Uniface User Conference Face2face

Enhancing Uniface Web services

Berry Kuijer Saat

May 23, 2012

Agenda

- Transformation
- The Struct
- Applying Struct
- SOAP complex parameter support
- SOAP Header support
- SOAP Fault support

Compuware UNIFACE® Development for the Cloud

Challenge

With the various formats used for Web Services we needed a solution that allowed us to interface with and transform to and from the objects and constructs used within Uniface.

Plus it had to be fast and extensible

Compuware UNIFACE® Development for the Cloud

Challenges of formats

Entity		↘	↘	↘	↘	↘
List	↘		↘	↘	↘	↘
XML	↘	↘		↘	↘	↘
HTML	↘	↘	↘		↘	↘
JSON	↘	↘	↘	↘		↘
…	↘	↘	↘	↘	↘	
	Entity	List	XML	HTML	JSON	…

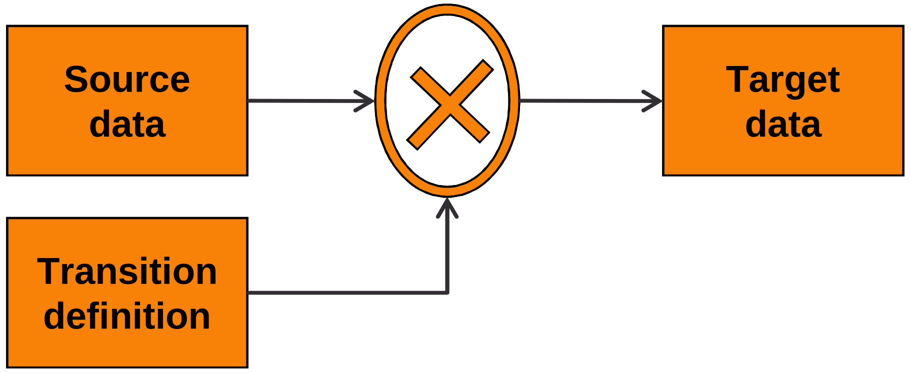
Transformation Challenges

- Naming conflicts
- Name casing (Upper/lowercase)
- Duplicate Names
- Data Type conflicts
- Entity/Occ/Field/Properties v.s. Element/Attributes
- Nesting (complex-types)
- Data Type incompatibilities (boolean T/F 1/0 Y/N)
- Huge Issue: Transformation can result in information loss!!!

