dell'entità passata
    /// es: 200458917
    /// NB: questo valore potrebbe non essere passato, ad esempio le PEC non hanno anno.
    /// </summary>
    public int? Number { get; set; }
    /// <summary>
    /// Identificativo univoco dell'entità passata
    /// es: 665AEE0F-1BB4-464B-992E-8719183594B4
    /// </summary>
    public Guid UniqueId { get; set; }
    /// <summary>
    /// Identificativo univoco dell'entità passata in forma numerica (dove prevista)
    /// es: le collaborazioni che non sono identificate con anno e numero avranno EntityId.
    /// </summary>
    public int? EntityId { get; set; }
    /// <summary>
    /// Lista che conterrà proprietà specifiche dell'integrazione col fornitore, esempio :
    /// Il mittente delle ricevute delle PEC verrà identificato con la chiave "Sender"
    /// Lo stream in formato base64 dell'XML della fattura B2B/PA ricevuta, identificato con la chiave "InvoiceXML"
    /// </summary>
    public IDictionary<string, string> CustomProperties { get; set; }
    /// <summary>
    /// Tipologia che determina a quale entità DocSuite si riferisce l'istanza del modello
    /// </summary>
    public DocSuiteType ModelType { get; set; }
    /// <summary>
    /// Stato dell'entità DocSuite
    /// </summary>
    public DocSuiteStatus ModelStatus { get; set; }
}

public enum DocSuiteType : short
{
    Invalid = 0, /// Non valido
    Protocol = 1, /// Protocollo
    PEC = 2, /// PEC
    Collaboration = 3, /// Firma digitale
    Fascicle = 4, ///
    Dossier = 5, /// PEC
    Workflow = 8,
    Build = 9
}

public enum DocSuiteStatus : short
{
    Invalid = 0,
    Activated = 1,
    Canceled = 2,
    Send = 4,
    Received = 8,
    Rejected = 16,
    Completed = 32,
    Error = 64
}
```

### Tabella dei modelli degli eventi

Evento	Identificazione evento	Informazioni comunicate
Protocollo	<b>EventModel:</b> ModelType=Protocol ModelStatus=Activated	<b>EventModel:</b> "Year"=anno di protocollazione "Number"=numero di protocollazione <b>ReferenceModel:</b> "EntityID"=numero collaborazione <b>Event:</b> "EventDate"=data protocollazione
Annulla protocollo	<b>EventModel:</b> ModelType=Protocol ModelStatus=Canceled	<b>EventModel:</b> "Year"=anno di protocollazione "Number"=numero di protocollazione <b>Event:</b> "EventDate"=data annullamento protocollo
Ricezione PEC da parte del destinatario	<b>EventModel:</b> ModelType=PEC ModelStatus=Received	<b>EventModel:</b> "CustomProperties[Sender]" = mittente della PEC <b>ReferenceModel:</b> "Year"=anno di protocollazione "Number"=numero di protocollazione <b>Event:</b> "EventDate"=data ricezione PEC



## 6. Esempi dei modelli JSON di Avvio workflow ed eventi di risposta

### Esempio 1 – Linguaggio C# - Avvio Workflow Firma digitale con protocollazione

Mostriamo un esempio generico di un avvio di un workflow (preconfigurato nella DocSuite) chiamato "Firma digitale con protocollazione". L'esempio evidenzia le strutture e i modelli necessari scritto nel linguaggio C#, che un generico fornitore potrebbe riutilizzare per comprendere le strutture.

```
//Modello che specifica l'Identità dell'utente che sta eseguendo l'avvio del Workflow
IdentityModel identityModel = new IdentityModel(accountName, AuthorizationType.External);
IdentityContext identityContext = new IdentityContext(identityModel);

/// Modello che permette l'avvio del Workflow "Firma digitale con protocollazione"
WorkflowModel workflowModel = new WorkflowModel("Firma digitale con protocollazione");

///Modello della collaborazione
CollaborationModel collaborationModel = new CollaborationModel("Richiesta firma digitale", "campo note di collaborazione", DocumentUnitDirection.Outgoing, DocumentUnitType.Protocol);
WorkflowParameterModel workflowParameterModel12 = new WorkflowParameterModel(collaborationModel, WorkflowParameterNames.CollaborationNames.COLLABORATION_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel12);

