Protocolli e interfacce di comunicazione DATEX II

per il comparto autostradale italiano

Controllo Documento

Oggetto	Documento tecnico di riferimento per l'implementazione di protocolli e interfacce di comunicazione DATEX II per il comparto autostradale italiano
Versione	1.0

REDATTO:	Autostrade Tech S.p.A.	Fabrizio Paoletti Paolo Orsini
VERIFICATO:	Autostrade Tech S.p.A.	Fabrizio Paoletti
APPROVATO:	gruppo tecnico DATEX ITA	
DATA DI APPROVAZIONE:		

Storia delle modifiche:

Data	Autore	Versione	Operazione	Stato
2017-04-20	Paolo Orsini	0.1	Creazione	Draft
2017-05-10	Fabrizio Paoletti	0.1	Revisione	Draft
2017-05-26	Fabrizio Paoletti Paolo Orsini	0.1	Integrazione e Revisione	Proposta

SOMMARIO

SIGLE, ACRONIMI, DEFINIZIONI

Sigla Internazionale	Denominazione EN		Sigla IT
Alert C		Codifica di Eventi e localizzazioni per canale RDS-TMC	
Client		Nodo DATEX che riceve informazioni da altri nodi DATEX fornitori detti Supplier	
DATEX	"**DAT**a **EX**change"	Protocollo di scambio dati di informazioni sul traffico in uso fra centri di controllo e gestione traffico, centri informativi sul traffico e Operatori di Servizi informativi sul traffico	
DATEX1		Sistema di scambio dati su traffico basato su FTP e sintassi EDIFACT, implementato da un gruppo di lavoro operante in ERTICO sotto l'egida della commissione Europea negli anni '90 e implementato nei paesi europei (FR, DE, NL, UK, BE, ES, IT, ecc) nel corso degli anni. Rappresenta fino al 2006 la quasi totalità dei sistemi di scambio dati sul traffico. Alla base del sistema DATEX1 il Data Dictionary che codifica le situazioni di traffico e le definisce per l'armonizzazione dell'uso dei termini in Europa. E' stato proposto come pre norma Europea nel 2000 ma successivamente non è stato confermato per il successivo lancio del progetto DATEX2	
	D2 o DATEX 2 Project	Progetto della commissione Europea tramite DG TREN per l'evoluzione del protocollo DATEX nell'ottica di un sistema attualizzato con tecnologia Moderna, UML, XML, webservices. Il progetto è stato	**DATEX II** DATEX2
DATEX II		Versione attuale del protocollo per lo scambio dati sul traffico originato dal progetto delle commissione Europea cosiddetto D2 Project . Il protocollo si basa su XML / http o webservices. La versione 1.0 del protocollo DATEX II risale a dicembre 2006 ed è stato recepito nel 2011 come Technical Specification CEN TS 16157 per le parti 1-2-3 sulla base della versione 2.0 emanata a Luglio 2011. E' in programma l'iter di recepimento delle rimanenti parti	**DATEX II** DATEX2

		del protocollo da parte della commissione CEN TC278 per opera nel sottogruppo WG8 e con il contributo dello Studio Europeo ESG5 nell'ambito del programma finanziato Easyway (I e II) 2009-2013. Per riferimento www.datex2.eu	
Alert C Location **RDS TMC Location**	**RDS TMC Location Table**	Il database che codifica le località per l'uso nel protocollo RDS-TMC, utilizzato in modo esclusivo come metodo di localizzazione nel DATEX1 e consentito anche nel DATEX2 dove sono previsti anche dei metodi di localizzazione alternativa. In Italia il DB è mantenuto dal CCISS per conto del Ministero dei Trasporti, e operativamente dal 2003 da Autostrade per l'Italia per le necessità del comparto autostradale.	**Database Località RDS TMC**
Mare Nostrum		Progetto del programma Easyway che ha l'obiettivo di ricercare l'armonizzazione dell'uso dei PMV in Europa. Il gruppo Italiano Mare Nostrum definisce le regole di gestione di PMV in ambito nazionale per il comparto autostradale esteso alla rete Anas.	
RDS	Radio Data System	Sistema trasmissivo dati per canale Radio FM	
SP	Service Provider	Operatore che eroga servizi di informazioni sul Traffico come la diffusione via radio, tv, web, o tramite canali automatici RDS-TMC	
Supplier		Nodo DATEX che fornisce informazioni ai vari nodi DATEX ricevitori detti Client	
TMC	Traffic Message Channel	Canale Traffico del sistema RDS standardizzato per la diffusione di notizie sul traffico	
VMS	Variable Message Signs	Pannelli a Messaggio Variabile	PMV

INTRODUZIONE

Il presente documento presenta le possibili interfacce e i protocolli Datex II, che in ambito italiano gli attori di una comunicazione Datex II possono implementare. Per ogni tipo di comunicazione vengono elencati le interfacce standard Datex II utilizzate e la descrizione degli elementi di exchange Datex II standard utilizzati per sopperire alla mancanza di specifiche europee in merito.

MODELLO DATI DATEX II

Nella comunicazione Datex II del comparto Italiano il modello dati Datex II utilizzato è quello della versione 2.3 con profilo ITALIA. Si è trattato in sintesi di creare un "profilo DATEX II" come base di Linee Guida DATEX II per il comparto Italiano. Il documento che descrive tale modello è DII Profile IT discusso e approvato nelle varie riunioni del gruppo Datex II Italia.

EXCHANGE E MODI OPERATIVI

Le specifiche di Echange nel modello Datex II 2.3 sono carenti di vari aspetti ad esempio la definizione di funzionalità minime di sincronizzazione e la gestione di errori. Quindi in ogni proposta di interfaccia e protocollo vengono descritte le estensioni effettuate e l'utilizzo di attributi standard di exchange, anche se il loro utilizzo non è completamente compliant con le specifiche.

Per le specifiche di scambio con http puro e XML dalla versione 1.0 è definito la modalità cosiddetta Low Cost Profile (Simple HTTP Pull) che rende disponibili le informazioni globali di un nodo mediante la pubblicazione sul web di file statici.

Per lo scambio tramite Web Service, cosiddetta modalità Push, ci sono nelle specifiche alcune carenze che non consentono un uso di questa modalità di invio senza il bisogno di integrazioni, ad esempio la definizione di funzionalità minime di sincronizzazione e la gestione di errori.

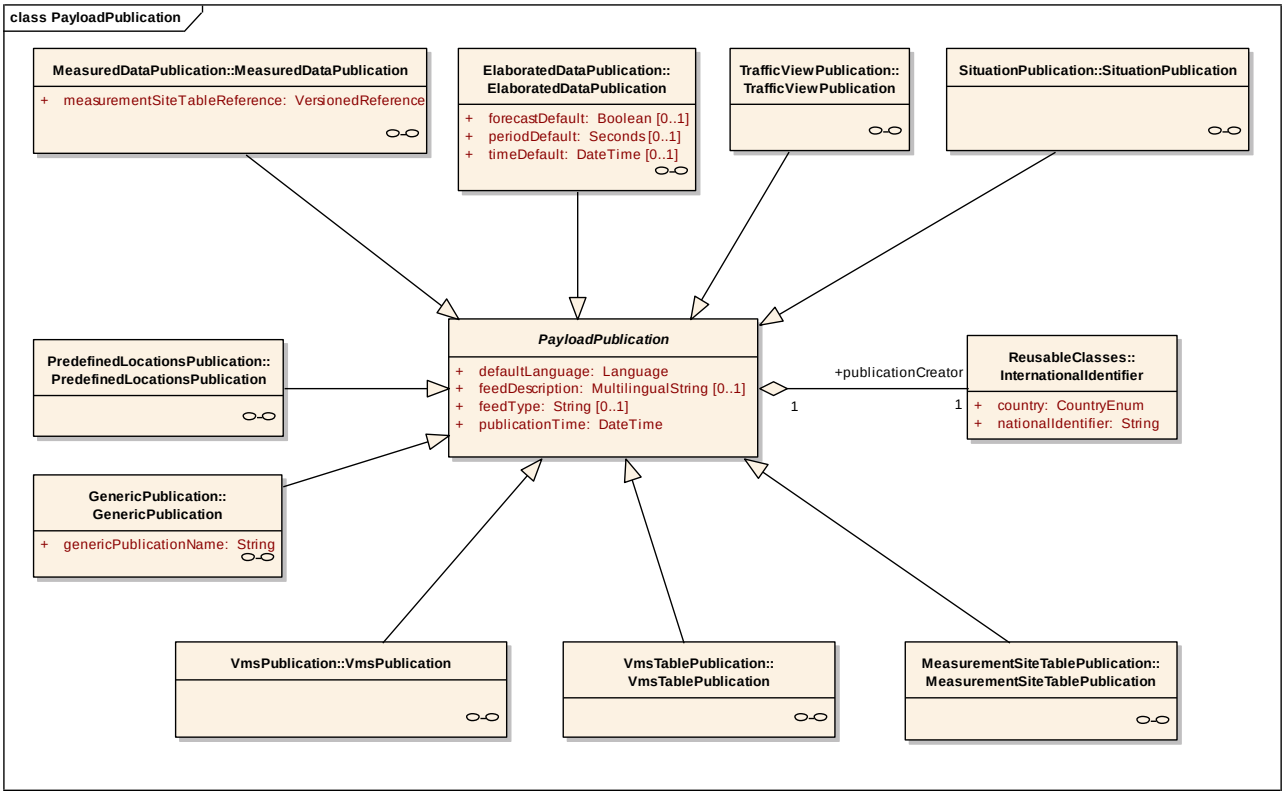
Inoltre dall' analisi svolta dal gruppo di lavoro sulle emergenze si è evidenziata la necessità di rendere disponibile al Supplier informazioni sulla corretta acquisizione ed eventuale trattamento delle informazioni lato Client. Per queste necessità sono state fatte delle ipotesi minimali di lavoro mediante paradigma Client Pull / Low Cost Profile e invio di feedback con estensione del modello DATEX2 originario.

