



# TVirtualInterface

...un caso d'uso reale (eInvoice4D)



SPEAKER: MAURIZIO DEL MAGNO





## MAURIZIO DEL MAGNO DEVELOPER



**Lev@nte software**



**i-ORM**

[github.com/mauriziodm/iORM](https://github.com/mauriziodm/iORM)

**DJSON**

[github.com/mauriziodm/DJSON](https://github.com/mauriziodm/DJSON)



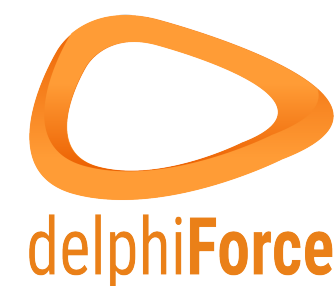
[mauriziodm@levantesw.it](mailto:mauriziodm@levantesw.it)

[mauriziodelmagno@gmail.com](mailto:mauriziodelmagno@gmail.com)



[facebook.com/maurizio.delmagno](https://facebook.com/maurizio.delmagno)

iORM + DJSON (group)



Membro fondatore

**eInvoice4D**

<https://github.com/delphiforce/eInvoice4D>



# TVirtualInterface

...un caso d'uso reale (eInvoice4D)



SPEAKER: MAURIZIO DEL MAGNO

# eInvoice4D

libreria per la fattura elettronica

# eInvoice4D

libreria per la fattura elettronica

- IFatturaElettronicaType
- From/To XML as `string`, `file`, `stream`, `base64`
- Validators
- Send/Receive to providers (*fatture attive/passive, notifiche ecc.*)