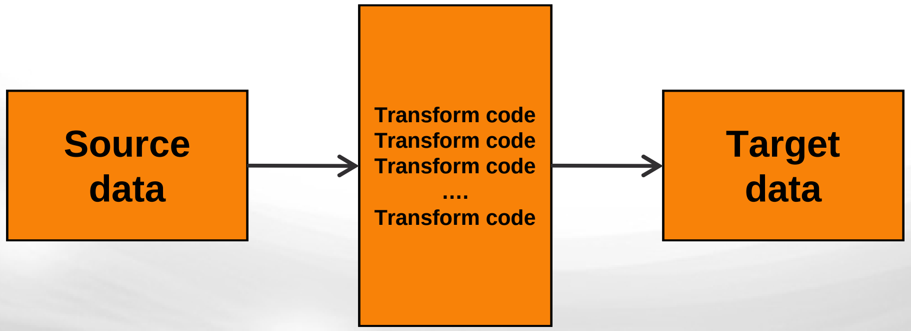
Transformation solutions

Declarative

e.g. XSLT



Procedural

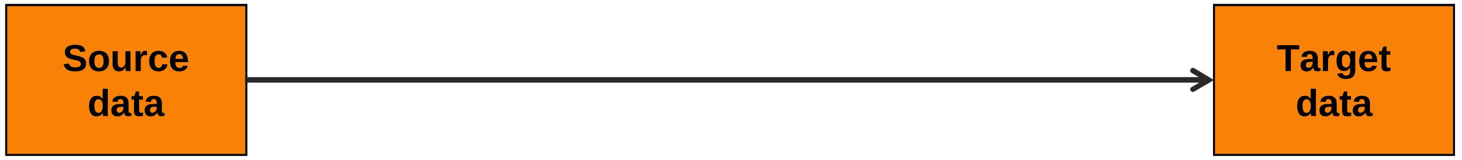


Our Solution

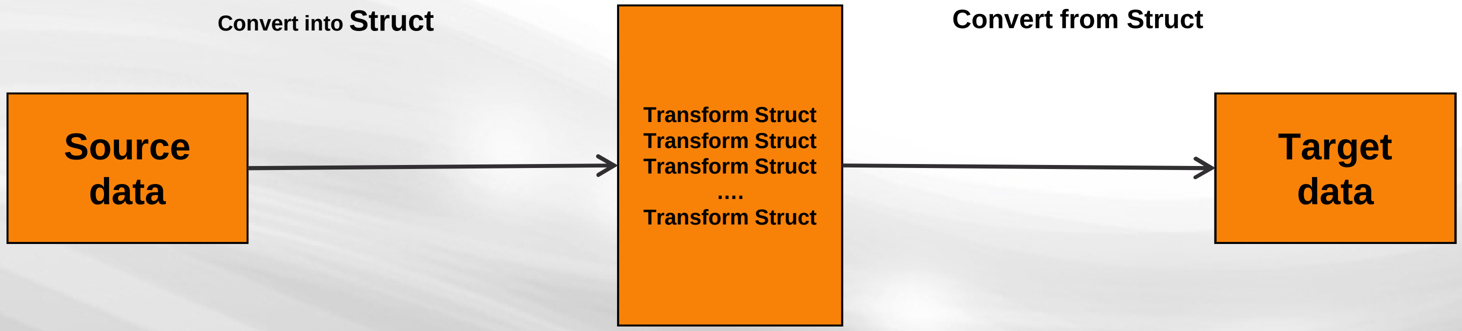
- New Struct data type
- New proc translation instructions

What is Struct – conceptual

Procedural solution to resolve complex data transformation



Regular translation



Retains origin attributes

What is Struct – technical

```
variables
  struct uOrder
endvariables
putmess uOrder->$dbgString
```

- New data type in Proc language: struct
- A struct variable is a reference (handle) to a Structure
- A Structure is a collection of members in memory that are hierarchical organized
- A Structure is created once it is referenced
- A Structure is deleted once it references count is zero
- A Structure has no specification
- A struct variable has no specification
- A Structure is created dynamically (at runtime)
- A member of a Structure is either a node or a leave
- Every member can be directly addressed
- Every member can be created, moved, copied, deleted, renamed using Proc
- Leaves have a value that can be read and set using Proc

```
[MYORDERFORM]
  [ORDER.SALES]
    [occ]
      [ORDER_ID] = "21EC2020"
      [DATE] = 20110801
      [STATUS] = 2
      [SHIP_TO] = "My house"
      [ORDERLINE.SALES]
        [occ]
          [LINE_ID] = "213A3AAB"
          [ITEM_NAME] = "Les Paul"
          [UNIT_PRICE] = 1200
          [QUANTITY] = 1
          [LINE_TOTAL] = 1200
        [occ]
          [LINE_ID] = "1234FFFF"
          [ITEM_NAME] = "PHB Slater"
          [UNIT_PRICE] = 3225
          [QUANTITY] = 2
          [LINE_TOTAL] = 6450
      [TOTAL] = 7650
```

```
quantity = uOrder->MYORDERFORM->ORDER.SALES->occ{1}->ORDERLINE.SALES->occ{2}->QUANTITY
; quantity = 2
```