E' in corso nello studio europeo DATEX2 il completamento del profilo di scambio in tempo reale.

GENERALITÀ DELLE PUBBLICAZIONI

Il Modello DATEX II di versione 2.0 ha come radice una Classe D2LogicalModel che riferisce 2 sotto-parti, una parte cosiddetta Exchange contenente informazioni relative ai dati necessari allo scambio dati stesso e una parte di contenuto di informazione stradale detta PayloadPublication.

I diversi contenuti di informazione stradale sono codificati nelle pubblicazioni DATEX originate come specializzazione dalla classe PayloadPublication:



In particolare sono analizzate nel nostro lavoro le seguenti

Informazione	**Pubblicazione**
Eventi totale	Situation
Anagrafica PMV	VmsTable
Stato e Messaggi PMV	Vms
Anagrafica Sensori e Centraline	MeasurementSites
Dati e Stato dei Sensori	MeasuredData
Tempi di Percorrenza, dati elaborati	ElaboratedData
Estensioni	Generic

SCAMBIO DATI

Lo scambio dati Datex II definito nei documenti ufficiali rilasciati dal gruppo DATEX II Europeo sul sito www.datex2.eu prevede che ci sia una parte Supplier che esporta l'informazione e una parte client che riceve l'informazione.

Le modalità operative con cui avviene questo scambio, rese disponibili dalla release Datex II v.2.0 (http://www.datex2.eu/content/datex-ii-exchange-psm-20) e mantenuti nelle successive release fino all'attuale, possono essere scelte fra le seguenti:

1. **http Low Cost Profile (LCP) :** export da parte del supplier di file static con contenuto Datex II standard con in più header soap (header richiesto dallo standard, all'interno della user guide Datex II, per compatibilità con webservice soap). Il client interroga l'url esportato dal supplier con chiamata http standard
2. **Web service Pull**: Invocazione da parte del client del metodo getDatex2Data esportato nel Pull.wsdl standard reso disponibile dalla controparte supplier.
3. **Web service Push**: Il supplier invia le informazioni invocando il metodo putDatex2Data esportato nel Push.wsdl standard reso disponibile dalla controparte client.

Mentre il LCP e Pull sono esaustivi delegando tutta una serie di controlli sull'aggiornamento delle informazioni lato Client, i meccanismi di delivery Push presentano delle lacune nelle specifiche a riguardo della gestione dello stato della connessione col Supplier che deve prevedere alcuni aspetti come la lettura di dati di situazione attuale alla prima connessione (Sincronizzazione) o in caso di perdita di dati per problemi di rete o indisponibilità dei sistemi per manutenzioni. In questi casi le best practice europee hanno evidenziato soluzioni che integrano le modalità precedenti per sopperire alle carenze.

I requisiti principali per uno scambio dati affidabile e coerente sono:

1. **Client e Supplier hanno conoscenza dello stato della connessione e monitorano eventuali disconnessioni o errori di trasmissione / ricezione.**
2. **Il supplier nella continuità di una sessione di scambio dati, deve essere in grado di inviare al client le informazioni aggiornate possedute internamente, gestire eventuali condizioni di errore nella trasmissione ed eventuali disconnessioni temporanee riallineando il Client alla riconnessione.**
3. **Il Client deve essere in grado di richiede al Supplier un riallineamento per sue necessità di gestione: ad esempio all' Inizio della Comunicazione, per Recupero dati dopo Disconnessione o Errore di trasmissione / sospensione del servizio per manutenzione sistema / work-around per bug verificati.**

Con questi requisiti nel 2014 è stato definito un supplemento di specifica da utilizzare a livello italiano per lo scambio dati tempestivo (Pull) ed affidabile (gestione sessioni e sincronizzazione) che è stato documentato nel profilo italiano condiviso dalle concessionarie e pubblicato sul sito DATEX.

In parallelo altri ambiti di ricerca e sperimentazione hanno evidenziato la possibilità di altri tipi di protocollo utilizzabili in ambito Datex II con tecnologie diverse e nuovi approcci, sensibilmente diversi dalla logica dei paradigmi DATEX ufficiali:

1. Full Push con Sincronizzazione: webservice Push DATEX che integra lato Supplier una logica di memorizzazione messaggi non trasmessi e successiva ritrasmissione alla riconnessione.
2. WebService RESTFull: export da parte del supplier dei servizi che consentono lo scambio dati Datex II, orientato alle risorse.
3. WebService Soap Ocit-C: webservice soap esposto da parte del supplier che prevede dei wsdl con metodi differenti dal wsdl standard (protocollo utilizzato in Germania e Austria).

PROTOCOLLI E INTERFACCE

In base a questi requisiti principali analizziamo le seguenti interfacce e i seguenti protocolli con i suoi pro e i suoi contro da poter essere implementati in ambito del comparto italiano:

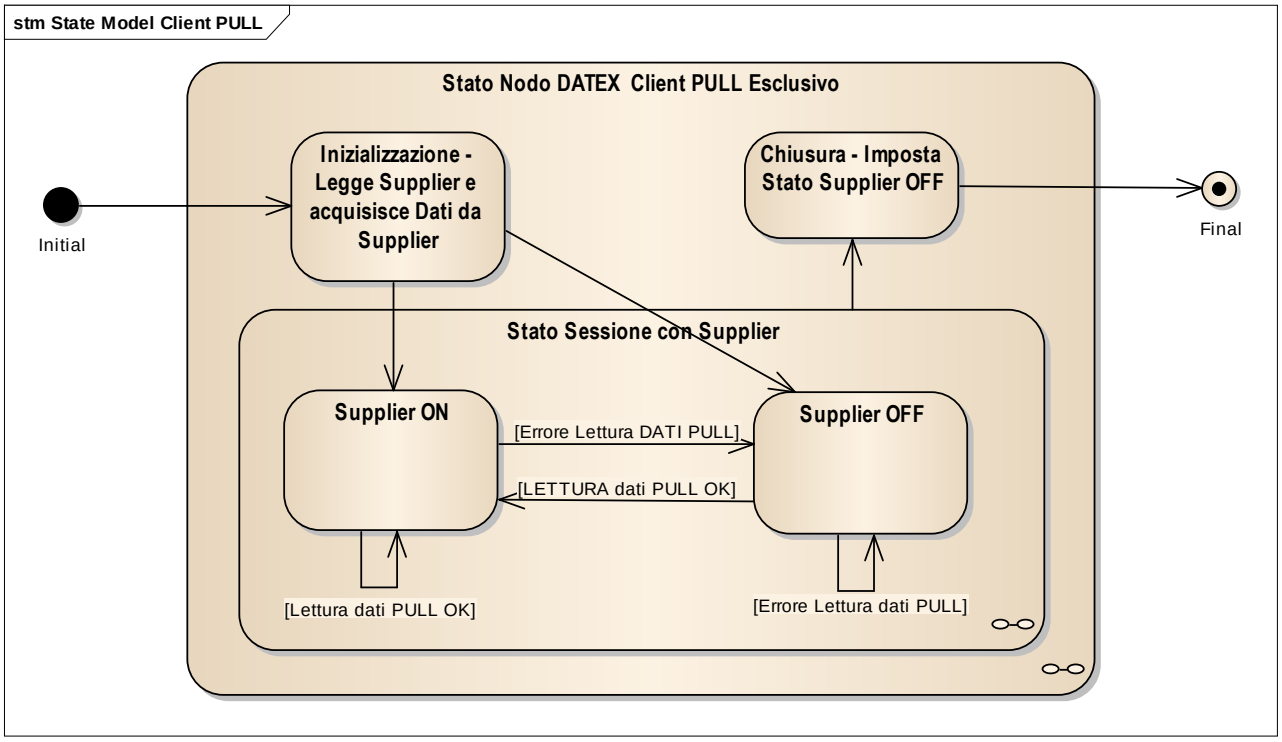
1. Low Cost Profile
2. Pull
    a. Pull Snapshot semplice (analogo a Low Cost Profile)
    b. Delta Pull (con estensione, introdotto nel Profilo Italiano 2014)
3. Push
    a. Con Riallineamento Pull (introdotto nel profilo Italiano 2014)
    b. Con Riallineamento Push (introdotto nel profilo italiano 2017)

I diversi paradigmi di scambio (Exchange Pattern) sono illustrati nel dettaglio nei successivi Paragrafi.

Il Low Cost profile prevede l'interazione fra Supplier e Client attraverso la creazione di file XML statici accessibili mediante protocollo http semplice a cui il Client accede con una logica definita dal Client stesso (periodicamente o su eventi definiti ed innescati lato client).