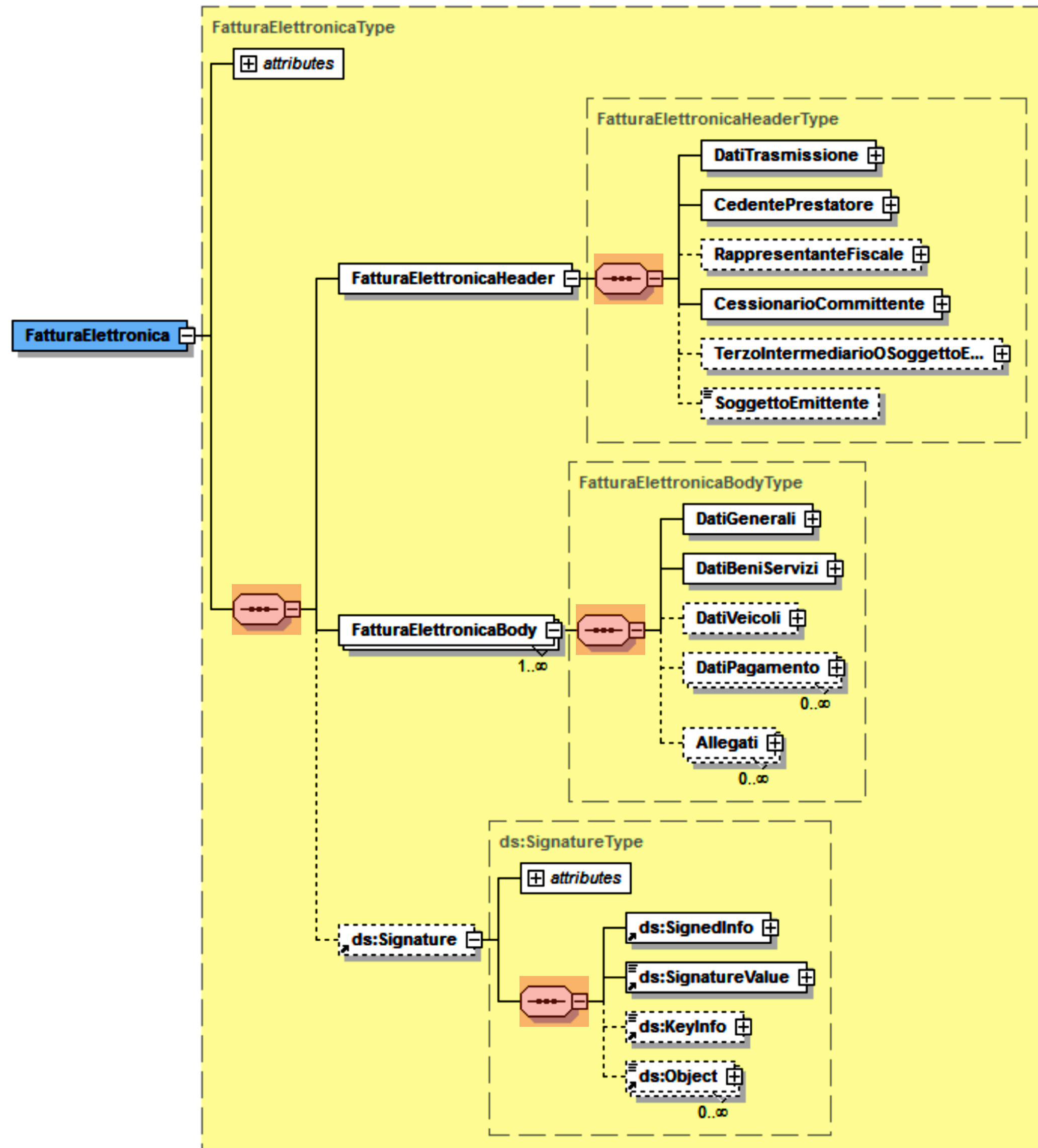
# Fattura Elettronica

l'oggetto

# Fattura Elettronica

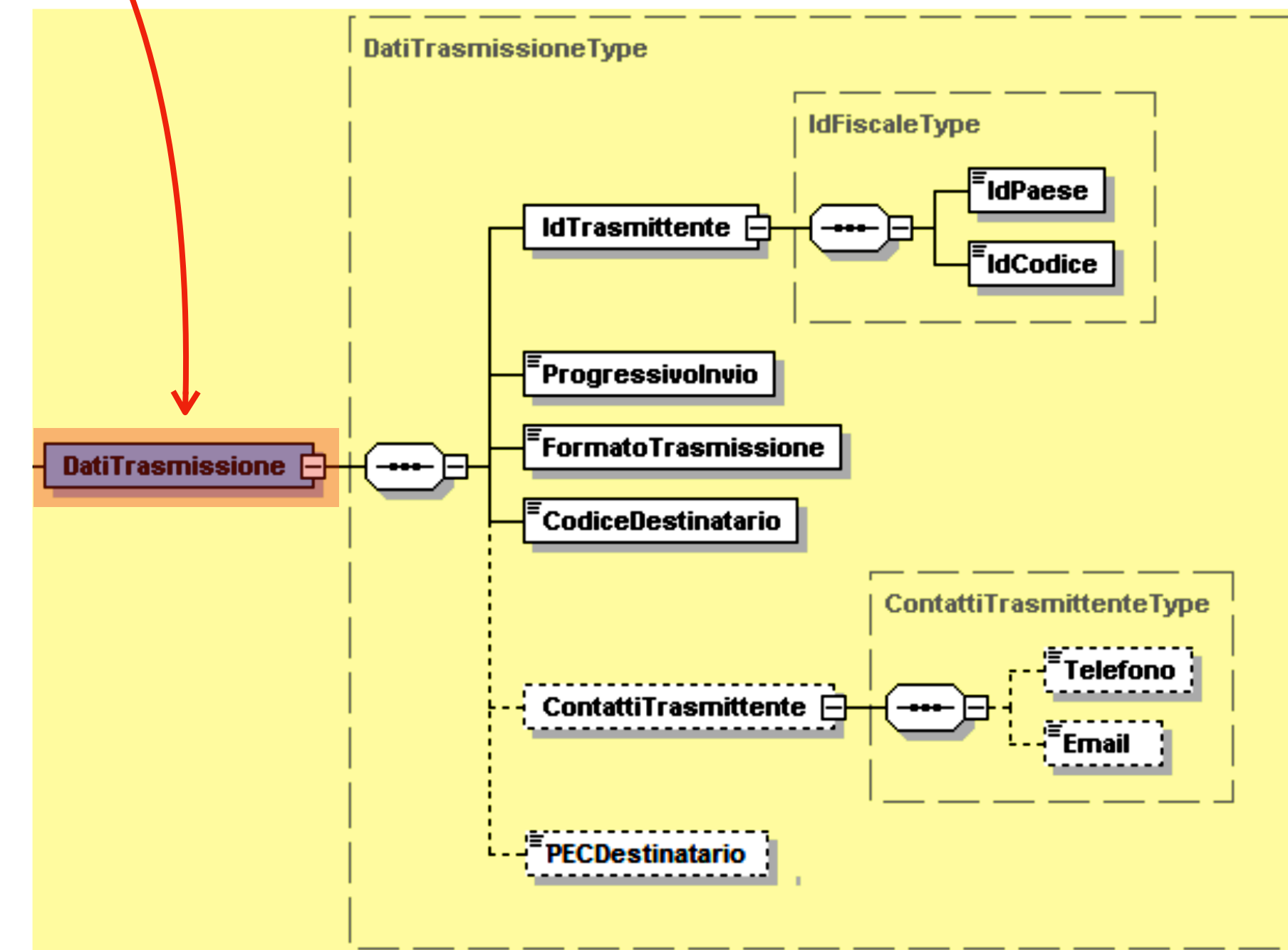
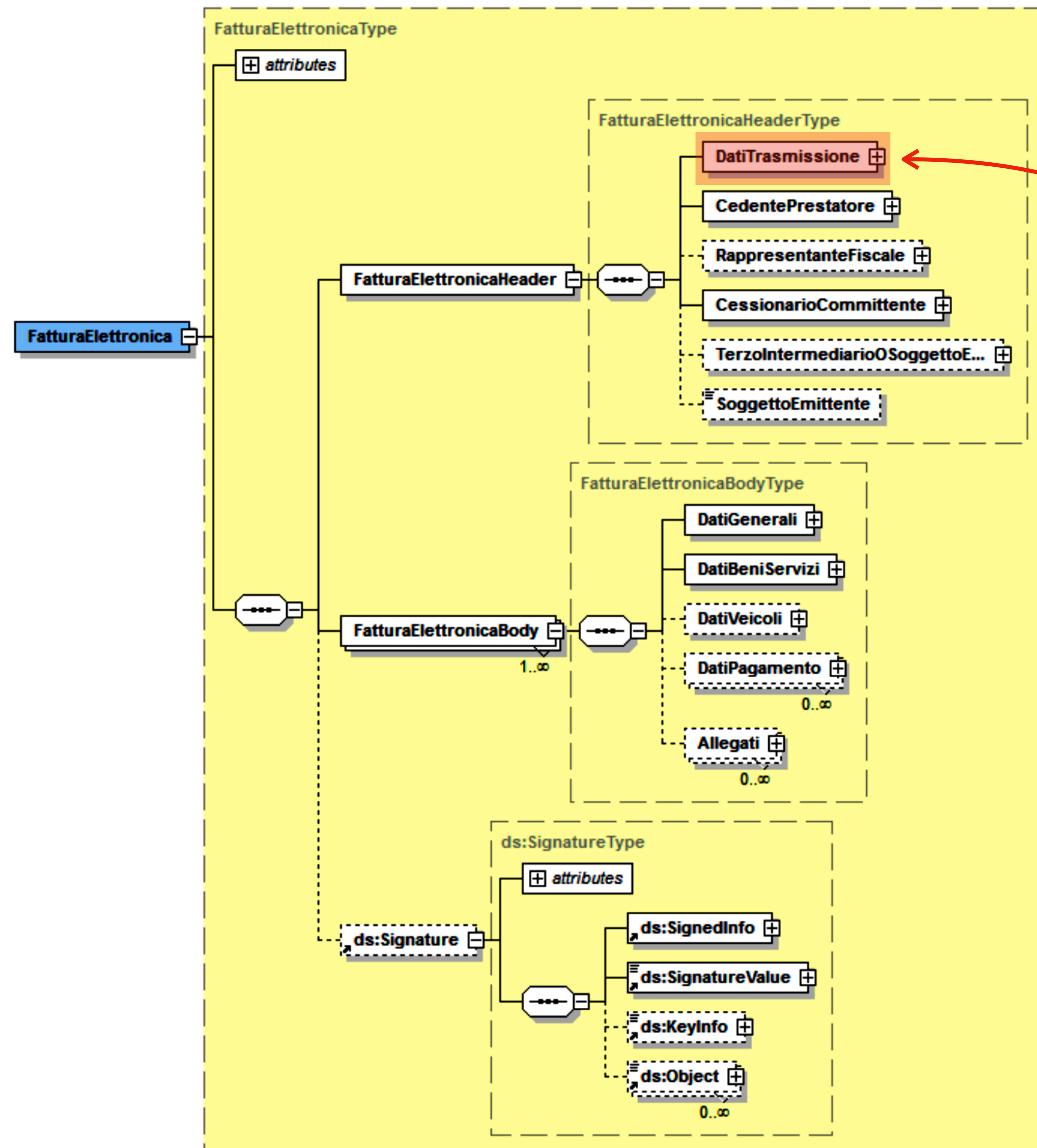
l'oggetto

Interfaces



# Fattura Elettronica

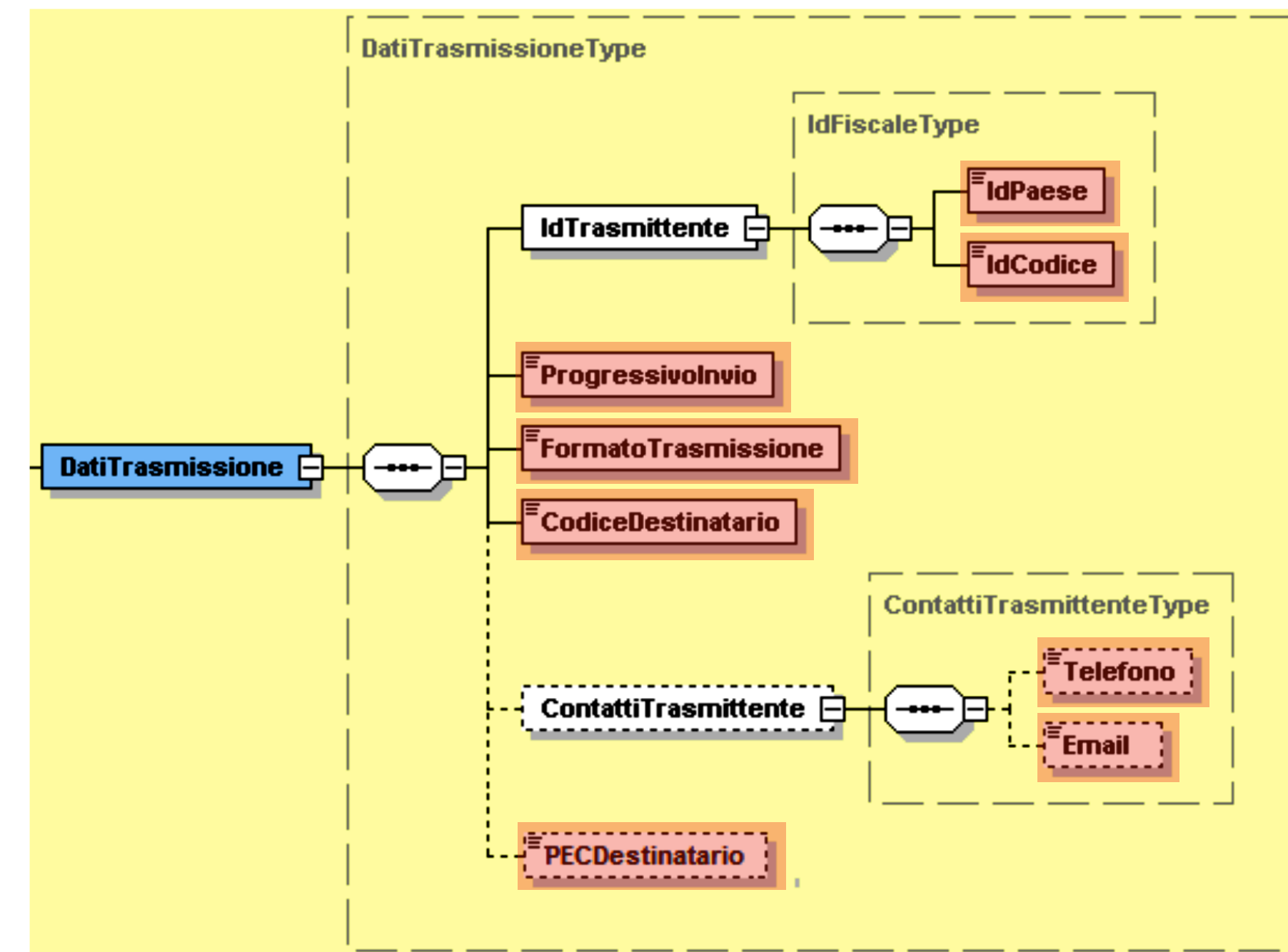
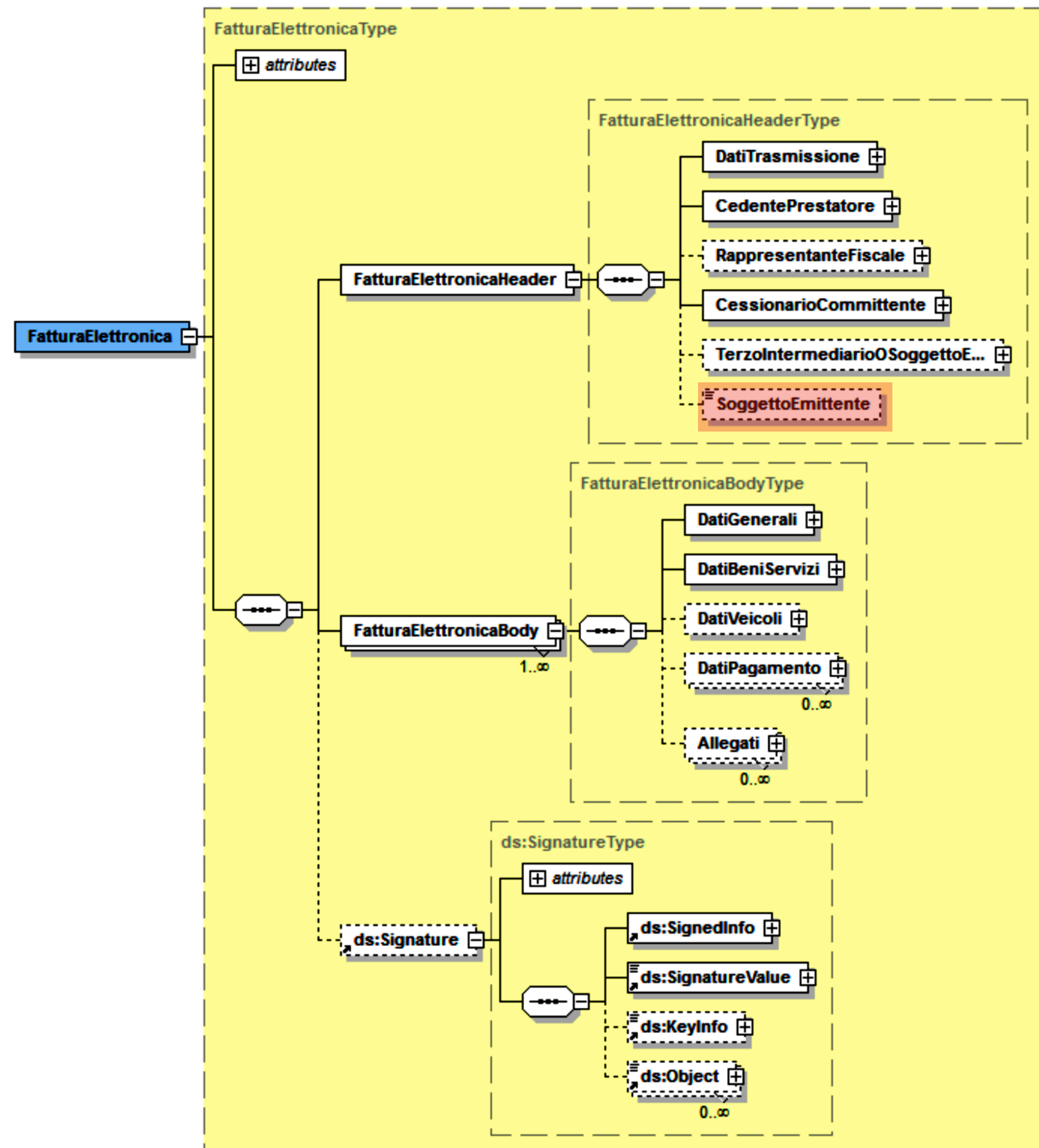
l'oggetto