Compuware UNIFACE® Development for the Cloud

What is Struct – technical

- A Structure can be inspected e.g. in the debugger
- A Structure can take any hierarchical format
- A component structure can be converted into a Structure and vice verse
- An XML stream can be converted into a Structure and vice verse
- When created by conversion, a member maintains information about what it originally was (member type + value type)
- This information can be get and set

```
[ORDER.SALES]
  [$tags]
    [u_type] = "entity"
```

```
<aaa>
  abc
</aaa>
```

```
[aaa] = "abc"
  [$tags]
    [xmlClass] = "element"
    [xmlTypeCategory] = "simple"
    [xmlDataType] = "string"
    [xmlTypeNamespace] = "http://www.w3.org/2001/XMLSchema"
```

Supportive Proc Instructions

XMLTOSTRUCT

STRUCTTOXML

STRUCTTOCOMPONENT

COMPONENTTOSTRUCT

FOR - NEXT

New Struct Functions

$newstruct	-	Creates a struct member
$collSize	-	Get the number of Structs in the collection
$dbgString	-	Get a string that represents the Struct collection.
$index	-	Get or set the index of the Struct collection.
$isLeaf	-	Check whether a Struct is a Struct leaf
$isScalar	-	Check whether a Struct is a scalar Struct.
$istags	-	Check whether the Struct is a $tags Struct for another Struct
$memberCount	-	Get the number of members in a Struct.
$name	-	Get the name of a Struct
$parent	-	Get or set the parent of the Struct.
$tags	-	Get or set annotations for a Struct.

Compuware UNIFACE® Development for the Cloud

Use a Struct in Proc

Move members within its collection

```
variables
  struct uOrder
endvariables
  uOrder->ORDER->OCC{2}->$index = 1
  OUTPUT = uOrder->$dbgstring
end
```

```
[MYORDERFORM]
  [ORDER]
    [OCC]
      [ORDER_ID] = 1
      [DATE] = 20110812
      [SHIP_TO] = "Gerton"
    [OCC]
      [ORDER_ID] = 2
      [DATE] = 20110813
      [SHIP_TO] = "Kees"
```

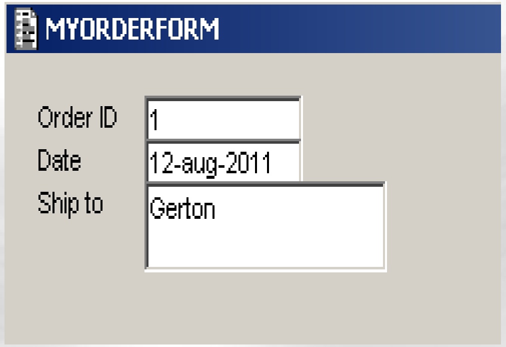
```
[MYORDERFORM]
  [ORDER]
    [OCC]
      [ORDER_ID] = 2
      [DATE] = 20110813
      [SHIP_TO] = "Kees"
    [OCC]
      [ORDER_ID] = 1
      [DATE] = 20110812
      [SHIP_TO] = "Gerton"
```

Use a Struct in Proc

Conversion to component

```
variables
  struct uOrder
endvariables
  uOrder->$name = "<$componentname>"
  uOrder->ORDER = $newstruct
  uOrder->ORDER->OCC = $newstruct
  uOrder->ORDER->OCC->ORDER_ID = 1
  uOrder->ORDER->OCC->DATE = $date
  uOrder->ORDER->OCC->SHIP_TO = "Gerton"
  putmess uOrder->$dbgstring
  structtocomponent uOrder
end
```

```
[MYORDERFORM]
  [ORDER]
    [OCC]
      [ORDER_ID] = 1
      [DATE] = 20110812
      [SHIP_TO] = "Gerton"
```