Accedendo ai dati via http, il Client è in grado di stabilire se il Supplier è attivo o non attivo/irraggiungibile per errori di risposta su protocollo http

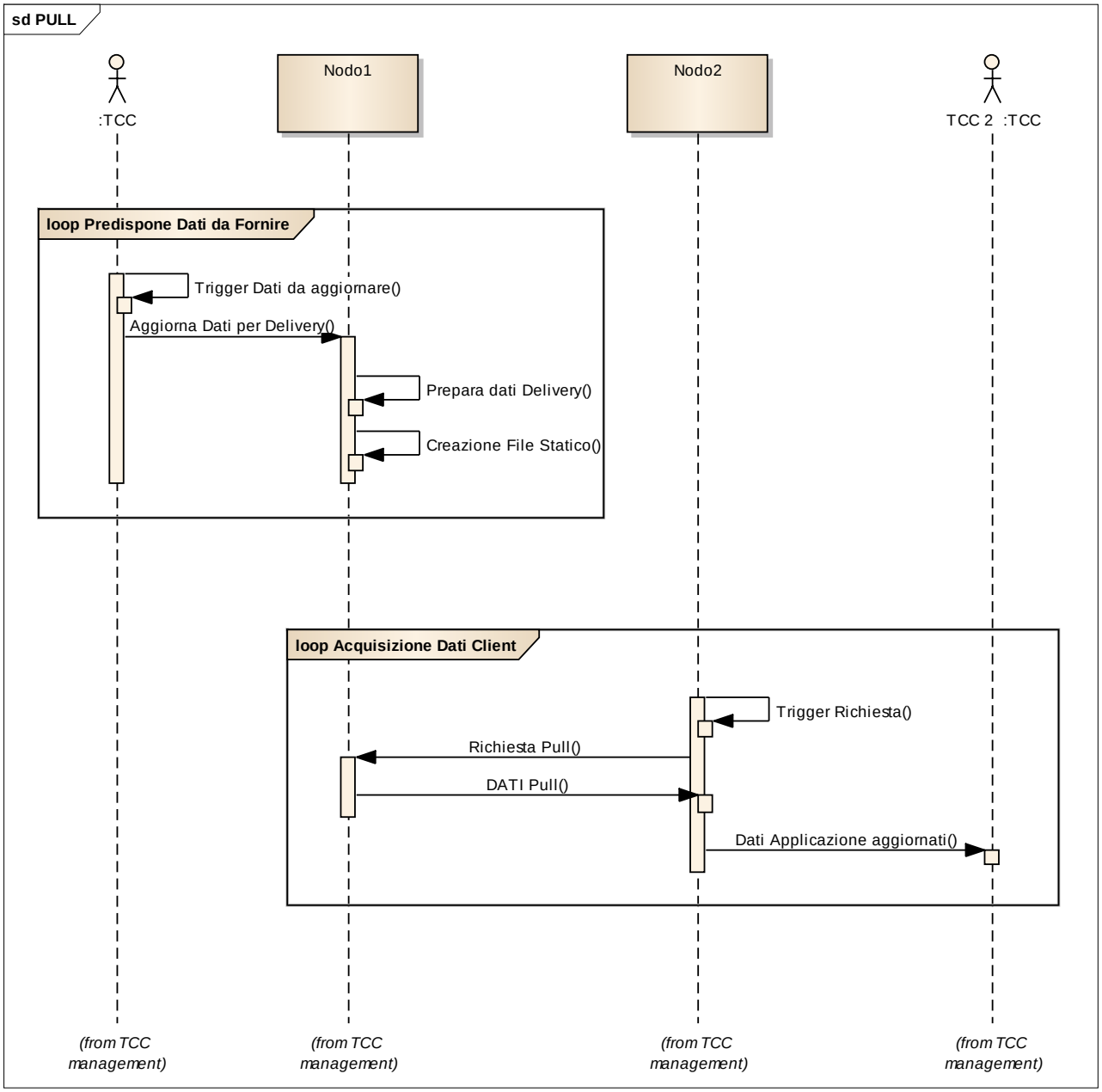
Lo stato della comunicazione col Supplier elaborato del client è riassunto nel seguente schema:



All'inizializzazione del processo di acquisizione, una volta acquisiti gli URL di riferimento dei supplier il primo tentativo di accesso se più o meno a buon fine pone lato client lo stato del supplier a ON o a OFF, e il passaggio di stato viene aggiornato potenzialmente a ogni tentativo successivo di acquisizione dei dati con le logiche di cui il client si vorrà dotare.

Nel momento in cui il processo di acquisizione termina pone convenzionalmente lo stato di connessione di tutti i supplier a OFF e interrompe l'acquisizione, senza necessariamente comunicare niente ai supplier.

Il diagramma di sequenza delle attività di questo tipo di protocollo è il seguente che individua in modo asincrono le attività di ciclo lato supplier e lato client (TCC = Traffic Control Center, si intende l'applicazione gestionale traffico del Centro di Controllo/ Sala Radio).



Nel caso di Pull Statico per semplificare le regole di accesso ai dati e le elaborazioni necessarie, si useranno url differenziati per i diversi contenuti di pubblicazione / payload, con una nomenclatura / URL di riferimento:

- Eventi
eventualmente differenziati in
    - Eventi Totali
    - Cantieri
    - Eventi non Cantieri
    - Avviso Pericoli (veicoli contromano e simili)
- Pmv
    - Anagrafica
    - Messaggi
- Sensori
    - Anagrafica
    - Misure
- Travel Times
    - Anagrafica
    - Misure
- Feedback

Ogni URL avrà il suffisso base del percorso e poi conterrà una stringa valorizzata come da seguente tabella

Informazione	**Completamento URL: nome file.xml**
Eventi totale	SituationPublication.xml
Cantieri	SituationPublicationRoadworks.xml
Eventi non Cantieri	SituationPublicationNotRoadworks.xml
Avviso Pericoli	SituationPublicationImmediateDanger.xml
Anagrafica PMV	VmsTablePublication.xml
Stato e Messaggi PMV	VmsPublication.xml
Anagrafica Sensori e Centraline	MeasurementSitesPublication.xml
Dati e Stato dei Sensori	MeasuredDataPublication.xml
Tempi di Percorrenza, anagrafica tratte	ElaboratedDataSites.xml
Tempi di Percorrenza, dati elaborati	ElaboratedData.xml
Feedback	Feedback.xml

USO File di Controllo Metadata file per test funzionamento Supplier

Per finalità di controllo del buon funzionamento del supplier in caso di dati non aggiornati lato client, si può usare un URL di controllo contenente un messaggio del supplier che indica il corretto funzionamento del sistema, piuttosto che aggiornare un dato di timestamp nel D2LogicalModel che necessiterebbe di una rielaborazione completa del file XML con tutti i dati per comprendere se ci sono variazioni con la versione precedente.

Informazione	**Metadata file**
<payload_publication>	<payload_publication>_metadata.xml

Il funzionamento del controllo metadata e la struttura del file è definita nel documento PSM 2.0 par 4.3 (http://www.datex2.eu/content/datex-ii-exchange-psm-20)

Per web services si prevede Client Pull standard per invio dei dati raggruppati nelle stesse categorie previste dal low cost profile, utilizzando quindi URL di riferimento con nomenclature simili che daranno luogo a diversi WSDL ciascuno, per prelevare le informazioni di interesse (Eventi, PMV, Lavori, ecc).

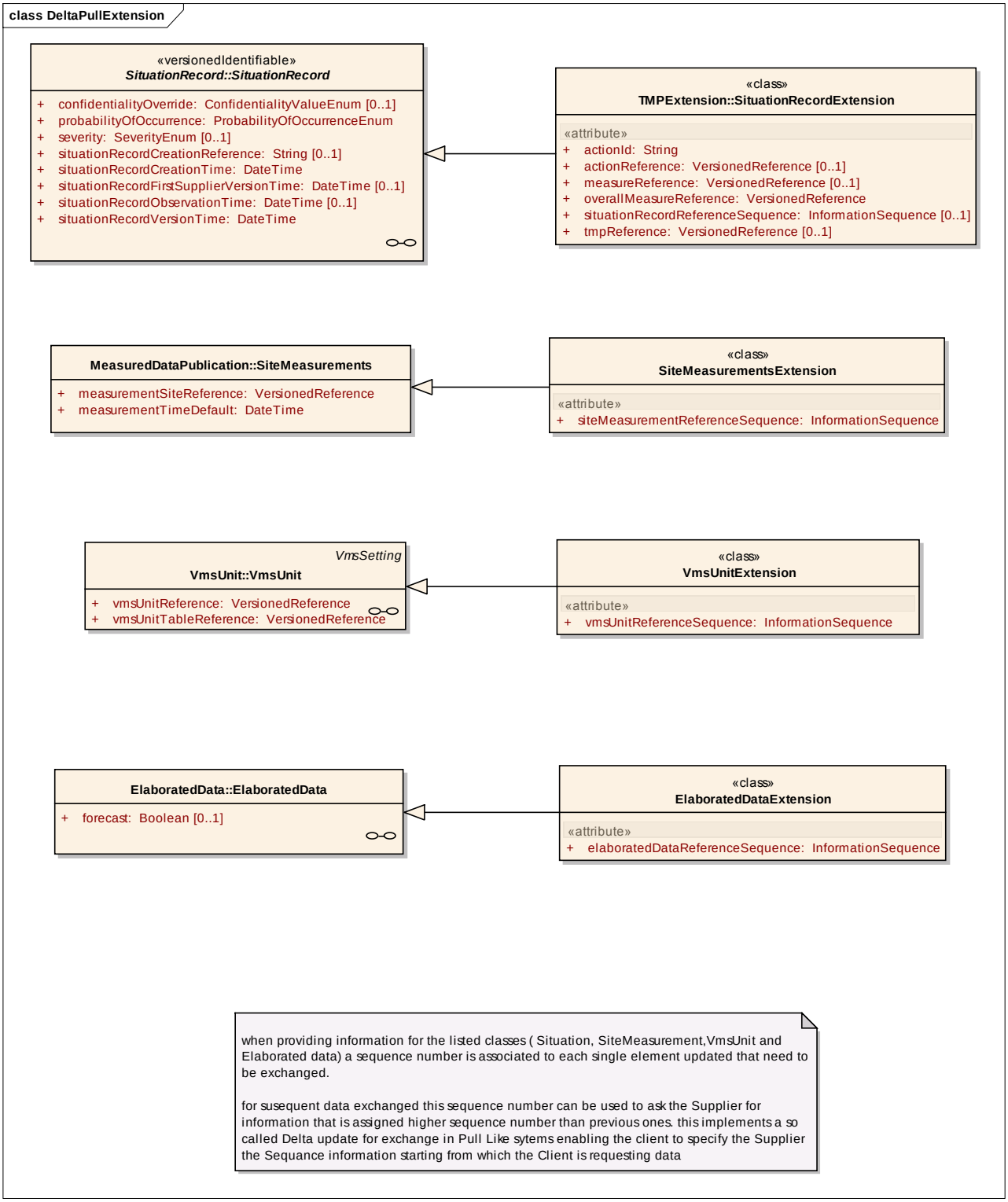
Nel Pull Web Services si richiama al posto del file statico previsto nel PULL statico http / Low Cost Profile, un servizio che riporta gli stessi contenuti ma implementato appunto tramite WebServices.