# Fattura Elettronica

l'oggetto

1 - Proprietà **semplici** (*string, integer, decimal, date, time*)



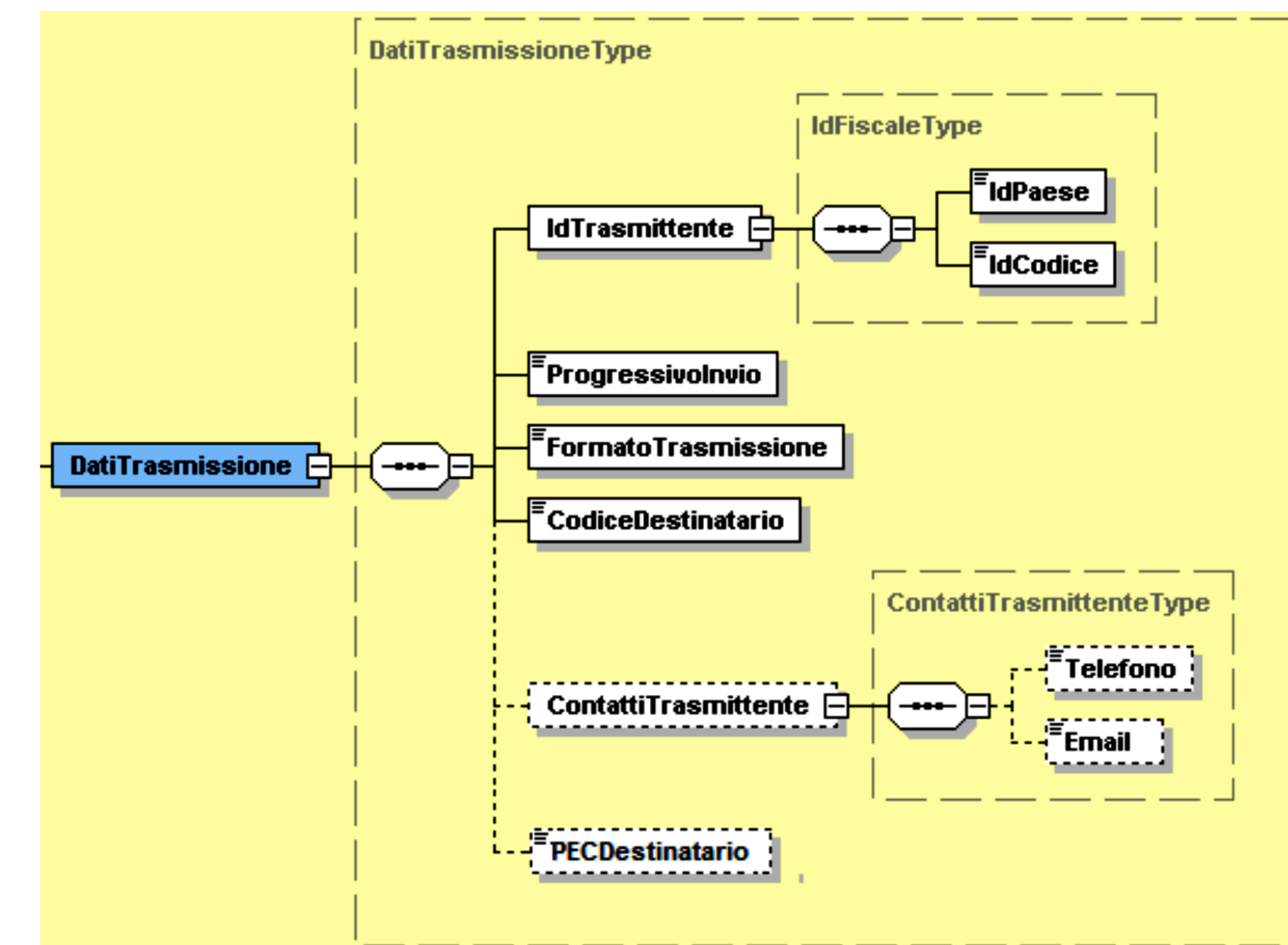
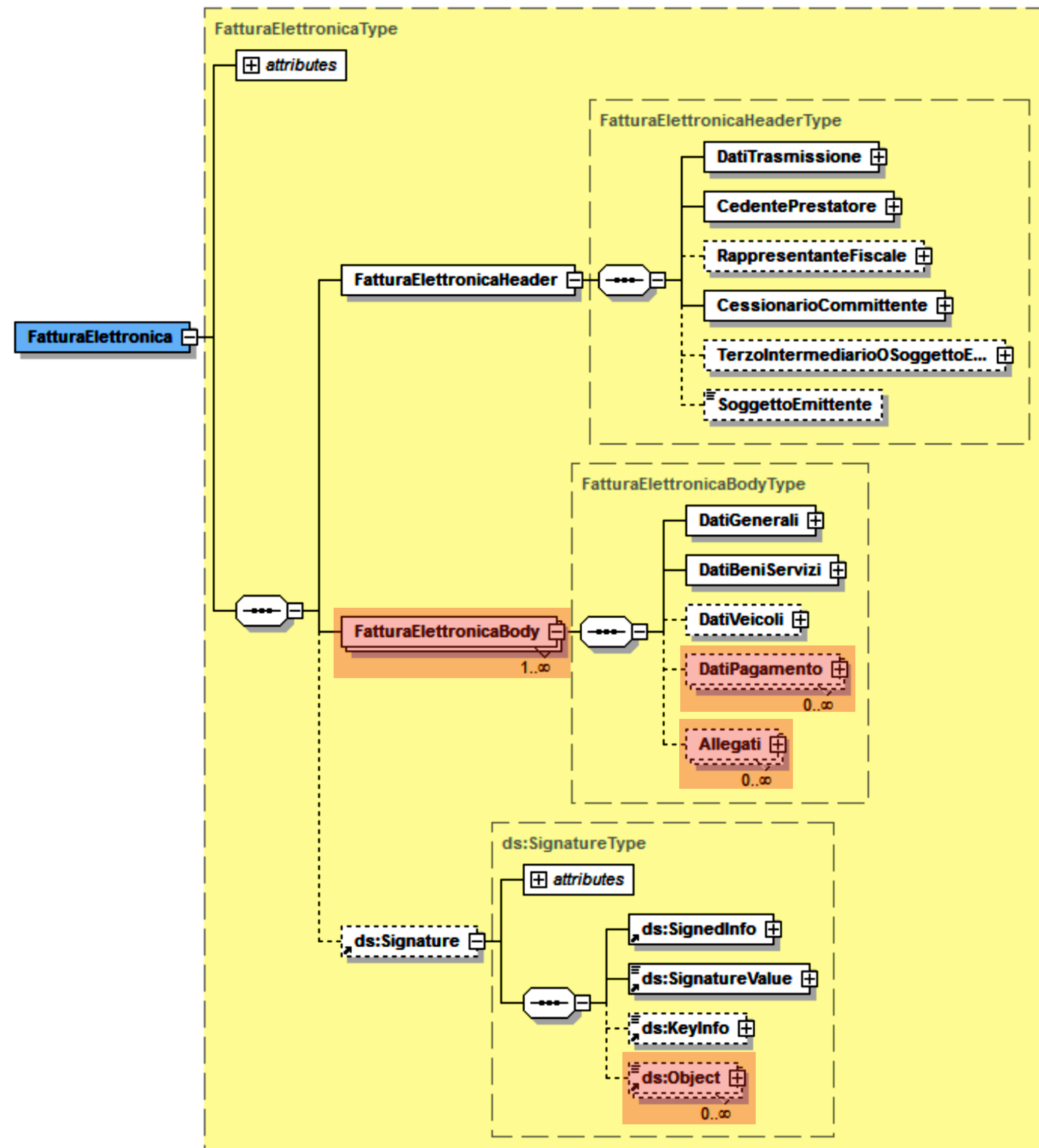