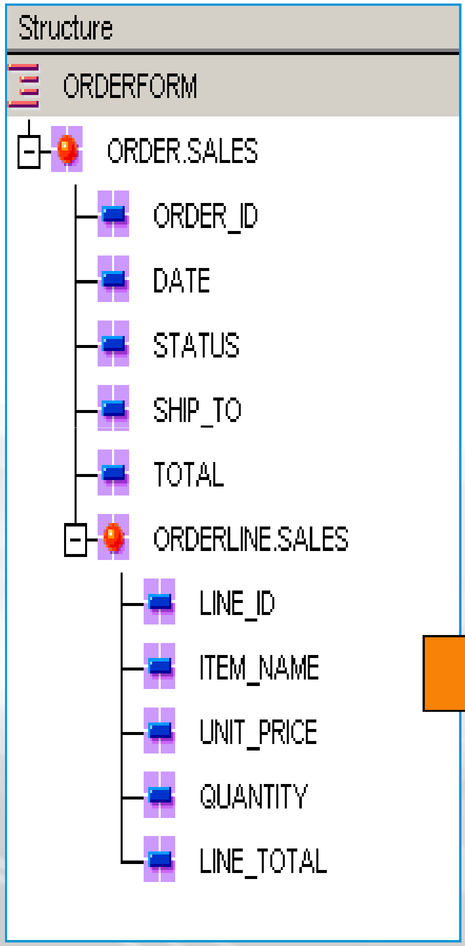
MYORDERFORM

Order ID	1
Date	12-aug-2011
Ship to	Gerton

Compuware UNIFACE Development for the Cloud

Use a Struct in Proc

Conversion from component and meta tags



```
Structure
ORDERFORM
ORDER.SALES
    ORDER_ID
    DATE
    STATUS
    SHIP_TO
    TOTAL
    ORDERLINE.SALES
        LINE_ID
        ITEM_NAME
        UNIT_PRICE
        QUANTITY
        LINE_TOTAL
```

```
variables
  struct uOrder
Endvariables

  retrieve

  componenttostruct uOrder

  putmess uOrder->$dbgstring

end
```

```
[ORDERFORM]
  [$tags]
    [u_type] = "component"
  [ORDER.SALES]
    [$tags]
      [u_type] = "entity"
    [OCC]
      [$tags]
        [u_type] = "occurrence"
      [ORDER_ID] = "1"
        [$tags]
          [u_type] = "field"
      [DATE] = "20110105"
        [$tags]
          [u_type] = "field"
      [STATUS] = "02"
        [$tags]
          [u_type] = "field"
      [SHIP_TO] = "Gerton"
        [$tags]
          [u_type] = "field"
      [TOTAL]
...
```

Compuware UNIFACE Development for the Cloud

Use a Struct in Proc

Conversion from XML and meta tags

```
<?xml version="1.0"?>
<Order>
  <OrderLine
id="21EC2020">
    Gibson Les Paul
  </OrderLine>
  <OrderLine>
    PHB Slater I
  </OrderLine>
</Order>
```

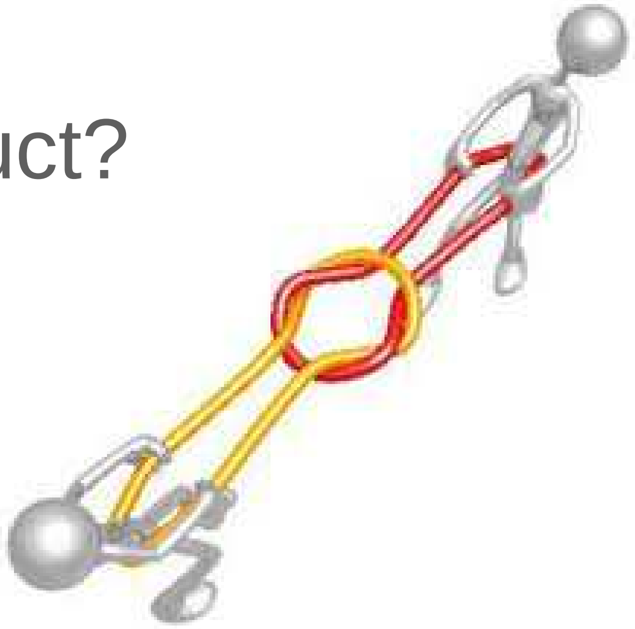
```
variables
  struct uOrder
endvariables
  xmltostruct uOrder, $xml$
  putmess uOrder->$dbgstring

end
```

```
[]
  [$tags]
    [xmlVersion] = 1.0
  [Order]
    [$tags]
      [xmlClass] = element
    [OrderLine]
      [$tags]
        [xmlClass] = element
      [id] = 21EC2020
        [$tags]
          [xmlClass] = attribute
      [] = Gibson Les Paul
    [OrderLine] = PHB Slater I
      [$tags]
        [xmlClass] = element
```