Per migliorare l'efficienza di trasmissione si ipotizza di poter implementere per i relativi sottodomini di interesse dei servizi cosiddetti "DeltaPull" per invio al Client dei soli dati aggiornati basati su un timestamp / id dell'ultimo dato ricevuto dal Client per invio dei soli Delta, non standard DATEX II ma più efficiente.

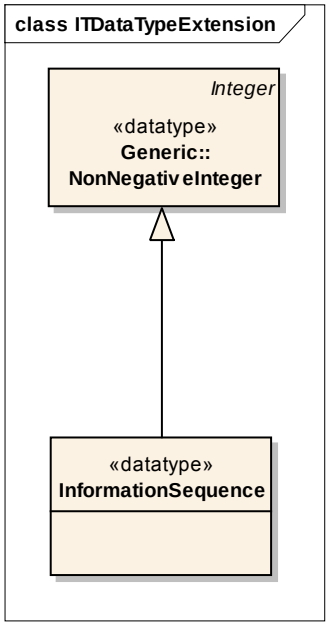
Dettagli per l'invio del timestamp/id da parte del Client

- Il sistema fornitore di informazioni associa alle informazioni elementari da trasmettere (situation record, messaggio vms, misura sensore) un progressivo di inserimento nel sistema (long integer).
- Il Client riceve la pubblicazione snapshot da Pull o da DeltaPull e ricava e memorizza il massimo progressivo ricevuto, per fare questo si sceglie di implementare il progressivo come un numerico strettamente crescente in modo che l'implementazione della funzione che determina il massimo progressivo sia unicamente identificabile indipendentemente dal client.
- Alla successiva richiesta DeltaPull il client fornisce come parametro il dato dell'ultimo e massimo progressivo ricevuto in modo che il Supplier gli fornisce le informazioni che sono associate a progressivi successivi.
- Prerequisito creare una estensione che consenta di memorizzare il timestamp associato agli elementi che si decide di inviare in modalità DeltaPull (situation record, vms message, measured data)

La seguente estensione del modello consente di traferire, insieme alle informazioni del SituationRecord, un progressivo di inserimento :



Dove :



DETTAGLIO DELTAPULL

Il WSDL standard attuale del Pull Datex II prevede che il sistema esporti il seguente metodo come servizio Datex II:

```
      <message name="inputMessage"/>
<!-- This version of the DATEX II Pull service doesn't use any input message but
its declaration is here mandatory for a few Web Service frameworks-->
      <message name="exchangeMessage">
            <part name="body" element="d2ns:d2LogicalModel"/>
      </message>
      <portType name="clientPullInterface">
            <operation name="getDatex2Data">
                  <input message="tns:inputMessage"/>
<!-- This version of the DATEX II Pull service doesn't use any input message but
its declaration is here mandatory for a few Web Service frameworks-->
                  <output message="tns:exchangeMessage"/>
            </operation>
      </portType>
```

Nel caso del DeltaPull, la proposta è quella di esportare un servizio Datex II Pull esteso nel seguente modo:

```
      <message name="inputMessage"/>
<!-- This version of the DATEX II Pull service doesn't use any input message but
its declaration is here mandatory for a few Web Service frameworks-->

      <message name="inputMessageDelta"/>
            <part name="sequenceNumber" element="xsd:long"/>
      <message name="exchangeMessage">
            <part name="body" element="d2ns:d2LogicalModel"/>
      </message>
      <portType name="clientPullInterface">
            <operation name="getDatex2Data">
                  <input message="tns:inputMessage"/>
                  <output message="tns:exchangeMessage"/>
```

```
        </operation>
        <operation name="getDeltaDatex2Data">
              <input message="tns:inputMessageDelta"/>
              <output message="tns:exchangeMessage"/>
        </operation>
  </portType>
```

Cioè lo stesso servizio Pull esporta il metodo classico getDatex2Data per la comunicazione standard e in più il metodo getDeltaDatex2Data che ha come parametro il sequence number di partenza da cui si volgiono i dati

Per web service Push si intendono servizi previsti dallo standard DATEX II che prevedono che il Supplier invii i dati tramite l'attivazione di una chiamata webservice Push implementata lato Client. La temporizzazione della chiamata è dettata da logiche che vengono definite ed implementate esclusivamente lato client.

Si distinguono:

Condition Triggered Push: attivazione della chiamata Push al momento del verificarsi di una data condizione applicativa.

Push Periodic: attivazione della chiamata Push in base an periodo ciclico prefissato.

Normalmente il Push Periodico viene utilizzato in situazioni di disponibilità dei dati periodico, per esempio per raccolta misure periodiche dai sensori stradali, o quando non sia necessario garantire una disponibilità immediata di aggiornamenti.

Il Push On Occurrence normalmente usato in contesti di situazioni che accadono in modalità casuale, garantisce la massima tempestività nella disponibilità del dato verso il Client.

Nella comunicazione Push il wsdl standandard Datex II è illustrato nella figura:

```
▼<wsdl:message name="putDatex2DataResponse">
   <wsdl:part element="ns1:d2LogicalModel" name="body"></wsdl:part>
 </wsdl:message>
▼<wsdl:message name="putDatex2Data">
   <wsdl:part element="ns1:d2LogicalModel" name="body"></wsdl:part>
 </wsdl:message>
▼<wsdl:portType name="supplierPushItSpInterface">
  ▼<wsdl:operation name="putDatex2Data" parameterOrder="body">
     <wsdl:input message="tns:putDatex2Data" name="putDatex2Data"></wsdl:input>
     <wsdl:output message="tns:putDatex2DataResponse" name="putDatex2DataResponse"></wsdl:output>
   </wsdl:operation>
 </wsdl:portType>
▼<wsdl:binding name="SupplierPushItSpServiceSoapBinding" type="tns:supplierPushItSpInterface">
   <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  ▼<wsdl:operation name="putDatex2Data">
     <soap:operation soapAction="http://datex2.eu/wsdl/supplierPush/2_0/putDatex2Data" style="document"/>
    ▼<wsdl:input name="putDatex2Data">
       <soap:body use="literal"/>
     </wsdl:input>
    ▼<wsdl:output name="putDatex2DataResponse">
       <soap:body use="literal"/>
     </wsdl:output>
   </wsdl:operation>
 </wsdl:binding>
▼<wsdl:service name="SupplierPushItSpService">
  ▼<wsdl:port binding="tns:SupplierPushItSpServiceSoapBinding" name="supplierPushItSpSoapEndPoint">
     <soap:address location="http://100.52.8.110:8080/datex2ws/PushItSp?wsdl"/>
   </wsdl:port>
 </wsdl:service>
```

FIGURA 1 PUSH.WSDL

Importante far notare che nel caso di comunicazione Push è previsto un messaggio di response che indica al supplier l'avvenuta ricezione dell'informazione.

Il messaggio di response è una pubblicazione Datex II che comunemente contiene solo la classe Exchange con valorizzati gli attributi di response, delivery break e keep alive.