# Fattura Elettronica

l'oggetto

1 - Proprietà **semplici** (*string, integer, decimal, date, time*)

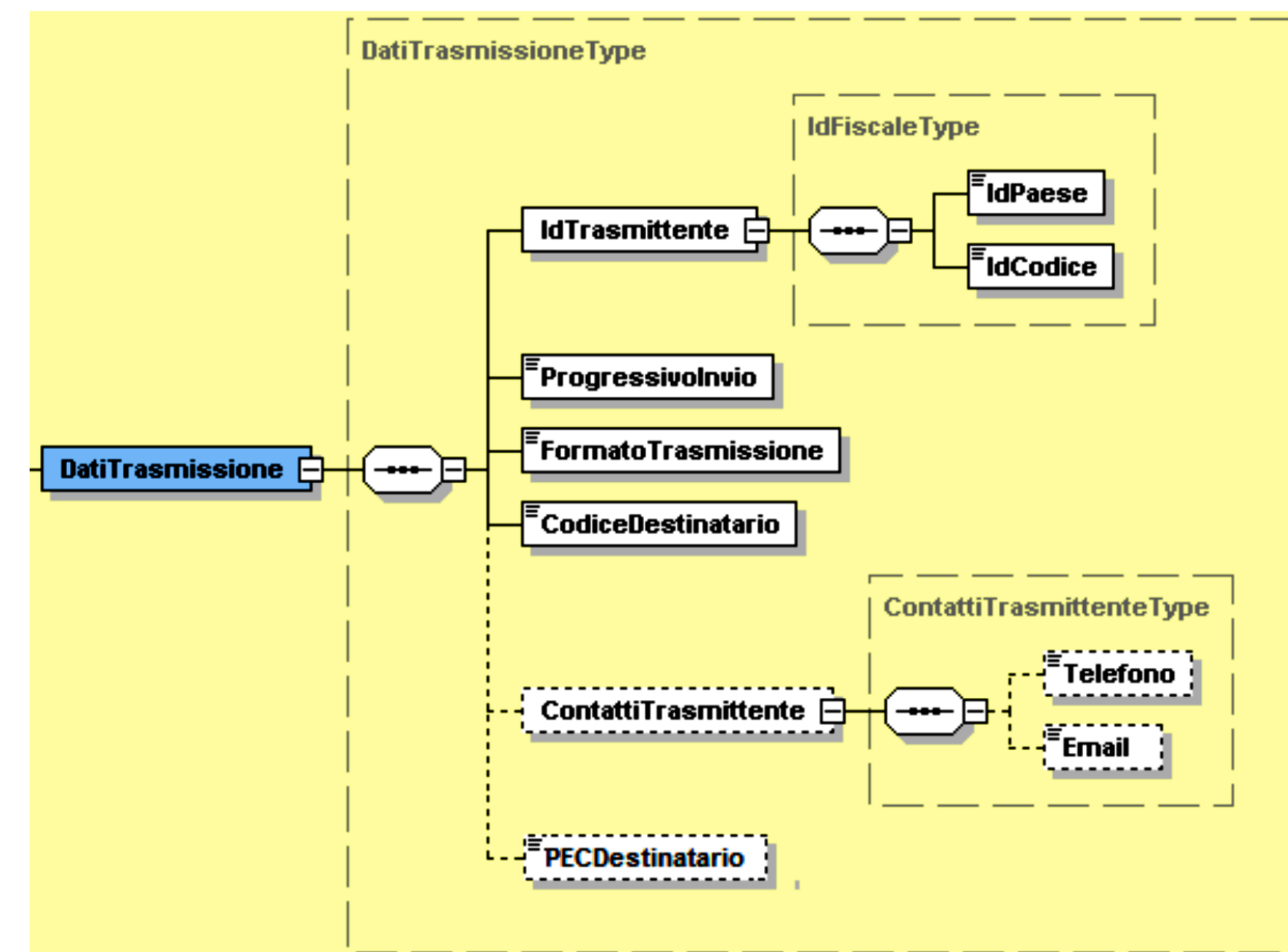
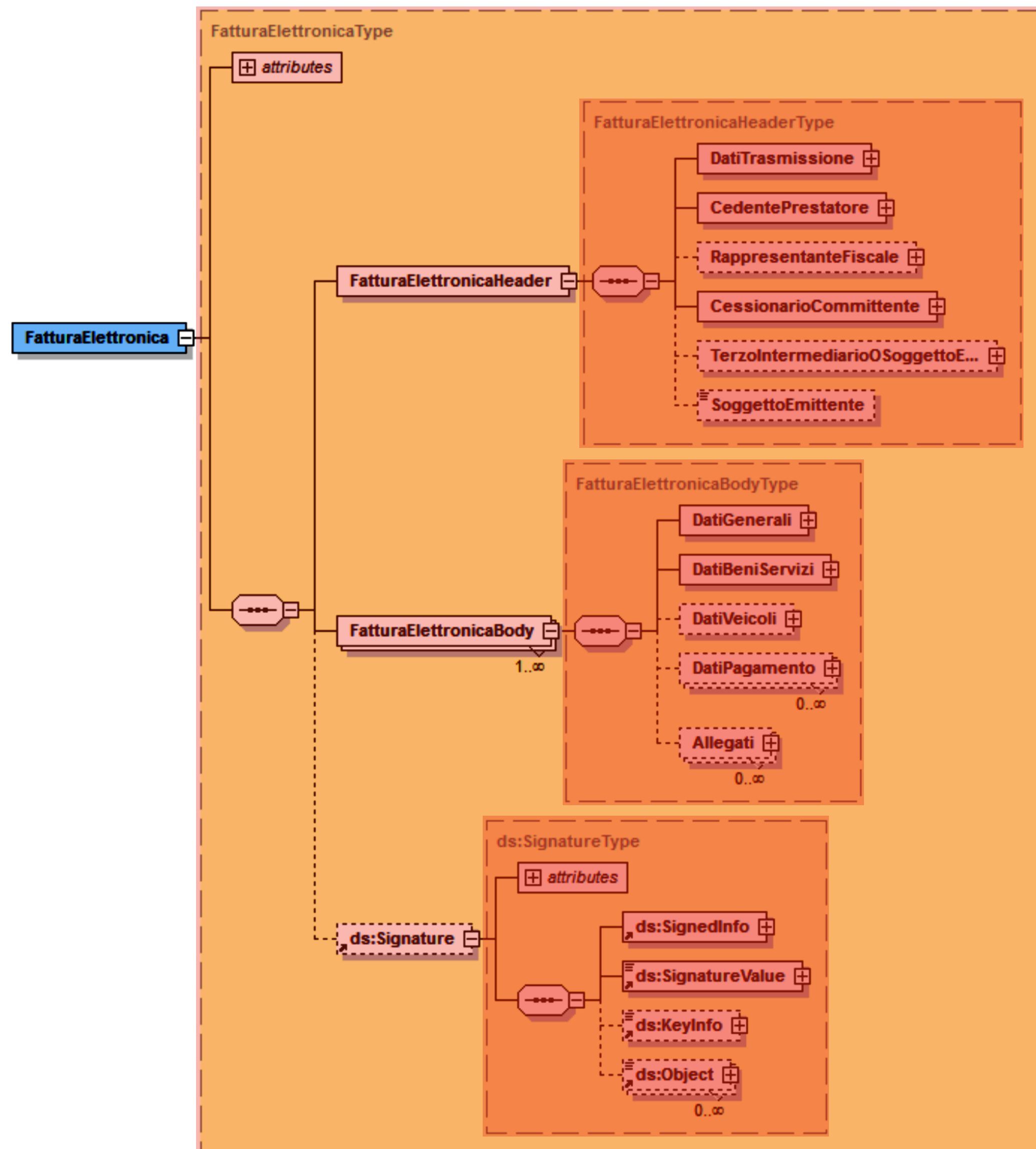
2 - Liste