Compuware UNIFACE
Development for the Cloud

Application

Where can we use Struct?

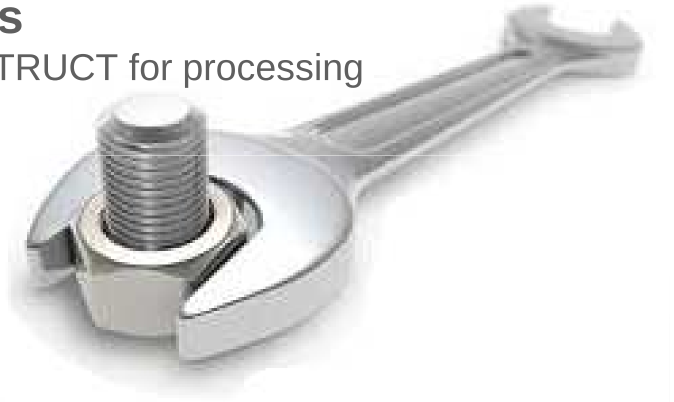


Application examples 9.5

- **Complex parameter support of Web Services**
  complex parameters are created using Structs and then converted into XML to be passed as parameter and vice versa.
  (Struct supports XML schemas and all XML data types)

- **Transformation of SOAP Headers and Messages**
  SOAP headers are made available as XML and converted into STRUCT for processing and/or encryption

- **Splitting and merging entities**



Application in the future

- **Complex data exchange between 4GL component instances**
  the Struct is directly used as a parameter, both for component instances running in the same and in different processes. Serialization is done automatically and on demand.
- **3-Tier communication**
  the 4GL developer already has a 3-tier application and wants to gradually replace xmlSave and xmlLoad statements (including all the DTDs and other overhead) with Struct constructions, where the Struct basically takes over the DTD and mapping administration
- **Exchange of JavaScript objects between browser and server**
  the serialized format of these JavaScript objects is JSON which can be converted into a Struct (on the server) for further processing. JavaScript objects are typically used in DSPs to exchange field properties and valreps, parameters and return values of custom JavaScript functions, parameters and return values of JavaScript functions of third party technology with a JS API (e.g. Google Maps)
- **Replacement of expensive list processing** –
  The Uniface list is a String and therefore inefficient for any type of manipulation; the Struct is an ordered collection of references to individual data members in memory and therefore very efficient for any type manipulation
- **Complex data exchange between functions/entries/operations**
  the 4GL developer already has entries/operations that exchange complex data using lists and he wants to interact with those. The lists can be converted to Struct for further processing.