Si fa notare come esempio che in termini implementativi il webservice lato cliente esporterà un metodo come segue (java + jax-ws webservice):

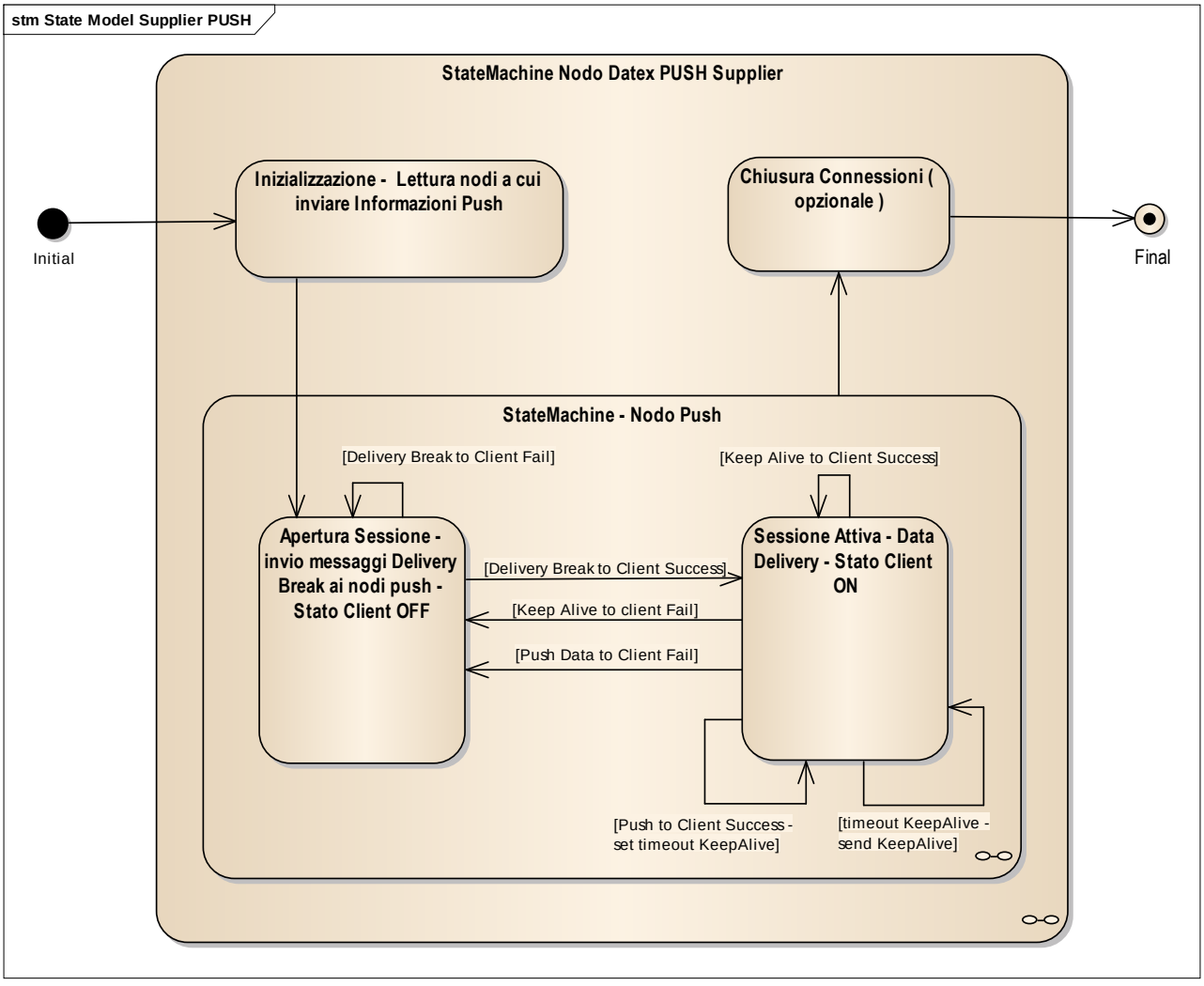
public void putDatex2Data(D2LogicalModel body){}

il parametro body in questo caso funziona sia da parametro in input che come parametro in output per l'invio del dato e del messaggio di response.

Sara compito del supplier andare al leggere le informazioni all'interno dell'oggetto d2logicalModel per verificare la response al messaggio inviato.

Per un funzionamento consistente del Push si devono prevedere anche delle fasi di allineamento dei dati dal Supplier come fasi di avvio del sistema o di ripristino del collegamento dopo la mancata comunicazione per scollegamenti di rete o attività di manutenzione. In tali casi si prevede l'utilizzo di una chiamata Pull a carico del Client per risincronizzare la base dati e recuperare i contenuti non trasmessi in fase di scollegamento.

La comunicazione Client – supplier con l'utilizzo della metodologia Push avviene secondo il paradigma illustrato negli schemi seguenti.



La figura illustra la macchina a stati per un nodo Datex in comunicazione PUSH con un dato supplier.

Dopo la lettura dei nodi da connettere il supplier deve prevedere un primo stato di "Apertura Sessione" che consente al sistema di verificare che il client sia operativo.

Si individua la necessita di inviare in Push un messaggio DATEX II denominato "*Delivery Break*" che non conterrà contenuto Payload, e che notifica al Client la disponibilità del Supplier.

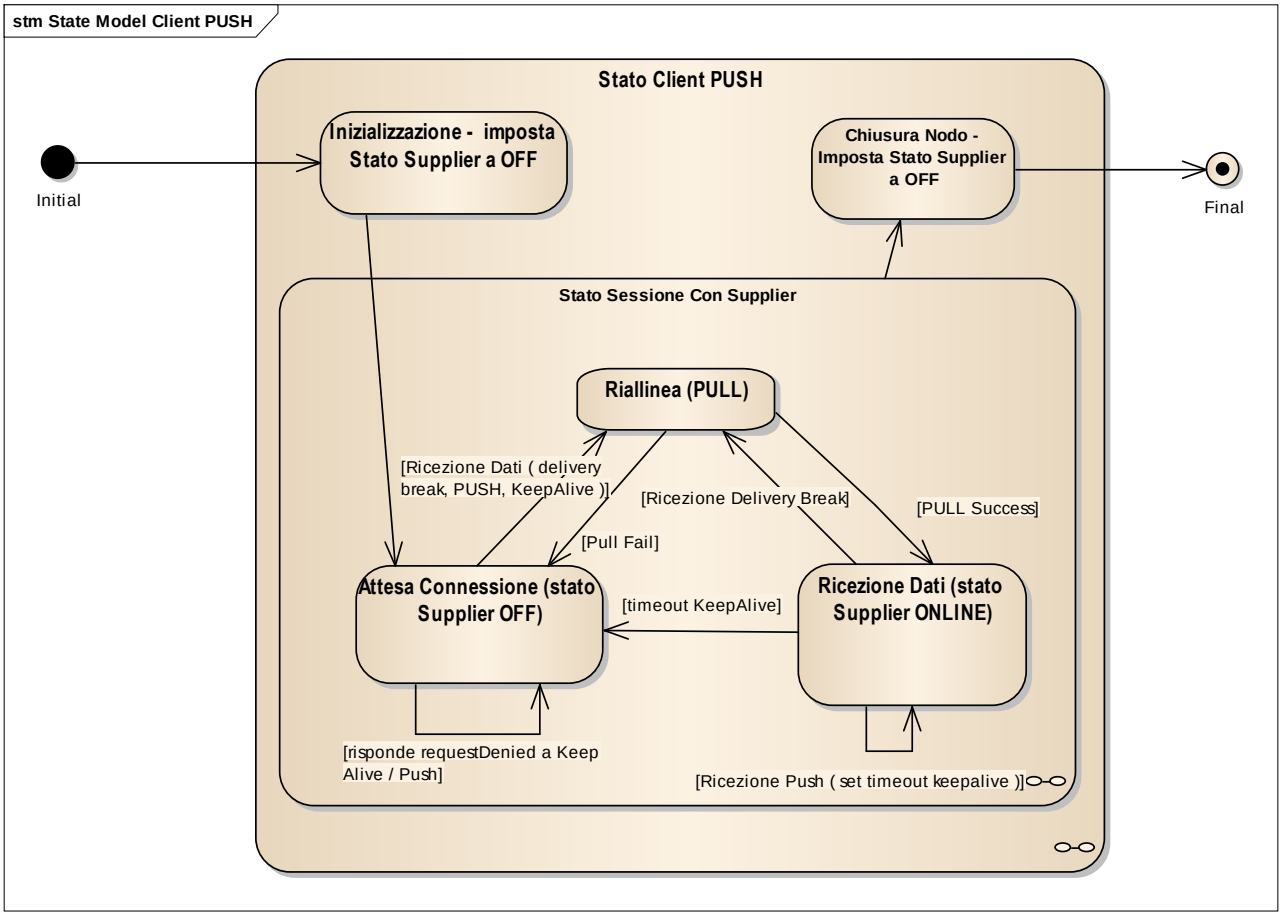
In seguito al corretto invio del messaggio "*Delivery Break*" il Supplier considera che la Sessione è Attiva per quel dato Client ed inizia a inviare continuativamente i dati Push quando disponibili o il messaggio

di *"Keep Alive"*, in caso di assenza di necessità di aggiornamento contenuti, per le necessità di controllo dello stato del canale.

In caso di errori di comunicazione che intervengano nelle notifiche Push e KeepAlive, dovuti a mancanze di Rete o eventuali chiusure del servizio lato Client, il Supplier indica che la sessione con il Client è OFF e rientra nello stato di "Apertura Sessione" tentando nuovamente la connessione al servizio mediante messaggio di *Delivery Break*.

In caso di chiusura del servizio è opzionale porre lo stato del Client a OFF. Ovviamente cessando le notifiche Push e KeepAlive nel timeout prefissato il Client considererà il Supplier non disponibile.