# Fattura Elettronica

l'oggetto

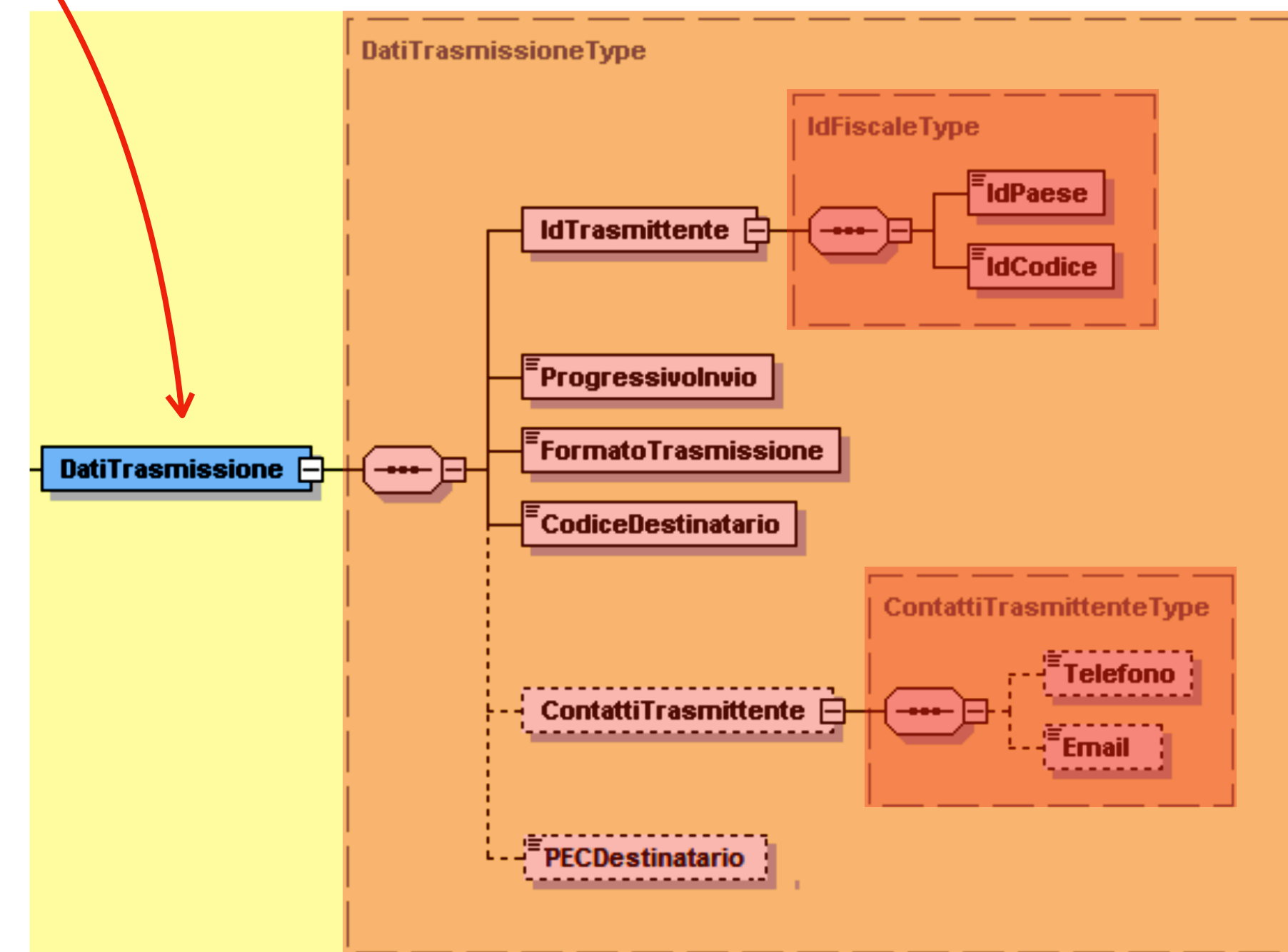
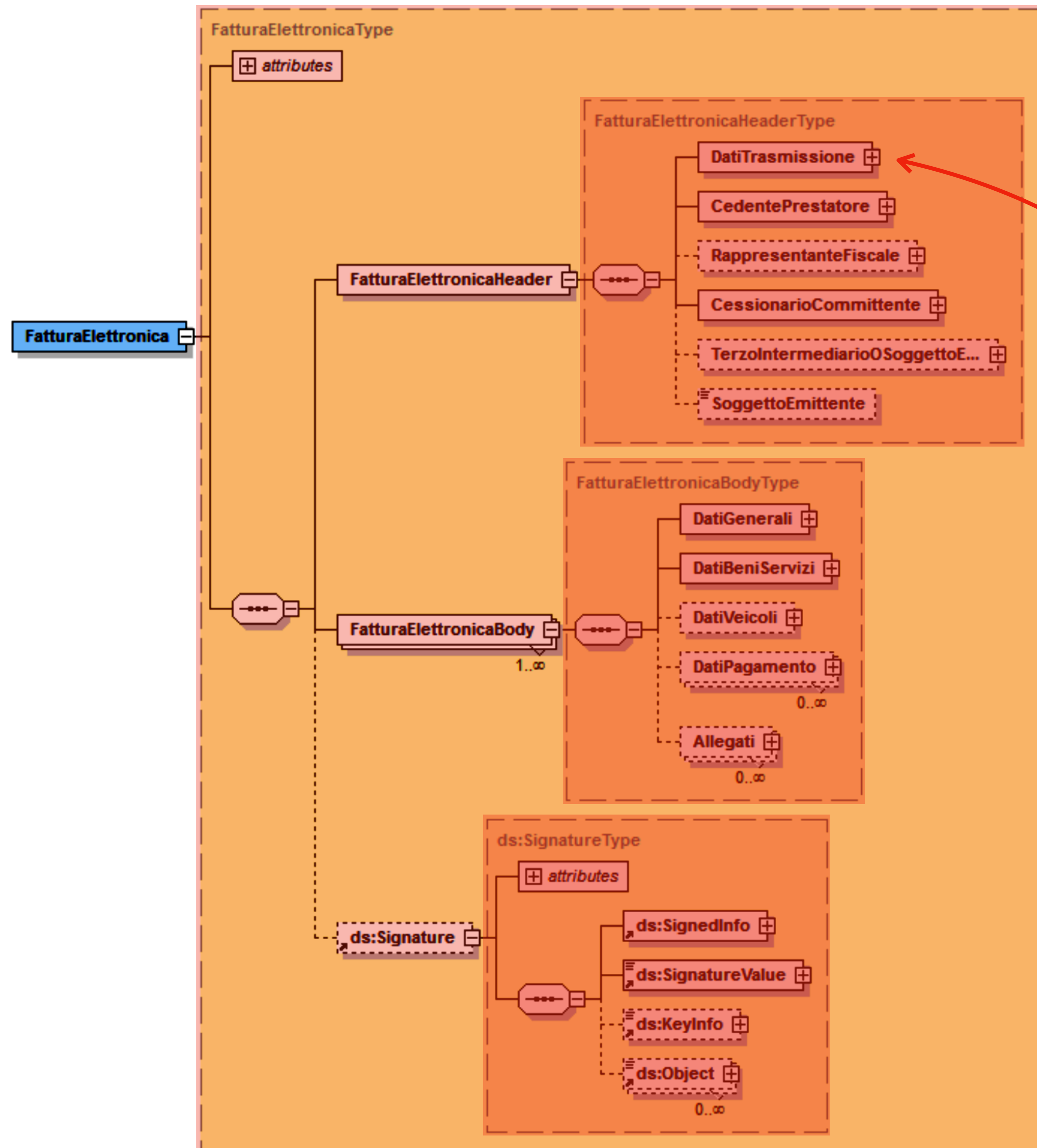
- 1 - Proprietà **semplici** (*string, integer, decimal, date, time*)
- 2 - Liste
- 3 - Blocchi



# Fattura Elettronica

l'oggetto

- 1 - Proprietà **semplici** (*string, integer, decimal, date, time*)
- 2 - **Liste**
- 3 - **Blocchi**