SOAP Support

SOAP Call-out

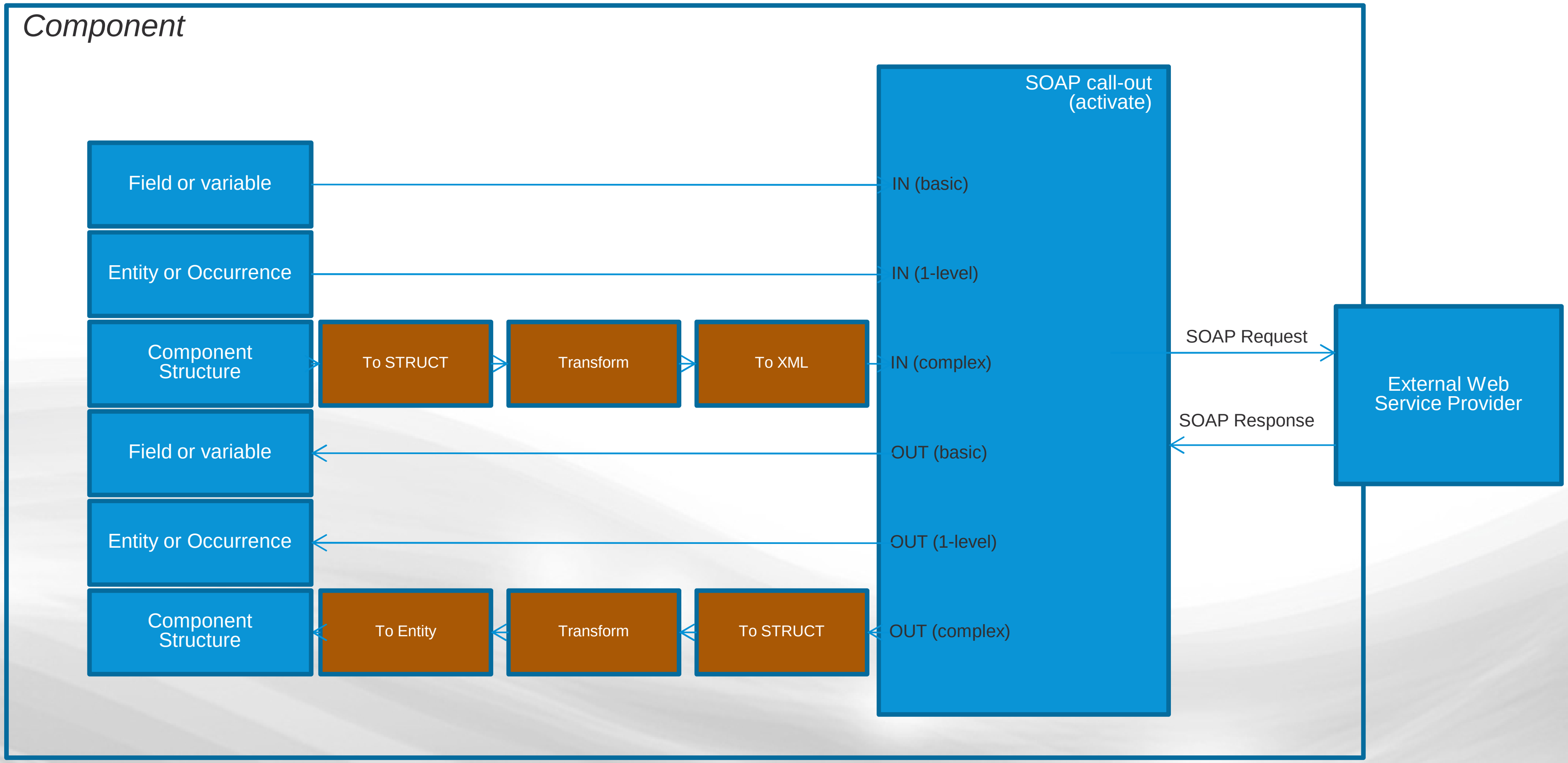
SOAP Header

SOAP Fault

SOAP call-out

- No changes in the SOAP implementation of 9.5
- Complex parameters are mapped to String (raw XML)