La macchina a stati lato Client è la seguente:

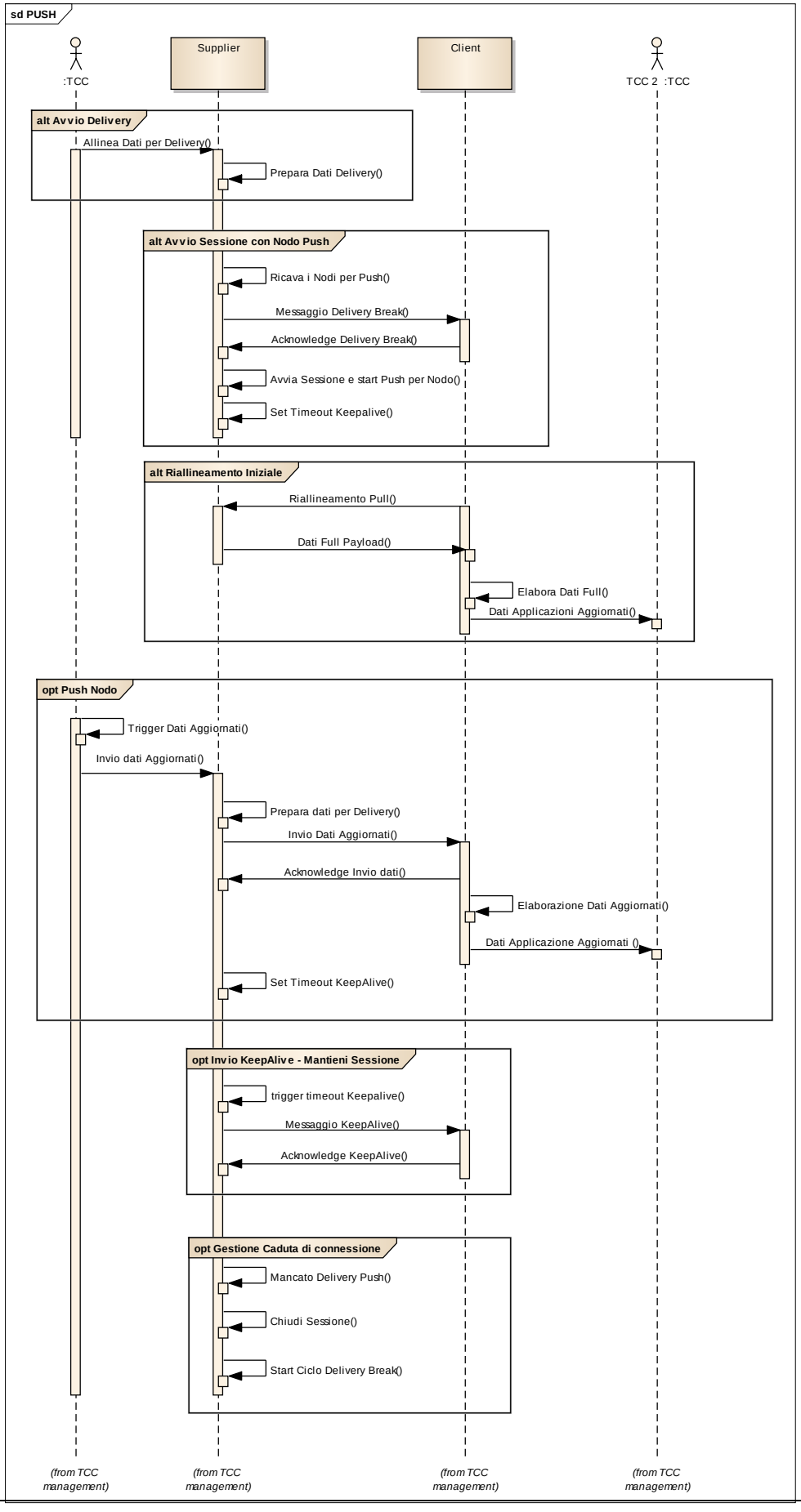


Il client dopo una fasi di inizializzazione si pone in uno stato di attesa connessione in cui aspetta la notifica di un messaggio da parte del client (normalmente *Delivery Break).*

Dopo la ricezione di questo messaggio il Client deve effettuare un riallineamento dati dal Supplier e quindi effettua una chiama PULL per ripristinare tutto il contenuto corrente e successivamente si pone nuovamente in uno stato di normale ricezione di attesa delle successive notifiche Push o KeepAlive in caso di mancanza di disponibilità dei dati.

In caso di assenza comunicazione (mancata ricezione Push di contenuti o KeepAlive) il Client porrà il Supplier in stato di scollegamento in attesa di nuovi messaggi che porteranno ad un nuovo riallineamento.

I Diagrammi di sequenza che riportano i vari passaggi e aggiornamenti di stato del sistema sono illustrati nella seguente figura.

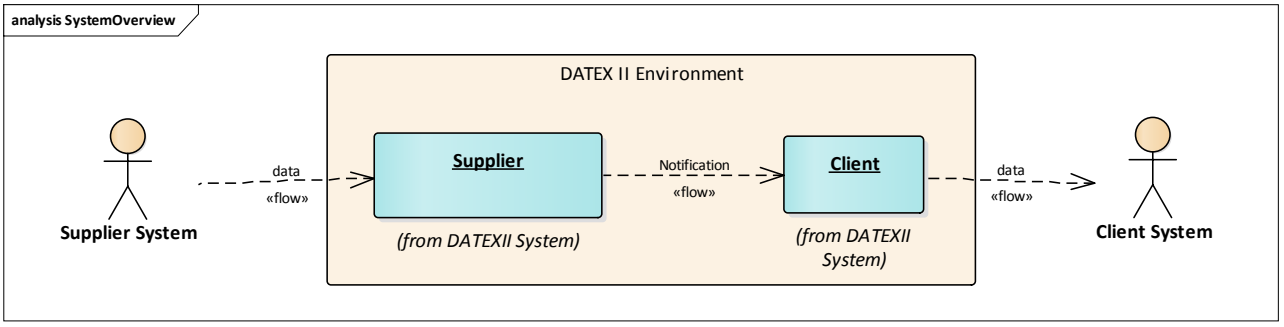

sd PUSH
:TCC
Supplier
Client
TCC 2 :TCC
alt Avvio Delivery
Allinea Dati per Delivery()
Prepara Dati Delivery()
alt Avvio Sessione con Nodo Push
Ricava i Nodi per Push()
Messaggio Delivery Break()
Acknowledge Delivery Break()
Avvia Sessione e start Push per Nodo()
Set Timeout Keepalive()
alt Riallineamento Iniziale
Riallineamento Pull()
Dati Full Payload()
Elabora Dati Full()
Dati Applicazioni Aggiornati()
opt Push Nodo
Trigger Dati Aggiornati()
Invio dati Aggiornati()
Prepara dati per Delivery()
Invio Dati Aggiornati()
Acknowledge Invio dati()
Elaborazione Dati Aggiornati()
Dati Applicazione Aggiornati ()
Set Timeout KeepAlive()
opt Invio KeepAlive - Mantieni Sessione
trigger timeout Keepalive()
Messaggio KeepAlive()
Acknowledge KeepAlive()
opt Gestione Caduta di connessione
Mancato Delivery Push()
Chiudi Sessione()
Start Ciclo Delivery Break()
(from TCC management)
(from TCC management)
(from TCC management)
(from TCC management)

Note di Implementazione

1. Il sistema supplier invia i dati in modalità push verso il client richiamando la funzione Push del webservice implementato lato Client.
2. Per il monitoraggio del canale il Supplier utilizza un apposito messaggio di “Keep Alive” che è implementato un messaggio DATEX II contentente la sola Classe Exchange con settato a true l’attributo keepAlive.
    - Nel caso che la connessione sia stabilita già il Client risponde con un messaggio di risposta positivo (proprietà della risposta acknowledged)
    - Nel caso che il Client ancora non abbia intrapreso la connessione per la prima volta o dopo uno scollegamento risponde con un messaggio negativo (requestDenied)
3. Per una questione di economicità di implementazione i nomi dei servizi possono essere raggruppati nelle stesse categorie di contenuti previste dal low cost profile, utilizzando quindi URL di riferimento con nomenclature simili che daranno luogo a diversi WSDL ciascuno, per prelevare le informazioni di interesse (Eventi, PMV, Lavori, ecc).

Questo paradigma utilizza solo il servizio PUSH esposto dal Client sia per l'invio dei dati a regime, che per l'invio dei dati di riallineamento da parte del Supplier.

Si considera che uno scambio dati deve avvenire fra il **Sistema Supplier** e il **Sistema Client** per necessità di allineamento di dati quando il *Sistema Supplier* ha una informazione aggiornata da rilasciare al *Sistema Client*. Lo scambio dati avviene nel sistema DATEX II fra il **DATEX Supplier** e il **DATEX Client**, riferiti più brevemente con i termini di **Supplier** e **Client**



- Il sistema di scambio dati deve garantire che il *Sistema Client* possa riallinearsi rispetto al *Sistema Supplier* in un qualsiasi momento, per necessità del *Sistema Client* stesso o in seguito ad errori di rete o di indisponibilità dello scambio o dei sistemi.
- Il *Client* deve poter richiedere due tipi di riallineamento:
  - **Riallineamento totale**, cioè richiesta di tutti dati attivi (snapshot) che deve poter essere richiesto in qualsiasi momento.
  - **Riallineamento incrementale (delta)**, cioè richiesta dei dati che il client non ha ancora ricevuto (disconnessione temporanea), ma che il supplier ha già preparato per l'invio ma non è stato possibile inviarlo (errori di rete, servizi non funzionanti).
- Il *Supplier* fornendo i dati deve **distinguere** due condizioni:
  - quando lo *scambio di informazione* ***non completa*** *la fase di riallineamento* (senza che con questo scambio si realizzi una condizione di allineamento dati completo fra Supplier e Client)
  - quando lo *scambio di informazione* ***completa*** *l'allineamento dei due sistemi*..