# Demotime

... alla ricerca dell'implementazione misteriosa



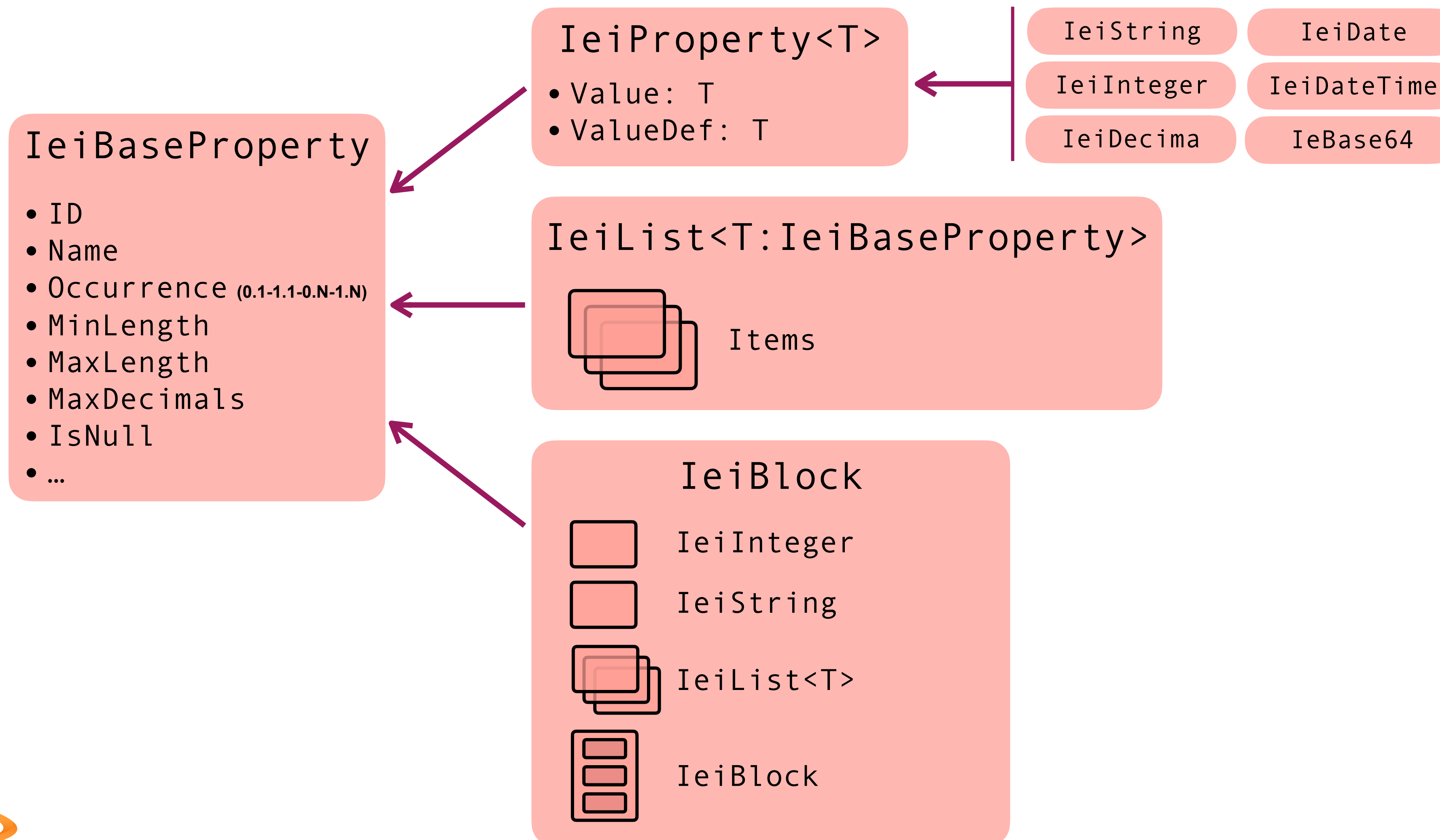


# eInvoice4D

## Basics

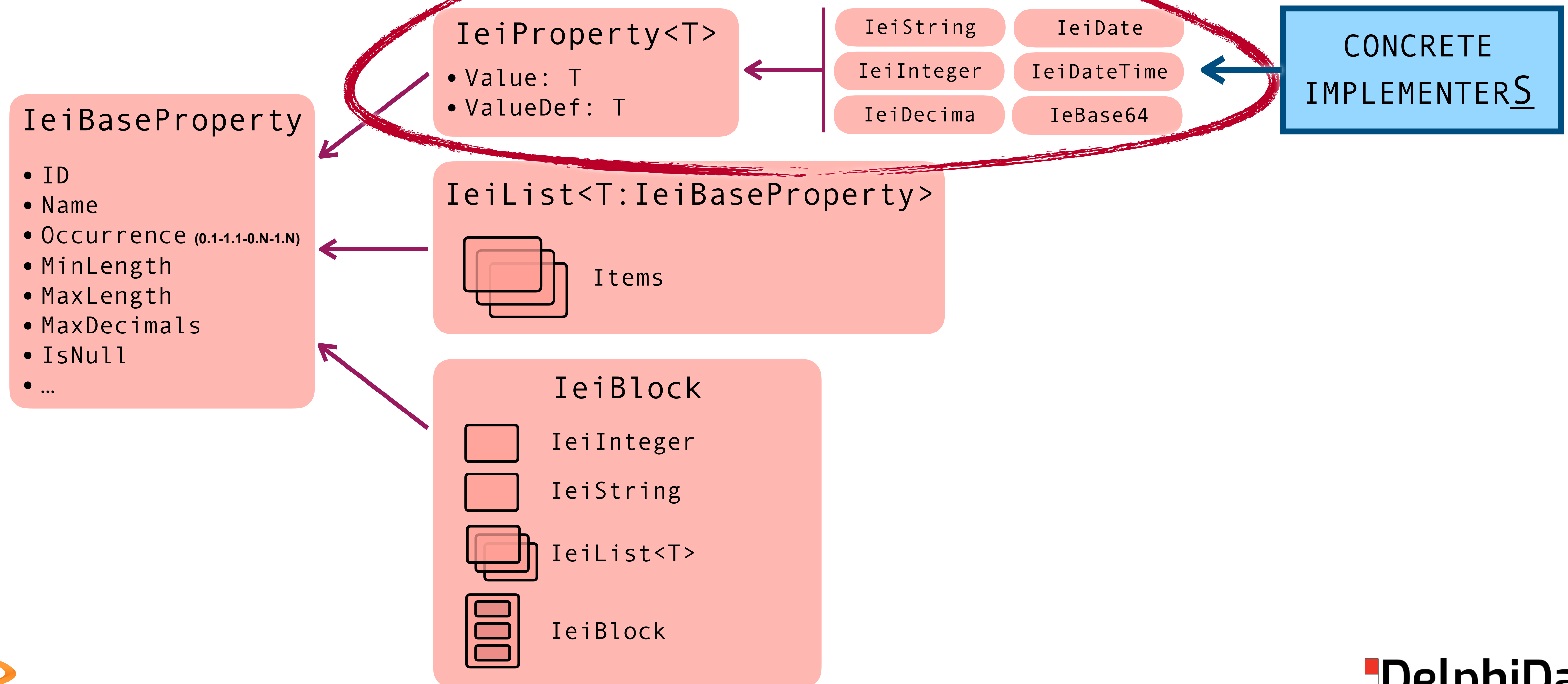
# eInvoice4D

## Basics



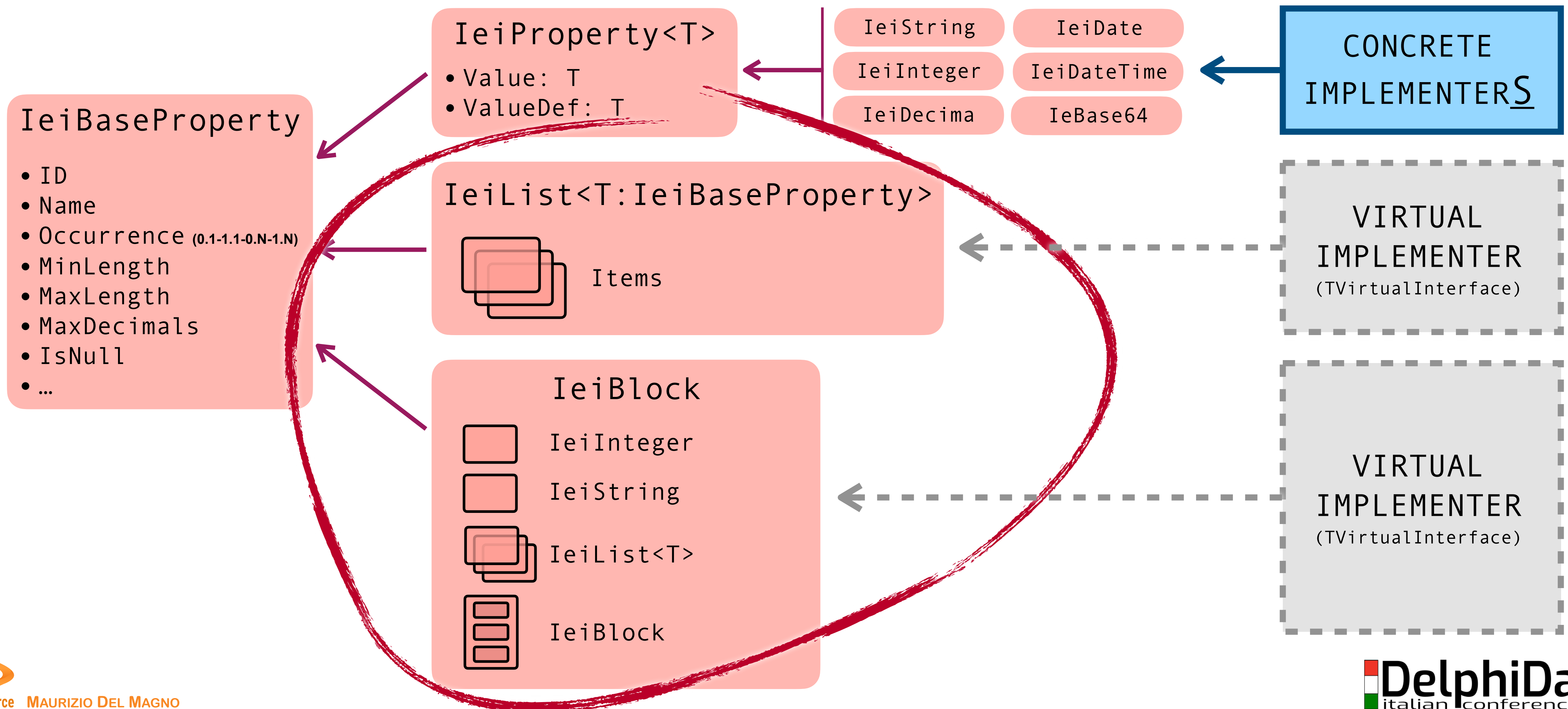
# eInvoice4D

## Basics



# eInvoice4D

## Basics





# StaticImplementers

# StaticImplementers





# StaticImplementers



# StaticImplementers





# StaticImplementers



# TVirtualInterface

...a magic class





# TVirtualInterface

...a magic class



# TVirtualInterface

...a magic class





# TVirtualInterface

...a magic class





# TVirtualInterface

...a magic class





# TVirtualInterface

...a magic class

```
TMyImpl = class(TVirtualInterface)
```

```
var  
  MyImpl: TMyImpl;  
begin  
  MyImpl := TMyImpl.Create(TypeInfo(IMyATM));  
  MyImpl.OnInvoke := DoInvoke;  
end;
```



← Empty!!!

