Web service description Language (WSDL)

In Web service there is multiple messages exchanging format, each have specific role and structure.

1. **Rpc Encoded**
2. **Document Literal**
3. **Document Wrapped**

Rpc Encoded (Remote Procedural call)

Let’s consider one example to make Rpc encoded base WSDL as below.

Interface **Railway** {

Public Ticket bookTicket(PassengerInfo pInfo,JourneyInfo jInfo);

}

# WSDL:

<?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>

<wsdl:definitions xmlns:rs=*"http://irctc.com/online/reservation/service"*

xmlns:soap=*"http://schemas.xmlsoap.org/wsdl/soap/"* xmlns:wsdl=*"http://schemas.xmlsoap.org/wsdl/"*

xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"* name=*"railway"*

targetNamespace=*"http://irctc.com/online/reservation/service"*

xmlns:rt=*"http://irctc.com/online/reservation/types"*>

<wsdl:types>

<xsd:schema targetNamespace=*"http://irctc.com/online/reservation/types"*>

<xsd:complexType name=*"PassengerInfo"*>

<xsd:sequence>

<xsd:element name=*"name"* type=*"xsd:string"* />

<xsd:element name=*"age"* type=*"xsd:int"* />

<xsd:element name=*"gender"* type=*"xsd:string"* />

<xsd:element name=*"mobNo"* type=*"xsd:string"* />

</xsd:sequence>

</xsd:complexType>

<xsd:complexType name=*"JourneyInfo"*>

<xsd:sequence>

<xsd:element name=*"source"* type=*"xsd:string"* />

<xsd:element name=*"dest"* type=*"xsd:string"* />

<xsd:element name=*"jDate"* type=*"xsd:date"* />

<xsd:element name=*"seatNo"* type=*"xsd:string"*/>

</xsd:sequence>

</xsd:complexType>

<xsd:complexType name=*"Ticket"*>

<xsd:sequence>

<xsd:element name=*"source"* type=*"xsd:string"* />

<xsd:element name=*"dest"* type=*"xsd:string"* />

<xsd:element name=*"price"* type=*"xsd:double"* />

</xsd:sequence>

</xsd:complexType>

</xsd:schema>

</wsdl:types>

<wsdl:message name=*"Railway\_bookTicket"*>

<wsdl:part name=*"JourneyInfo"* type=*"rt:JourneyInfo"* />

<wsdl:part name=*"PassengerInfo"* type=*"rt:PassengerInfo"* />

</wsdl:message>

<wsdl:message name=*"Railway\_bookTicketResponse"*>

<wsdl:part name=*"Ticket"* type=*"rt:Ticket"* />

</wsdl:message>

<wsdl:portType name=*"Railway"*>

<wsdl:operation name=*"bookTicket"*>

<wsdl:input message=*"rs:Railway\_bookTicket"* />

<wsdl:output message=*"rs:Railway\_bookTicketResponse"* />

</wsdl:operation>

</wsdl:portType>



## Here incoming soap request follow the below structure:



## Advantages:

1. The WSDL is self-explanatory.
2. The method name act as a root element in soap request which can be easily understandable by Receiver

## Dis-Advantages:

1. The xsi:type encoding specify in soap request message which is extra overhead

2. It is not WS-I complaint.

Document Literal

interface **Railreservation**{

public ticket bookTicket(passengerInfo pinfo,JOyrneyinfo jinfo);

}

# WSDL:

<defination name="RailReservationinfo" targetNamespace="http://ticket.org/service"

xmlns:rrs="http://ticket.org/service" xmlns:rrt="http://ticket.org/types">

<wsdl:types>

<xs:schema targetnameSpace="http://ticket.org/types">

<xs:element name="journeyInfo">

<xs:complexType>

<xs:sequence>

<xs:element name="destination" type="xs:string" />

<xs:element name="source" type="xs:string" />

<xs:element name="journeyDate" type="xs:string" />

</xs:sequence>

</xs:complexType>

<xs:element name="PassengerInfo">

<xs:complexType >

<xs:sequence>

<xs:element name="age" type="xs:int" />

<xs:element name="passengerName" type="xs:string" />

<xs:element name="uniqId" type="xs:string" />

<xs:element name="gender" type="xs:string" />

</xs:sequence>

</xs:complexType>

<xs:element name="Ticket">

<xs:complexType>

<xs:sequence>

<xs:element name="pnrno" type="xs:string" />

<xs:element name="berth" type="xs:string" />

<xs:element name="ticketNo" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:schema>

<wsdl:types>

<wsdl:message name="RailReservation\_bookTicket">

<wsdl:part name="pinfo" element="rrt:PassengerInfo"/>

<wsdl:part name="jinfo" element="rrt:journeyInfo"/>

<wsdl:message>

<wsdl:message name="RailReservation\_bookTicketResponse">

<wsdl:part name="result" element="rrt:Ticket">

<wsdl:message>

<wsdl:portTYpe name="RailReservation">

<operation name="bookTicket">

<input message="rrs:RailReservation\_bookTicket">

<output message="rrs:RailReservation\_bookTicketresponse">

</operation>

</wsdl:portTYpe>

<wsdl:binding name="RailReservationSOAPBinding" type="rrs:RailReservation">

<soap:binding:transport="http" style="document">

<operation name="bookTicket">

<soap:operation:action="http://ticket.org/service\_bookTicket">

<input>

<soap:body use="literal" type="http://ticket.org/service">

</input>

<output>

<soap:body use="literal" type="http://ticket.org/service">

</output>

</operation>

</wsdl:binding>

<wsdl:service name="RailReservationService">

<port name="RailReservationPort" binding="rrs:RailReservationSOAPBinding">

<soap:address:location="http://localhost:4040/RailReservation/url"