///Modello della firmatario
SignerModel signerModel01 = new SignerModel(1, true, SignerType.AD, new IdentityModel("<utente firmatario nel formato dominio\\samAccount>", AuthorizationType.NTLM));
WorkflowParameterModel workflowParameterModel13 = new WorkflowParameterModel(signerModel01, WorkflowParameterNames.CollaborationNames.SIGNER);
workflowModel.WorkflowParameters.Add(workflowParameterModel13);

///Modello della firmatario
SignerModel signerModel02 = new SignerModel(2, true, SignerType.AD, new IdentityModel("<utente firmatario nel formato dominio\\samAccount>", AuthorizationType.NTLM));
WorkflowParameterModel workflowParameterModel14 = new WorkflowParameterModel(signerModel02, WorkflowParameterNames.CollaborationNames.SIGNER);
workflowModel.WorkflowParameters.Add(workflowParameterModel14);

///modello che determina la segreteria di collaborazione
DocSuiteSectorModel docSuiteSectorModel = new DocSuiteSectorModel("VALORE DA CONCORDARE", mappingTag: "VALORE DA CONCORDARE");
WorkflowParameterModel workflowParameterModel11 = new WorkflowParameterModel(docSuiteSectorModel, WorkflowParameterNames.CollaborationNames.MANAGE);
workflowModel.WorkflowParameters.Add(workflowParameterModel11);

///Documento principale della collaborazione
DocumentModel collaborationDocument = new DocumentModel("documentoprincipale.docx", new byte[] { 0x00, 0x01 }, DocumentType.Main);
WorkflowParameterModel workflowParameterModel15 = new WorkflowParameterModel(collaborationDocument, WorkflowParameterNames.CollaborationNames.MAIN_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel15);

///Modello del protocollo
CollaborationModel collaborationToProtocolModel = new CollaborationModel("Prima protocollazione di esempio", "node della collaborazione",
    DocumentUnitDirection.Outgoing, DocumentUnitType.Protocol,
    category: new CategoryModel(uniqueId: Guid.Parse("189586C9-EA83-45DF-8816-E05F058AD0CA")),
    container: new ContainerModel(uniqueId: Guid.Parse("0A0E03B7-E459-41A1-B64D-94405AEC0B6C")),
    collaborationPriority: CollaborationPriorityType.Normal);
collaborationToProtocolModel.Contacts.Add(new ContactModel(ContactType.Administration, ContactDirectionType.Recipient)
{
    Description = "Dgroove Srl",
    Address = "Via Monte Baldo 6, Villafranca di Verona",
    PECAddress = "info@pec.it"
});
collaborationToProtocolModel.Contacts.Add(new ContactModel(ContactType.AOO, ContactDirectionType.Sender, contactId: Guid.Parse("189586C9-EA83-45DF-8816-E05F058AD0CA")));
collaborationToProtocolModel.Contacts.Add(new ContactModel(ContactType.AOO, ContactDirectionType.Recipient) { Description = "specificare descrizione contatto non di rubrica" });
collaborationToProtocolModel.Sectors.Add(new DocSuiteSectorModel(string.Empty, sectorRoleId: Guid.Parse("56A8A8E9-A811-44B7-8BF6-2046E7CF8633")));
collaborationToProtocolModel.Metadata.Add(new MetadataModel("Esempio di metadata dinamico", "valore di esempio"));

WorkflowParameterModel workflowParameterModel16 = new WorkflowParameterModel(collaborationToProtocolModel, WorkflowParameterNames.CollaborationNames.DOCUMENT_UNIT_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel16);

/// Modello o busta del comando di avvio workflow. Questo contiene il modello sopra definito
StartWorkflowContentType startWorkflowContentType = new StartWorkflowContentType(workflowModel, accountName, Guid.NewGuid());
```

## Esempio 2 – Linguaggio C# - Avvio Workflow Protocollazione manuale

Mostriamo un esempio generico di un avvio di un workflow (preconfigurato nella DocSuite) chiamato “Protocollazione semplice”. L’esempio evidenzia le strutture e i modelli necessari scritto nel linguaggio C#, che un generico fornitore potrebbe riutilizzare per comprendere le strutture.