# TVirtualInterface

...a magic class

```
TMyImpl = class(TVirtualInterface)
```

```
var  
  MyImpl: TMyImpl;  
begin  
  MyImpl := TMyImpl.Create(TypeInfo(IMyATM));  
  MyImpl.OnInvoke := DoInvoke;  
end;
```



```
procedure TMyImpl.DoInvoke(const Method: TRttiMethod; const Args: TArray<TValue> out Result: TValue);  
begin  
  ...  
end;
```



# TVirtualInterface

...a magic class

```
TMyImpl = class(TVirtualInterface)
```

```
var  
    MyImpl: TMyImpl;  
begin  
    MyImpl := TMyImpl.Create(TypeInfo(IMyATM));  
    MyImpl.OnInvoke := DoInvoke;  
end;
```



```
procedure TMyImpl.DoInvoke(Method: TRttiMethod; const Args: TArray<TValue>; out Result: TValue);  
begin  
    ...  
end;
```

# TVirtualInterface

...a magic class



# Fattura Elettronica

l'oggetto

# Fattura Elettronica

l'oggetto

```
IFatturaElettronicaType = interface(IeiBlock)
    ['{FD813671-4E2F-44BA-827B-F2F15FADD5E4}']
    [eiBlock(1, o11)]
    function FatturaElettronicaHeader: IFatturaElettronicaHeaderType;
    [eiList(2, o1N)]
    function FatturaElettronicaBody: IeiList<IFatturaElettronicaBodyType>;
end;
```



# Fattura Elettronica

## l'oggetto

```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));  
MyFatturaElettronica.OnInvoke := DoInvoke;
```

TeiBlockProperty = class(TVirtualInterface, ...)

### Virtual Methods table

1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

**Props:** TDictionary<string, leiBaseProperty>;

Name

Instance

'FatturaElettronicaHeader'



'FatturaElettronicaBody'



procedure **DoInvoke**(**Method**: TRttiMethod; const **Args**: TArray<TValue>; out **Result**: TValue);

**Method**: TRttiMethod

**Args**: TArray<TValue>

**Result**: TValue

# Fattura Elettronica

## l'oggetto

```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));  
MyFatturaElettronica.OnInvoke := DoInvoke;
```

```
var  
  MyInvoice: IFatturaElettronicaType;  
begin  
  MyInvoice := ei.NewInvoice;  
  MyInvoice.FatturaEettronicaHeader...  
  ...  
end;
```

TeiBlockProperty = class(TVirtualInterface, ...)

### Virtual Methods table

1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

TRttiMethod

Props: TDictionary<string, leiBaseProperty>;

Name

Instance

'FatturaElettronicaHeader'



'FatturaElettronicaBody'



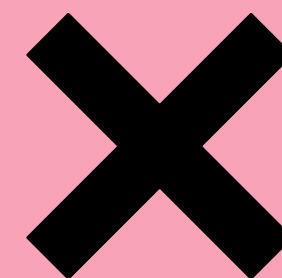
procedure **DoInvoke**(Method: TRttiMethod; const **Args**: TArray<TValue>; out **Result**: TValue);

**Method**: TRttiMethod

**Args**: TArray<TValue>

**Result**: TValue

**Name** = 'FatturaElettronicaHeader'



# Fattura Elettronica

## l'oggetto

```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));  
MyFatturaElettronica.OnInvoke := DoInvoke;
```

TeiBlockProperty = class(TVirtualInterface, ...)

### Virtual Methods table

1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

Props: TDictionary<string, leiBaseProperty>;

Name

Instance

'FatturaElettronicaHeader'



'FatturaElettronicaBody'



procedure **DoInvoke**(Method: TRttiMethod; const **Args**: TArray<TValue>; out **Result**: TValue);

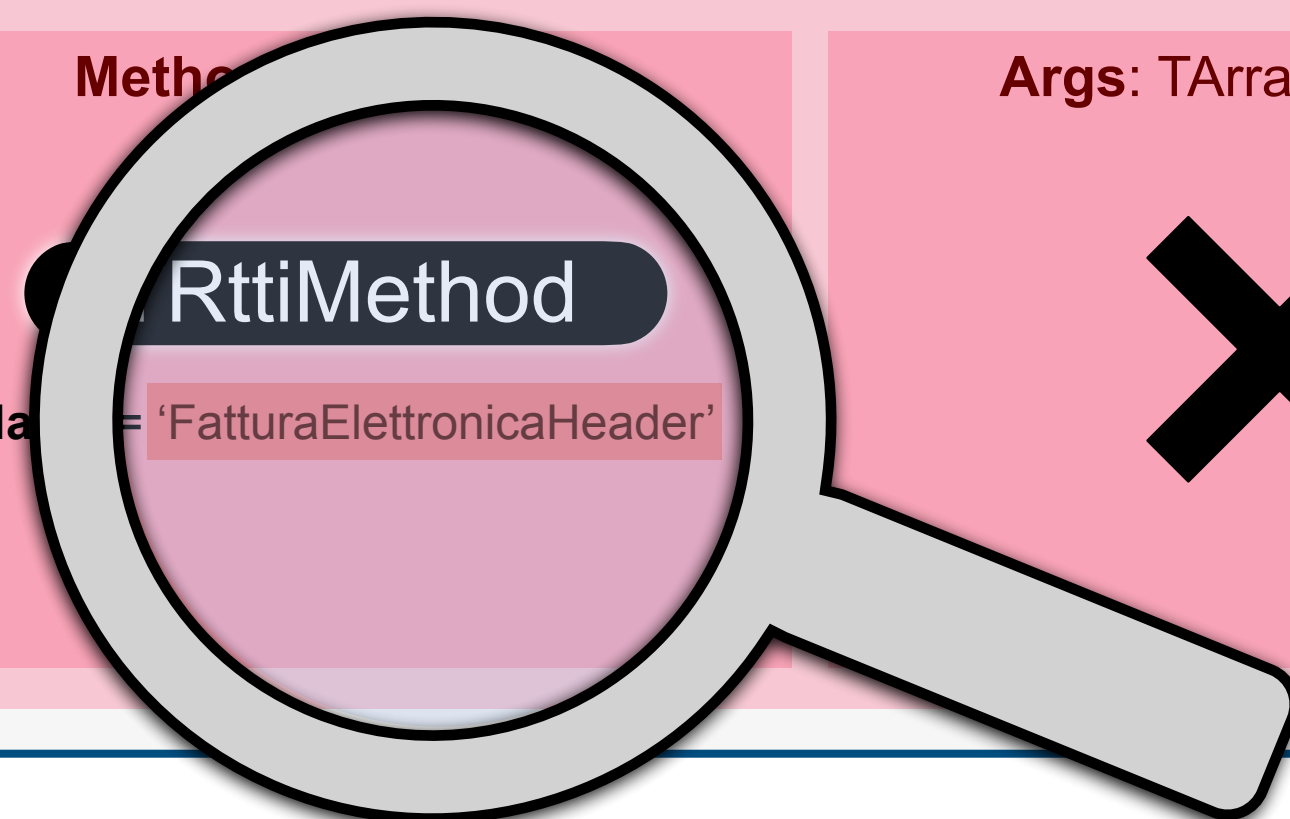
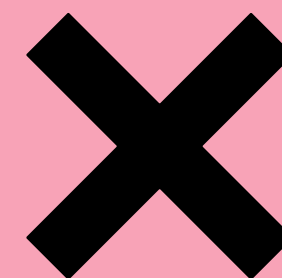
Method

Args: TArray<TValue>

Result: TValue

RttiMethod

Name = 'FatturaElettronicaHeader'





# Fattura Elettronica

## l'oggetto

```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));  
MyFatturaElettronica.OnInvoke := DoInvoke;
```



```
var  
  MyInvoice: IFatturaElettronicaType;  
begin  
  MyInvoice := ei.NewInvoice;  
  MyInvoice.FatturaEettronicaHeader...  
  ...  
end;
```

TeiBlockProperty = class(TVirtualInterface, ...)

Virtual Methods table

- 1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
- 2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

Props: TDictionary<string, leiBaseProperty>;

Name	Instance
'FatturaElettronicaHeader'	
'FatturaElettronicaBody'	

procedure **DoInvoke**(Method: TRttiMethod; const Args: TArray<TValue>; out Result: TValue);

Method: TRttiMethod

TRttiMethod

Name = 'FatturaElettronicaHeader'

Args: TArray<TValue>

X

Result: TValue

TValue



# Fattura Elettronica

## l'oggetto

```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));  
MyFatturaElettronica.OnInvoke := DoInvoke;
```

TeiBlockProperty = class(TVirtualInterface, ...)