</wsdl:service>

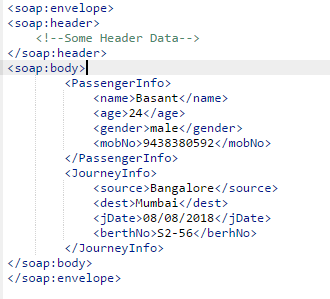
<defination>

## What is the change in Document-Literal MEF in compare with Rpc-Encoded:-

1:- in Document literal type section we are not taking complexType coz here we don’t want to send type of data, here we use Element instead of complexType coz here we want to send what are data required to my service method that’s why we directly send data

2:- in message section we use part name which is not affect here u can take any name and here in part it doesn't contain type coz here we don’t want to send request as a type of data

Here instead of type we have to use element as attribute coz we are specify here which element i want to send as input



# Dis-advantages:

See in above request soap xml here no root element mention in <body> which is became complicated to identify and it contain 2 child element ,means it indirectly saying to developer without method parameter u can't use this MEF approach Means u should have to take at least one parameter in provider web service method.....

interface RailReservation{

public Ticket bookTicket(); 🡪it is not possible coz no parameter, then in soap request xml what should be the child element of <body> ?,nothing that’s why it is not recommended

}

Document Wrapped:

Public interface SBI\_Loan{

Public Acknowledgement applyLoan(AccountInfo accInfo,TransactionInfo txInfo);

}

# WSDL:

<?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>

<wsdl:definitions xmlns:sbis=*"http://www.sbi.com/apply/loan/service"*

xmlns:soap=*"http://schemas.xmlsoap.org/wsdl/soap/"* xmlns:wsdl=*"http://schemas.xmlsoap.org/wsdl/"*

xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"* name=*"SBI\_Loan"*

targetNamespace=*"http://www.sbi.com/apply/loan/service"* xmlns:sbit=*"http://www.sbi.com/apply/loan/types"*>

<wsdl:types>

<xsd:schema targetNamespace=*"http://www.sbi.com/apply/loan/types"*>

<xsd:element name=*"AccountInfo "*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"name"*type=*"xsd:string"*/>

<xsd:element name=*"accNo* type=*"xsd:string"*/>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name=*"LoanInfo "*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"loanMode"*type=*"xsd:string"*/>

<xsd:element name=*"amount"* type=*"xsd:double"*/>

<xsd:element name=*"tenure"* type=*"xsd:string"*/>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name=*"Acknowledgement "*>

<xsd:complexType>

<xsd:sequence>

<xsd:element name=*"loanMode"* type=*"xsd:string"*/>

<xsd:element name=*"amount"* type=*"xsd:double"* />

<xsd:element name=*"status"* type=*"xsd:string"* />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

Here I do extra step means I bind my required request data in to one element that is my method name coz to get method call structure in my request soap xml

<xsd:element name=*"applyLoan"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element ref=*"sbit:AccountInfo"* />

<xsd:element ref=*"sbit:LoanInfo"* />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

Here I do extra step means I bind my required response data in to one element that is my method name coz to get method call structure in my request soap xml

<xsd:element name=*"applyLoanResponce"*>

<xsd:complexType>

<xsd:sequence>

<xsd:element ref=*"sbit:Acknowledgement"*/>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:schema>

</wsdl:types>

<wsdl:message name=*"applyLoan"*>

<wsdl:part element=*"sbit:applyLoan"* name=*"input"* />

</wsdl:message>

<wsdl:message name=*"applyLoanResponse"*>

<wsdl:part element=*"sbit:applyLoanResponce"* name=*"out"* />

</wsdl:message>

<wsdl:portType name=*"SBI\_Loan"*>

<wsdl:operation name=*"applyLoan"*>

<wsdl:input message=*"sbis:applyLoan"* />

<wsdl:output message=*"sbis:applyLoanResponse"* />

</wsdl:operation>

</wsdl:portType>



# Incoming soap request structure:



## What is the change in Document-Wrapped MEF in compare with Rpc-Encoded:-

See in above type section i add two extra element one is for input and one for output coz ,if i directly send element as per Document-literal then in my request soap xml there is no root element in <soap:body> section .so to overcome this problem i take two element exactly following my method name ,applyLoan is act as complete input Request coz i bind/wrapped my AccountInfo,and LoanInfo in single element which is applyLoan

applyLoanResponse whic is act as complete response

**Note:-**

This is not provided by JAXWS thats why in binding section till we have to mention style="Document" and use="literal”, cause this is given by Microsoft. And maximum implementation vender support it