```
//Modello che specifica l'Identità dell'utente che sta eseguendo l'avvio del Workflow
IdentityModel identityModel = new IdentityModel(accountName, AuthorizationType.External);
IdentityContext identityContext = new IdentityContext(identityModel);

/// Modello che permette l'avvio del Workflow "Protocolla Semplice"
WorkflowModel workflowModel = new WorkflowModel("Protocolla Semplice");

DocSuiteSectorModel docSuiteSectorModel = new DocSuiteSectorModel("VALORE DA CONCORDARE", mappingTag: "VALORE DA CONCORDARE");
WorkflowParameterModel workflowParameterModel11 = new WorkflowParameterModel(docSuiteSectorModel, WorkflowParameterNames.ProtocolNames.MAINAGE);
workflowModel.WorkflowParameters.Add(workflowParameterModel11);

///Modello del protocollo
CollaborationModel collaborationModel = new CollaborationModel("Prima protocollazione di esempio", string.Empty,
    DocumentUnitDirection.Outgoing, DocumentUnitType.Protocol);
collaborationModel.Contacts.Add(new ContactModel(ContactType.Administration, ContactDirectionType.Recipient)
{
    Description = "Dgroove Srl",
    Address = "via Monte Baldo 6, Villafranca di Verona",
    PECAddress = "info@pec.it"
});
collaborationModel.Contacts.Add(new ContactModel(ContactType.AOO, ContactDirectionType.Sender, contactId: Guid.Parse("1B9586C9-EA83-45DF-8816-E05F058AD0CA")));
collaborationModel.Sectors.Add(new DocSuiteSectorModel(string.Empty, sectorRoleId: Guid.Parse("56A8A8E9-A811-44B7-8BF6-2046E7CF8633")));
collaborationModel.Metadata.Add(new MetadataModel("Esempio di metadato dinamico", "valore di esempio"));

WorkflowParameterModel workflowParameterModel2 = new WorkflowParameterModel(collaborationModel, WorkflowParameterNames.ProtocolNames.PROTOCOL_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel2);

///Documento principale del protocollo
DocumentModel collaborationDocument = new DocumentModel("documentoprincipale.docx", new byte[] { 0x00, 0x01 }, DocumentType.Main);
WorkflowParameterModel workflowParameterModel3 = new WorkflowParameterModel(collaborationDocument, WorkflowParameterNames.ProtocolNames.MAIN_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel3);
DocumentModel collaborationAttachment = new DocumentModel("documentoallegato.docx", new byte[] { 0x00, 0x01 }, DocumentType.Attachment);
WorkflowParameterModel workflowParameterModel4 = new WorkflowParameterModel(collaborationAttachment, WorkflowParameterNames.ProtocolNames.ATTACHMENT_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel4);

///Classificatore
CategoryModel categoryModel = new CategoryModel(uniqueId: Guid.Parse("37579C9E-8883-49AD-BF64-5BD95E64E0E1"));
WorkflowParameterModel workflowParameterModel5 = new WorkflowParameterModel(categoryModel, WorkflowParameterNames.ProtocolNames.CATEGORY_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel5);

///Contenitore
ContainerModel containerModel = new ContainerModel(uniqueId: Guid.Parse("480F483B-A6A3-43DE-BA9D-149815E518F1"));
WorkflowParameterModel workflowParameterModel6 = new WorkflowParameterModel(containerModel, WorkflowParameterNames.ProtocolNames.CONTAINER_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel6);

/// Modello o busta del comando di avvio workflow. Questo contiene il modello sopra definito
StartWorkflowContentType startWorkflowContentType = new StartWorkflowContentType(workflowModel, accountName, Guid.NewGuid());

/// Creazione del comando di start. Nell'esempio si potrebbe sostituire Guid.NewGuid() con una chiave esterna che valore per il fornitore.
/// Utile per agganciare i messaggi di fine processo.
StartWorkflowCommand startWorkflowCommand = new StartWorkflowCommand(Guid.NewGuid(), tenantName, tenantId, tenantAooId, identityContext, startWorkflowContentType);
```

### Esempio 3 – Linguaggio C# - Avvio Workflow Protocollazione automatica

Mostriamo un esempio generico di un avvio di un workflow (preconfigurato nella DocSuite) chiamato "Protocollazione automatica". L'esempio evidenzia le strutture e i modelli necessari scritto nel linguaggio C#, che un generico fornitore potrebbe riutilizzare per comprendere le strutture.