  Il supplier è in grado quindi di segnalare al client l'inizio e la fine di un riallineamento

Ricordando che un Messaggio di Scambio Dati DATEX consiste nella composizione di una parte di contenuto di classi *Exchange* e opzionalmente un *Payload*, in questo paradigma si sfruttano i seguenti Messaggi valorizzando opportunamente alcuni attributi di *Exchange*:

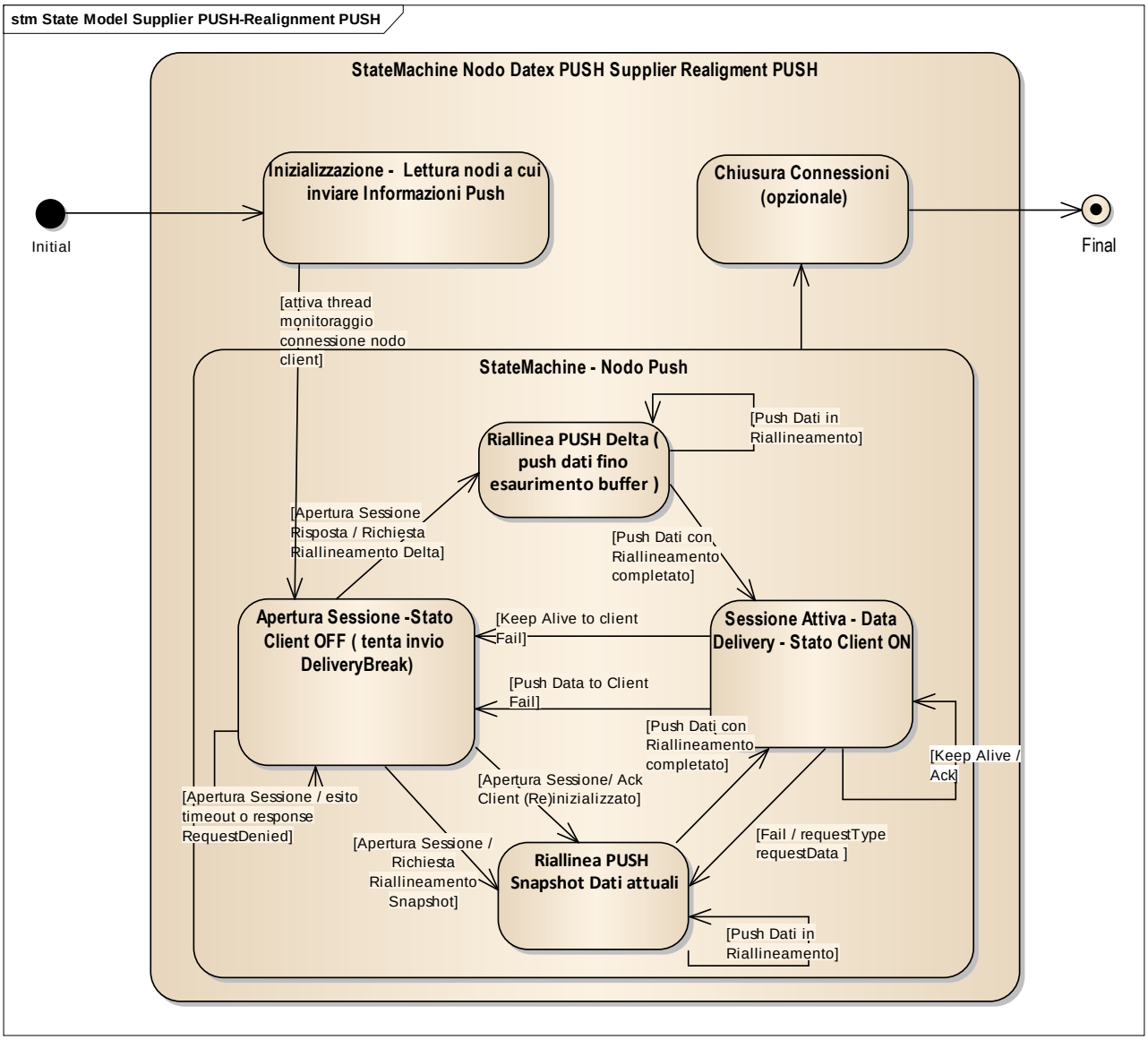
- ***deliveryBreak*** inviato dal supplier
  - con valore **True** con solo contenuto della classe di Exchange per la richiesta di apertura connessione
  - con valore **True** con Exchange e Payload per indicare che il dato inviato è di riallineamento
- ***keepAlive*** per indicare messaggio di keepAlive da Supplier
- ***response***
  - **Acknowledge** per indicare messaggio ricevuto
  - **requestDenied** per indicare messaggio ricevuto ma non corretto o rifiutato
- ***requestType***
  - **requestData** usato dal client in risposta per richiedere un riallineamento snapshot

- **requestHystoricalData** usato dal client per richiedere un riallineamento Delta

In base alla valorizzazione dei precedenti attributi, per comprensione e semplificazione dei messaggi da indicare nei diagrammi, introduciamo la seguente nomenclatura per distinguere i vari tipi di messaggio:

TIPO Messaggio	Verso Supplier Client	Dettaglio Contenuto Classi Exchange	Presenza Contenuto Classi Payload
Apertura Sessione	S → C	deliveryBreak=True	NO
Push Dati in Riallineamento (non completato)	S → C	deliveryBreak=True	SI
Push Dati con Riallineamento completato	S → C	deliveryBreak=False	SI/NO
KeepAlive (nessun dato da scambiare ma sistema disponibile e funzionante)	S → C	keepAlive=True	NO
Push Dati con Sessione attiva	S → C	deliveryBreak = null, keepAlive = null	SI
Ack: Push ricevuto Correttamente	S ←C	response = Acknowledge	NO
Fail: Errore in ricezione push	S ←C	response = requestDenied	NO
Risposta Client con Richiesta Riallineamento Delta	S ←C	requestType = requestHistoricalData	NO
Risposta Client con Richiesta Riallineamento Snapshot	S ←C	requestType = requestData	NO

La macchina a stati del Supplier in questo paradigma è la seguente:



Si distinguono i seguenti stati:

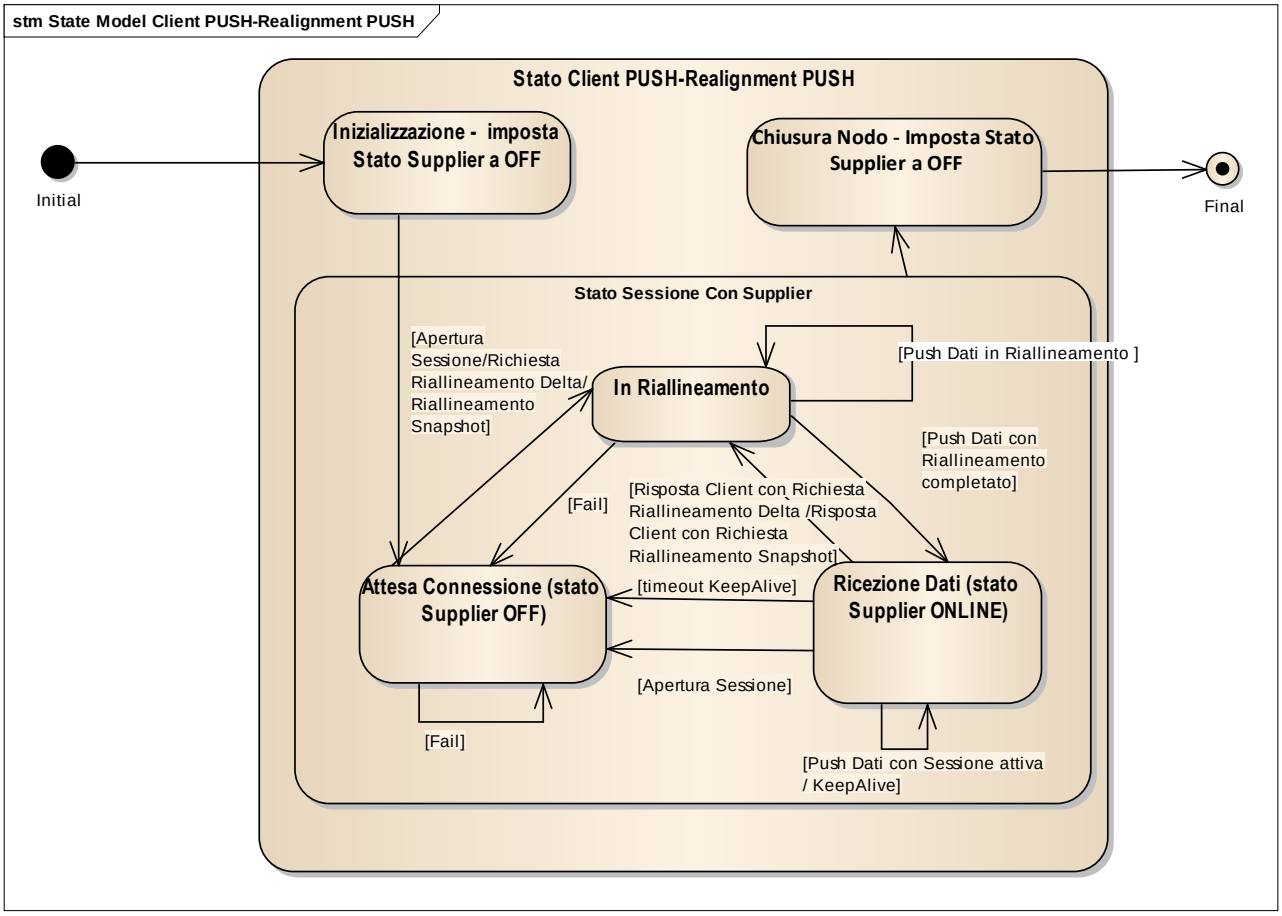
- **Inizializzazione**: Il supplier in questo stato legge i nodi configurati nel sistma a cui deve inviare inviare informazioni in push e attiva un thread per ogni nodo che tiene conto dello stato di scambio.
- **Apertura Sessione**: Il supplier passa in questo stato quando apre i thread interni per attivare le sessioni di scambio dati con i client.
  In questo stato il supplier tenta ciclicamente di inviare messaggi di apertura Sessione e in seguito all'esito positivo gestisce la risposta del Client che indicherà la modalità di riallineamento (Delta o Snapshot)
  Riallineamento Delta: Il supplier in questo stato:
  - invia i dati che erano rimasti non inviati (buffer di invio). Il push dei dati sarà con messaggi di *Push In Riallineamento* fino a completamento del buffer quando si invierà un *Push Dati Riallineamento Completato*.
  - passa in stato *Apertura Sessione* se fallisce un push verso il client
  - passa in stato *Sessione Attiva* se è finito il riallineamento
- **Riallineamento Snapshot:** Il supplier in questo stato:

- invia in push singolarmente tutti i dati attivi per allineare il client alla situazione attuale dei dati. Il push dei dati sarà con messaggi di *Push In Riallineamento* fino a completamento del buffer quando si invierà un *Push Dati Riallineamento Completato*.
- Passa in stato *Apertura Sessione* se un push fallisce
- passa in stato *Sessione Attiva* se è finito il riallineamento

- **Sessione Attiva**: Il supplier in questo stato:
  - invia i dati in push non appena disponibili
  - invia in push (con un certo timeout) Messaggi *KeepAlive* per verificare la presenza del client quando non ci sono dati da inviare.
  - passa in stato *Apertura Sessione* se fallisce un push di dati o un push di *keepAlive*
  - passa in stato di *Riallineamento Delta* se il client risponde ad un push di dati con *Richiesta di Riallineamento Delta*
  - passa in stato di **Riallineamento Snapshot** se il client risponde ad un push di dati con *Richiesta di Riallineamento Snapshot.*
- **Chiusura Connessioni:** Il supplier chiede tutti i thread delle connessioni attive configurate e esce.

E' importante notare che il Supplier, comandato dalla risposta del Client , prima di passare in *Sessione Attiva* deve passare da uno degli stati di riallineamento in base alla risposta del Client all'apertura Sessione. In questo modo il Supplier ha la garanzia di aver allineato il Client in base al proprio stato interno.

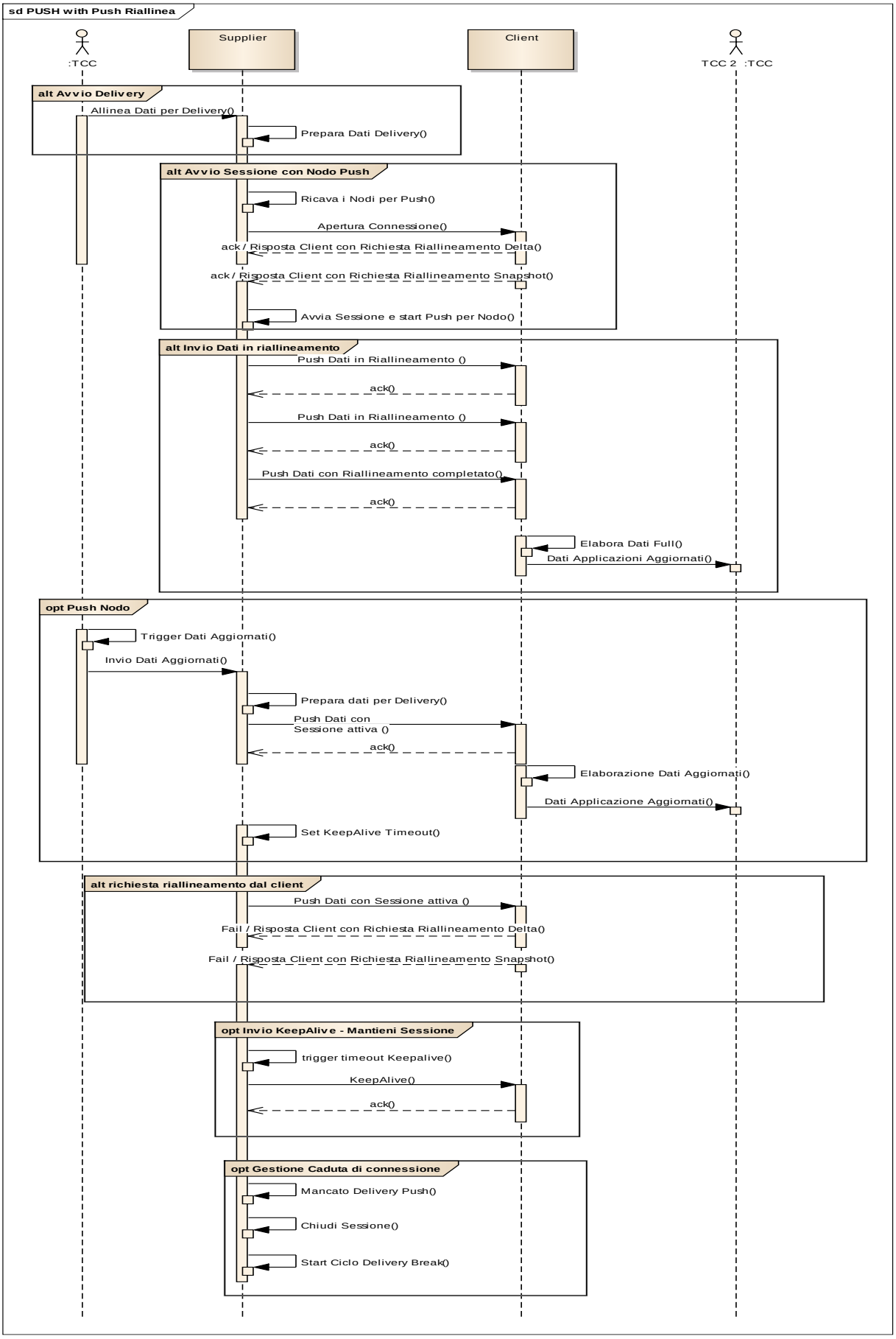
La macchina a stati del client è la seguente:



Si distinguono i seguenti stati:

- **Inizializzazione**: In questo caso il client setta inizializza lo stato della connessione col supplier ad OFF e passa in *Attesa Connessione* da parte del supplier.
- **Attesa Connessione**: Il client in questo stato:
    - Risponde ad una ricezione di messaggio di *Apertura Sessione* da parte del supplier con:
        - Messaggio **Richiesta Riallineamento Snapshot** se è la prima volta che si connette con il supplier o se vuole un riallineamento totale dei dati
        - Messaggio **Richiesta Riallineamento Delta** se vuole ricevere i dati non ancora inviati.
    - Transita in stato *In Riallineamento*
- **In Riallineamento**: In client in questo stato:
    - Riceve **Push Dati in Riallineamento** e risponde con messaggio **Ack** se il dato è corretto e passa in stato *Ricezione Dati* quando riceve il messaggio **Push Dati** con **Riallineamento completato** .
    - Riceve **Push Dati in Riallineamento** e risponde con messaggio **Fail** se il dato ricevuto non è corretto e di conseguenza passando in stato **Attesa connessione** per una nuova sessione.
- **Ricezione Dati:** Il client in questo stato
    - riceve **Push Dati** e risponde con messaggio **Ack**
    - riceve **Push Dati** e risponde con messaggio **Richiesta Riallineamento Snapshot** per richiesta riallineamento totale
    - riceve **Push Dati** e risponde con messaggio **Richiesta Riallineamento Delta** per richiesta riallineamento delta
    - Se non sta ricevendo dati controlla se sta ricevendo messaggi **KeepAlive** da parte del supplier, in caso di mancanza di keepalive passa in stato *Attesa Connessione* in quanto la sessione con il client si è interrotta.
- **Chiusura Nodo**: In questo stato il client chiude qualsiasi connessione con il supplier e risponde con messaggio **Fail** ad ogni richiesta da parte di un supplier.

I Diagrammi di sequenza che riportano i vari messaggi e aggiornamenti di stato del sistema sono illustrati nella seguente figura.


sd PUSH with Push Riallinea
:TCC
Supplier
Client
TCC 2 :TCC
alt Avvio Delivery
Allinea Dati per Delivery()
Prepara Dati Delivery()
alt Avvio Sessione con Nodo Push
Ricava i Nodi per Push()
Apertura Connessione()
ack / Risposta Client con Richiesta Riallineamento Delta()
ack / Risposta Client con Richiesta Riallineamento Snapshot()
Avvia Sessione e start Push per Nodo()
alt Invio Dati in riallineamento
Push Dati in Riallineamento ()
ack()
Push Dati in Riallineamento ()
ack()
Push Dati con Riallineamento completato()
ack()
Elabora Dati Full()
Dati Applicazioni Aggiornati()
opt Push Nodo
Trigger Dati Aggiornati()
Invio Dati Aggiornati()
Prepara dati per Delivery()
Push Dati con Sessione attiva ()
ack()
Elaborazione Dati Aggiornati()
Dati Applicazione Aggiornati()
Set KeepAlive Timeout()
alt richiesta riallineamento dal client
Push Dati con Sessione attiva ()
Fail / Risposta Client con Richiesta Riallineamento Delta()
Fail / Risposta Client con Richiesta Riallineamento Snapshot()
opt Invio KeepAlive - Mantieni Sessione
trigger timeout Keepalive()
KeepAlive()
ack()
opt Gestione Caduta di connessione
Mancato Delivery Push()
Chiudi Sessione()
Start Ciclo Delivery Break()