### Virtual Methods table

1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

**Props:** TDictionary<string, leiBaseProperty>;

Name

Instance

'FatturaElettronicaHeader'



'FatturaElettronicaBody'



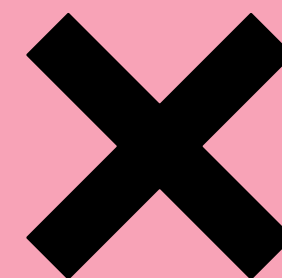
procedure **DoInvoke**(Method: TRttiMethod; const Args: TArray<TValue>; out Result: TValue);

**Method:** TRttiMethod

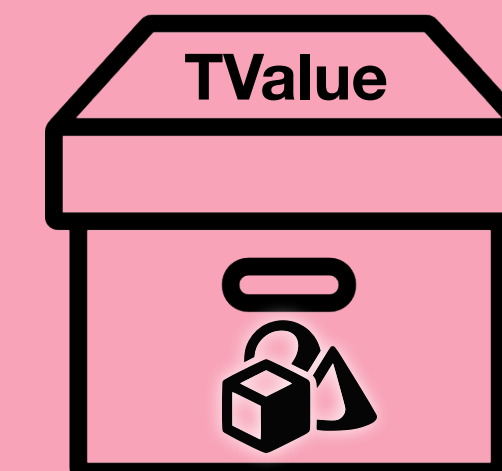
TRttiMethod

Name = 'FatturaElettronicaHeader'

**Args:** TArray<TValue>



**Result:** TValue



# Fattura Elettronica

## l'oggetto

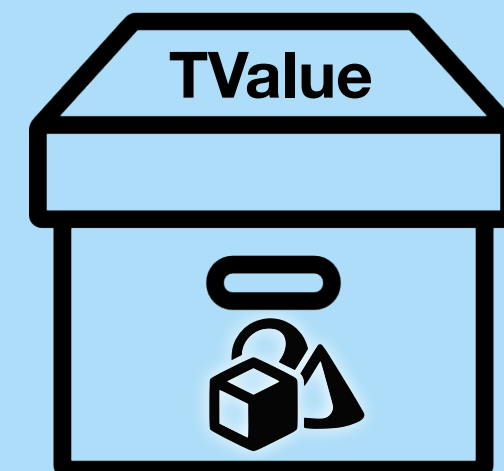
```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));  
MyFatturaElettronica.OnInvoke := DoInvoke;
```

```
var  
  MyInvoice: IFatturaElettronicaType;  
begin  
  MyInvoice := ei.NewInvoice;  
  MyInvoice.FatturaEettronicaHeader...  
  ...  
end;
```

TeiBlockProperty = class(TVirtualInterface, ...)

### Virtual Methods table

1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

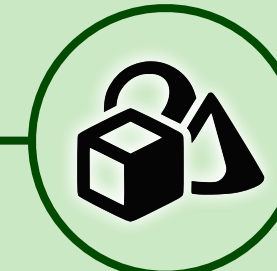


**Props:** TDictionary<string, leiBaseProperty>;

Name

Instance

'FatturaElettronicaHeader'



'FatturaElettronicaBody'



procedure **DoInvoke**(Method: TRttiMethod; const **Args**: TArray<TValue>; out **Result**: TValue);

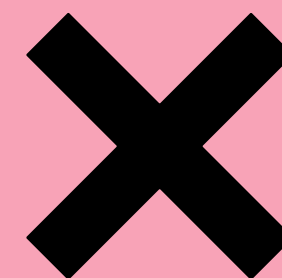
**Method:** TRttiMethod

**Args:** TArray<TValue>

**Result:** TValue

TRttiMethod

Name = 'FatturaElettronicaHeader'





# Fattura Elettronica

## l'oggetto

```
MyFatturaElettronica := TeiBlockProperty.Create(TypeInfo(IFatturaElettronicaType));
MyFatturaElettronica.OnInvoke := DoInvoke;
```

```
var
  MyInvoice: IFatturaElettronicaType;
begin
  MyInvoice := ei.NewInvoice;
  MyInvoice.FatturaEettronicaHeader...
  ...
end;
```



TeiBlockProperty = class(TVirtualInterface, ...)

### Virtual Methods table

- 1. function **FatturaElettronicaHeader**: IFatturaElettronicaHeaderType
- 2. function **FatturaElettronicaBody**: leiList<IFatturaElettronicaBodyType>

Props: TDictionary<string, leiBaseProperty>;

Name	Instance
'FatturaElettronicaHeader'	
'FatturaElettronicaBody'	

procedure **DoInvoke**(Method: TRttiMethod; const **Args**: TArray<TValue>; out **Result**: TValue);

Method: TRttiMethod	Args: TArray<TValue>	Result: TValue
<div>TRttiMethod</div> <div>Name = 'FatturaElettronicaHeader'</div>		

# Fattura Elettronica

l'oggetto

```
MyFatturaElettronicaBodyType = TPropertyCreator<TFatturaElettronicaBodyType, TFatturaElettronicaBodyType>((TFatturaElettronicaBodyType));  
FatturaElettronicaBodyType.OnInvoke := DoInvoke;
```

TVirtualInterface = class(TVirtualInterface, ...)

Virtual Methods table

Props: TList<TFatturaElettronicaHeader>;

Name	Instance
'FatturaElettronicaHeader'	
'FatturaElettronicaBody'	

procedure DoInvoke(Method: TRttiMethod; const Args: TArray<TValue>; out Result: TValue);

Method: TRttiMethod	Args: TArray<TValue>	Result: TValue
---------------------	----------------------	----------------

# Ringraziamenti

# Ringraziamenti



- Nuova classe provider **TeiProviderOSItalia** *(David Lastrucci)*
- Servizi fatturazione elettronica con **API REST 100% Delphi**
- DelphiDay **Gold Sponsor**



# Demo**time**



# Domande?





## MAURIZIO DEL MAGNO DEVELOPER



**Lev@nte software**



**I-ORM**

[github.com/mauriziodm/iORM](https://github.com/mauriziodm/iORM)

**DJSON**

[github.com/mauriziodm/DJSON](https://github.com/mauriziodm/DJSON)



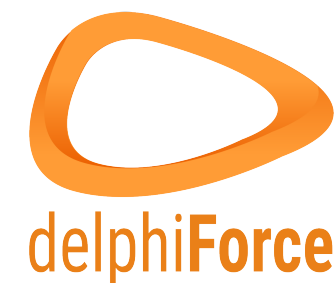
[mauriziodm@levantesw.it](mailto:mauriziodm@levantesw.it)

[mauriziodelmagno@gmail.com](mailto:mauriziodelmagno@gmail.com)



[facebook.com/maurizio.delmagno](https://facebook.com/maurizio.delmagno)

IORM + DJSON (GROUP)



Membro fondatore

**eInvoice4D**

<https://github.com/delphiforce/eInvoice4D>



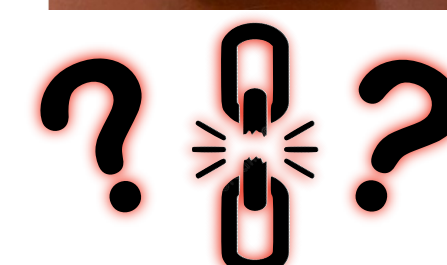
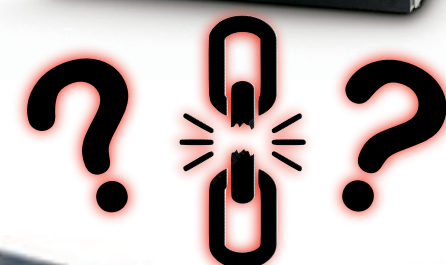
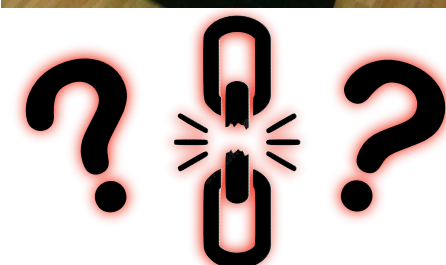
# Grazie



**SPEAKER: MAURIZIO DEL MAGNO**



# StaticImplementers





**OLTRE AL REFACTORING DI INVOICE4D VORREI ANNUNCIARE LA DISPONIBILITÀ DI UN NUOVO PROVIDER OLTRE A QUELLO ARUBA CHE ERA GIÀ PRESENTE.**

**A TALE SCOPO RINGRAZIO DAVID LASTRUCCI DI OPEN SOURCE ITALIA, CHE TRA L'ALTRO È SPONSOR DELL'EVENTO, PER AVERCI FORNITO L'IMPLEMENTAZIONE PER POTER UTILIZZARE I LORO SERVIZI DI FATTURAZIONE ELETTRONICA TRAMITE LA NOSTRA LIBRERIA.**

**LE LORO API REST SONO 100% DELPHI E, OPEN SOURCE ITALIA È, QUI FUORI, A DISPOSIZIONE PER CHIUNQUE DI VOI VOGLIA APPROFONDIRE L'ARGOMENTO FATTURAZIONE ELETTRONICA.**