```
///Modello che specifica l'Identità dell'utente che sta eseguendo l'avvio del Workflow
IdentityModel identityModel = new IdentityModel(accountName, AuthorizationType.External);
IdentityContext identityContext = new IdentityContext(identityModel);

/// Modello che permette l'avvio del Workflow "Protocollazione automatica"
WorkflowModel workflowModel = new WorkflowModel("Protocollazione automatica");

///Modello del protocollo
CollaborationModel collaborationModel = new CollaborationModel("Prima protocollazione di esempio", string.Empty,
    DocumentUnitDirection.Outgoing, DocumentUnitType.Protocol);
collaborationModel.Contacts.Add(new ContactModel(ContactType.Administration, ContactDirectionType.Recipient)
{
    Description = "Dgroove Srl",
    Address = "via Monte Baldo 6, Villafranca di Verona",
    PECAddress = "info@pec.it"
});
collaborationModel.Contacts.Add(new ContactModel(ContactType.AOO, ContactDirectionType.Sender, contactId: Guid.Parse("1B9586C9-EA83-45DF-8816-E05F058AD0CA")));
collaborationModel.Sectors.Add(new DocSuiteSectorModel(string.Empty, sectorRoleId: Guid.Parse("56A8ABE9-A811-44B7-8BF6-2046E7CF8633")));
WorkflowParameterModel workflowParameterModel2 = new WorkflowParameterModel(collaborationModel, WorkflowParameterNames.ProtocolNames.PROTOCOL_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel2);

///Documento principale del protocollo
DocumentModel collaborationDocument = new DocumentModel("documentoprincipale.docx", new byte[] { 0x00, 0x01 }, DocumentType.Main);
WorkflowParameterModel workflowParameterModel3 = new WorkflowParameterModel(collaborationDocument, WorkflowParameterNames.ProtocolNames.MAIN_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel3);
DocumentModel collaborationAttachment = new DocumentModel("documentoallegato.docx", new byte[] { 0x00, 0x01 }, DocumentType.Attachment);
WorkflowParameterModel workflowParameterModel4 = new WorkflowParameterModel(collaborationAttachment, WorkflowParameterNames.ProtocolNames.ATTACHMENT_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel4);

///Classificatore
CategoryModel categoryModel = new CategoryModel(uniqueId: Guid.Parse("37579C9E-8883-49AD-BF64-5BD95E64E0E1"));
WorkflowParameterModel workflowParameterModel5 = new WorkflowParameterModel(categoryModel, WorkflowParameterNames.ProtocolNames.CATEGORY_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel5);

///Contenitore
ContainerModel containerModel = new ContainerModel(uniqueId: Guid.Parse("4B0F483B-A6A3-43DE-BA9D-149815E518F1"));
WorkflowParameterModel workflowParameterModel6 = new WorkflowParameterModel(containerModel, WorkflowParameterNames.ProtocolNames.CONTAINER_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel6);

/// Modello o busta del comando di avvio workflow. Questo contiene il modello sopra definito
StartWorkflowContentType startWorkflowContentType = new StartWorkflowContentType(workflowModel, accountName, Guid.NewGuid());

/// Creazione del comando di start. Nell'esempio si potrebbe sostituire Guid.NewGuid() con una chiave esterna che valore per il fornitore.
/// Utile per agganciare i messaggi di fine processo.
StartWorkflowCommand startWorkflowCommand = new StartWorkflowCommand(Guid.NewGuid(), tenantName, tenantId, tenantAooId, identityContext, startWorkflowContentType);
```

## Esempio 4 – Linguaggio C# - Avvio Workflow crea archivio

Mostriamo un esempio generico di un avvio di un workflow (preconfigurato nella DocSuite) chiamato "Crea archivio". L'esempio evidenzia le strutture e i modelli necessari scritto nel linguaggio C#, che un generico fornitore potrebbe riutilizzare per comprendere le strutture.

```
//Modello che specifica l'Identità dell'utente che sta eseguendo l'avvio del Workflow
IdentityModel identityModel = new IdentityModel(accountName, AuthorizationType.External);
IdentityContext identityContext = new IdentityContext(identityModel);

/// Modello che permette l'avvio del Workflow "Crea contratto"
WorkflowModel workflowModel = new WorkflowModel("Crea contratto");

///Modello dell'archivio - il nome "Contratti" è un esempio ma in produzione va concordato
ArchiveModel archiveModel = new ArchiveModel("Contratti", archiveId: Guid.Parse("5A9586C9-EA83-45DF-8816-E05F08AD081"));
/// L'elenco dei metadati specifici dell'archivio sono da concordare
archiveModel.Metadata.Add(new MetadataModel("NUMERO RDA", "12345"));
archiveModel.Metadata.Add(new MetadataModel("NUMERO ODA", "K123"));
archiveModel.Metadata.Add(new MetadataModel("CIG", "Z123456"));
WorkflowParameterModel workflowParameterModel2 = new WorkflowParameterModel(archiveModel, WorkflowParameterNames.ArchiveNames.ARCHIVE_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel2);

///Documento principale del protocollo
DocumentModel collaborationDocument = new DocumentModel("documentoprincipale.docx", new byte[] { 0x00, 0x01 }, DocumentType.Main);
WorkflowParameterModel workflowParameterModel3 = new WorkflowParameterModel(collaborationDocument, WorkflowParameterNames.ArchiveNames.MAIN_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel3);
DocumentModel collaborationAttachment = new DocumentModel("documentoallegato.docx", new byte[] { 0x00, 0x01 }, DocumentType.Attachment);
WorkflowParameterModel workflowParameterModel4 = new WorkflowParameterModel(collaborationAttachment, WorkflowParameterNames.ArchiveNames.ATTACHMENT_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel4);

///Classificatore
CategoryModel categoryModel = new CategoryModel(uniqueId: Guid.Parse("37579C9E-8883-49AD-BF64-5BD95E64E0E1"));
WorkflowParameterModel workflowParameterModel5 = new WorkflowParameterModel(categoryModel, WorkflowParameterNames.ArchiveNames.CATEGORY_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel5);

///Contatto - archivesection è obbligatoria e da concordare
ContactModel contactModel = new ContactModel(ContactType.Administration, ContactDirectionType.Recipient, archiveSection: "Fornitore")
{
    Description = "Dgroove Srl",
    Address = "via Monte Baldo 6, Villafranca di Verona",
    PECAddress = "info@pec.it"
};
WorkflowParameterModel workflowParameterModel6 = new WorkflowParameterModel(contactModel, WorkflowParameterNames.ArchiveNames.CONTACT);
workflowModel.WorkflowParameters.Add(workflowParameterModel6);

/// Modello o busta del comando di avvio workflow. Questo contiene il modello sopra definito
StartWorkflowContentType startWorkflowContentType = new StartWorkflowContentType(workflowModel, accountName, Guid.NewGuid());

/// Creazione del comando di start. Nell'esempio si potrebbe sostituire Guid.NewGuid() con una chiave esterna che valore per il fornitore.
/// Utile per agganciare i messaggi di fine processo.
StartWorkflowCommand startWorkflowCommand = new StartWorkflowCommand(Guid.NewGuid(), tenantName, tenantId, tenantAooId, identityContext, startWorkflowContentType);
```

### Esempio 5 – Linguaggio C# - Avvio Workflow Invia PEC automatica

Mostriamo un esempio generico di un avvio di un workflow (preconfigurato nella DocSuite) chiamato “Invia PEC automatica”. L’esempio evidenzia le strutture e i modelli necessari scritto nel linguaggio C#, che un generico fornitore potrebbe riutilizzare per comprendere le strutture.

```
//Modello che specifica l'Identità dell'utente che sta eseguendo l'avvio del Workflow
IdentityModel identityModel = new IdentityModel(accountName, AuthorizationType.External);
IdentityContext identityContext = new IdentityContext(identityModel);

/// Modello che permette l'avvio del Workflow "Invia PEC"
WorkflowModel workflowModel = new WorkflowModel("Invia PEC automatica");

///Modello del protocollo
PECMailModel pecmailModel = new PECMailModel("oggetto della pec", "messaggio della PEC", "mittente@dominio.it", "destinatario@dominio.it", "destinatariocc@dominio.it");
WorkflowParameterModel workflowParameterModel2 = new WorkflowParameterModel(pecmailModel, WorkflowParameterNames.PECMAIL_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel2);

///Documento principale del protocollo
DocumentModel collaborationDocument = new DocumentModel("documentoprincipale.docx", new byte[] { 0x00, 0x01 }, DocumentType.Main);
WorkflowParameterModel workflowParameterModel3 = new WorkflowParameterModel(collaborationDocument, WorkflowParameterNames.PECMAIL_MODEL.MAIN_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel3);

/// Modello o busta del comando di avvio workflow. Questo contiene il modello sopra definito
StartWorkflowContentType startWorkflowContentType = new StartWorkflowContentType(workflowModel, accountName, Guid.NewGuid());

/// Creazione del comando di start. Nell'esempio si potrebbe sostituire Guid.NewGuid() con una chiave esterna che valore per il fornitore.
/// Utile per agganciare i messaggi di fine processo.
StartWorkflowCommand startWorkflowCommand = new StartWorkflowCommand(Guid.NewGuid(), tenantName, tenantId, tenantAooId, identityContext, startWorkflowContentType);
```

### Esempio 5 – Linguaggio C# - Avvio Workflow Creazione Fascicolo

Mostriamo un esempio generico di un avvio di un workflow (preconfigurato nella DocSuite) chiamato “Crea Fascicolo”. L’esempio evidenzia le strutture e i modelli necessari scritto nel linguaggio C#, che un generico fornitore potrebbe riutilizzare per comprendere le strutture.

```
//Modello che specifica l'Identità dell'utente che sta eseguendo l'avvio del Workflow
IdentityModel identityModel = new IdentityModel(accountName, AuthorizationType.External);
IdentityContext identityContext = new IdentityContext(identityModel);

/// Modello che permette l'avvio del Workflow "Crea Fascicolo"
WorkflowModel workflowModel = new WorkflowModel("Crea Fascicolo");

///Modello dell'fascicolo
FascicleModel fascicleModel = new FascicleModel("Oggetto del fascicolo",
//Classificatore del fascicolo
new CategoryModel(uniqueId: Guid.Parse("82FACA3C-08C5-439D-940D-A2B11394D6FE")),
//Contatto responsabile di procedimento
new ContactModel(ContactType.Citizen, contactId: Guid.Parse("B7A73278-3635-4229-A2D9-3028482748BE")));
//Elenco delle document units (protocolli)
fascicleModel.DocumentUnits.Add(new DocumentUnitModel(Guid.Parse("8AEFF4F8-0891-4816-8BC1-4F562404CD89"), 0, 0, string.Empty,
string.Empty, null, DocumentUnitDirection.Inbound, DocumentUnitType.Protocol, string.Empty));
fascicleModel.DocumentUnits.Add(new DocumentUnitModel(Guid.Parse("FEB130F8-08C7-4F9B-A29F-DF59F7EE8353"), 0, 0, string.Empty,
string.Empty, null, DocumentUnitDirection.Inbound, DocumentUnitType.Protocol, string.Empty));
//Elenco dei settori autorizzati al fascicolo
fascicleModel.Sectors.Add(new DocSuiteSectorModel("NOME DEL SETTORE AUTORIZZATO", sectorRoleId: Guid.Parse("05A03038-7623-47EA-9899-158EB70861F1")));
WorkflowParameterModel workflowParameterModel = new WorkflowParameterModel(fascicleModel, WorkflowParameterNames.FASCICLE_MODEL);
workflowModel.WorkflowParameters.Add(workflowParameterModel);

///Settore responsabile in competenza del fascicolo
DocSuiteSectorModel docSuiteSectorModel = new DocSuiteSectorModel("VALORE DA CONCORDARE", mappingTag: "VALORE DA CONCORDARE");
WorkflowParameterModel workflowParameterModel1 = new WorkflowParameterModel(docSuiteSectorModel, WorkflowParameterNames.FASCICLE_MODEL.MANAGE);
workflowModel.WorkflowParameters.Add(workflowParameterModel1);

///Documento principale del protocollo
DocumentModel collaborationDocument = new DocumentModel("documentoprincipale.docx", new byte[] { 0x00, 0x01 }, DocumentType.Main);
WorkflowParameterModel workflowParameterModel2 = new WorkflowParameterModel(collaborationDocument, WorkflowParameterNames.FASCICLE_MODEL.MAIN_DOCUMENT);
workflowModel.WorkflowParameters.Add(workflowParameterModel2);

/// Modello o busta del comando di avvio workflow. Questo contiene il modello sopra definito
StartWorkflowContentType startWorkflowContentType = new StartWorkflowContentType(workflowModel, accountName, Guid.NewGuid());
```

### Esempio 6 – Modello JSON - Evento "Protocollo creato" a fronte di un workflow "Protocollo documento".

```
{
  "UniqueId": "d0064306-96b5-404f-af04-5e0d1510f6c0",
  "WorkflowReferenceId": "1e7190f8-5a8c-4e1e-90f7-5eb759ce9c3a",
  "EventModel": {
    "Title": "Protocollo 2017/004589",
    "Year": 2017,
    "Number": 4589,
    "UniqueId": "51fe20c4-69ee-4c7d-8b36-a25cefd5d275",
    "EntityId": null,
    "ModelType": 1,
    "ModelStatus": 1
  },
  "EventDate": "2017-12-11T21:56:05.0898462+00:00"
}
```

### Esempio 7 – Modello JSON - Evento "Protocollo creato" a fronte di un workflow "Collaborazione firma digitale".

```
{
  "UniqueId": "d0064306-96b5-404f-af04-5e0d1510f6c0",
  "WorkflowReferenceId": "1e7190f8-5a8c-4e1e-90f7-5eb759ce9c3a",
  "EventModel": {
    "Title": "Protocollo 2017/004589",
    "Year": 2017,
    "Number": 4589,
    "UniqueId": "51fe20c4-69ee-4c7d-8b36-a25cefd5d275",
    "EntityId": null,
    "ModelType": 1,
    "ModelStatus": 1
  },
  "ReferenceModel": {
    "Title": "Collaborazione n° 157",
    "Year": null,
    "Number": null,
    "UniqueId": "6e80266f-ce5e-4cfc-a101-2cd2137f0b78",
    "EntityId": 157,
    "ModelType": 3,
    "ModelStatus": 1
  },
  "EventDate": "2017-12-11T21:56:05.0898462+00:00"
}
```



### Esempio 8 – Modello JSON - Evento “PEC ricevuta e associata al protocollo” a fronte di un workflow “Collaborazione firma digitale”.

```
{
  "UniqueId": "50e249b1-3910-42e8-88c2-a3bf17048b8a",
  "WorkflowReferenceId": "d4fce45e-ba38-462a-ae0-f77ca79757d1",
  "EventModel": {
    "Title": "PEC 1597",
    "Year": null,
    "Number": null,
    "UniqueId": "fcd723f9-ce5b-4e21-93ca-7cfe85524a72",
    "EntityId": 1597,
    "ModelType": 2,
    "ModelStatus": 4
  },
  "ReferenceModel": {
    "Title": "Protocollo 2017/004589",
    "Year": 2017,
    "Number": 4589,
    "UniqueId": "58958516-3d23-4f3a-a413-0bbd8d3361c7",
    "EntityId": null,
    "ModelType": 1,
    "ModelStatus": 1
  },
  "EventDate": "2017-12-11T21:56:05.0898462+00:00"
}
```

### Esempio 9 – Modello JSON - Evento “PEC ricevuta e associata al protocollo” con esempio di indirizzo destinatario PEC specificato

```
{
  "UniqueId": "920a0bf4-7b28-4717-af2d-336681e1d841",
  "WorkflowReferenceId": null,
  "EventModel": {
    "Title": "PEC 1597",
    "Year": null,
    "Number": null,
    "UniqueId": "d89ee72d-fb38-4129-ac86-26d75b545d5c",
    "EntityId": 1597,
    "CustomProperties": { "Receiver": "info@dgroove.it" },
    "ModelType": 2,
    "ModelStatus": 8
  },
  "ReferenceModel": {
    "Title": "Protocollo 2019/0000123",
    "Year": 2019,
    "Number": 123,
    "UniqueId": "5f8c188b-ad97-4440-b2ed-f5a91b8a65a3",
    "EntityId": null,
    "CustomProperties": null,
    "ModelType": 1,
    "ModelStatus": 1
  },
  "EventDate": "2019-06-12T09:43:54.4807717+00:00"
}
```