SOAP call-out

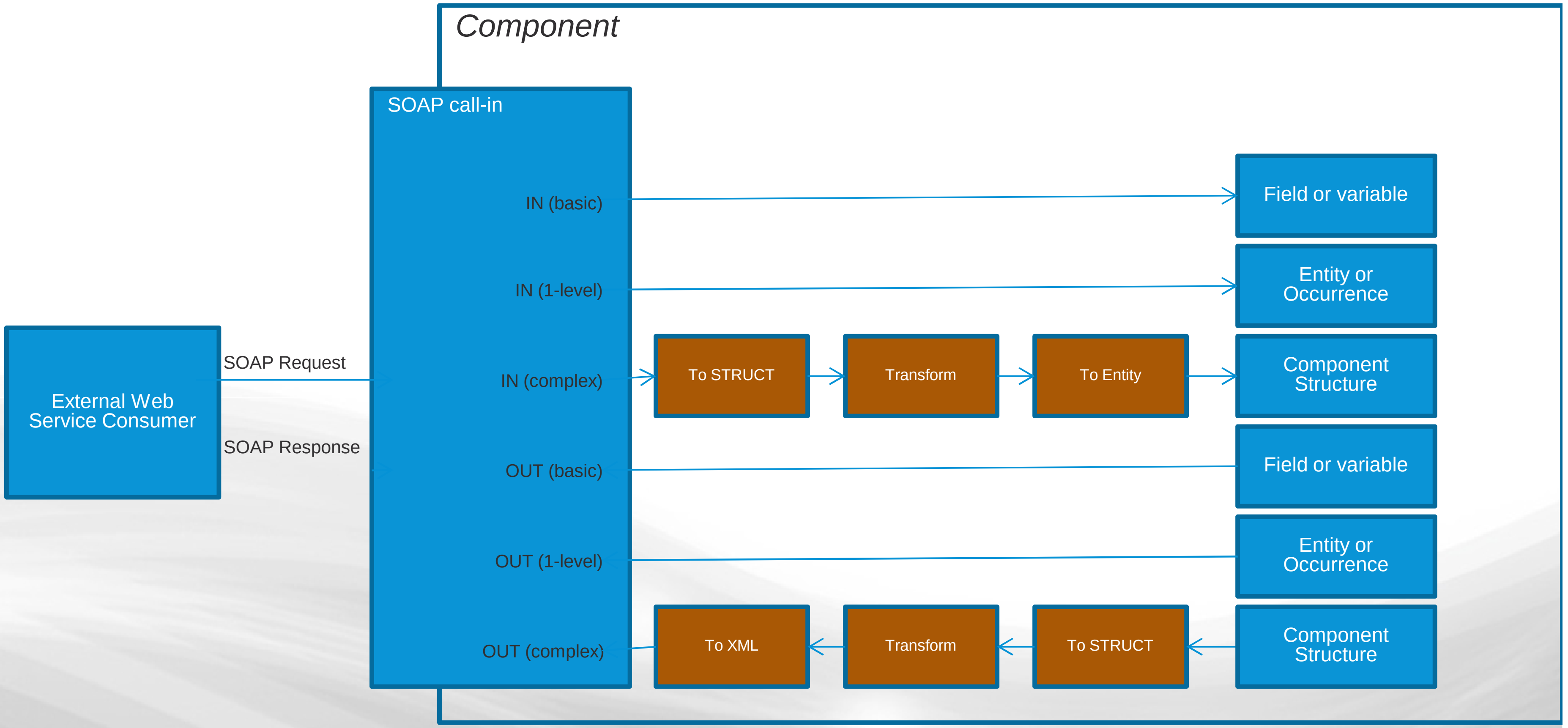


Compuware UNIFACE Development for the Cloud

```
entry getDetails
params
  string pOrderId : IN
endparams
variables
  struct    uOrder, uConfirmation
  xmlstream xOrder, xConfirmation
endvariables
  ; Get data
  ORDER_ID.ORDER/init = pOrderId
  retrieve "ORDER"
  if ($status < 0) return -1

  ; Transform data
  componentToStruct/one uOrder, "ORDER.SALES"                ; convert to STRUCT
  uOrder->ORDER->$name = "Order"                             ; rename entity
  uOrder->*->SHIP_TO->$name = "ShipTo"                   ; rename field
  uOrder->*->ORDER_ID->$tags->xmlClass = “attribute”     ; force to XML attribute iso. Element
  uOrder->ORDER->ORDER_ID->$name = "id"                      ; rename field
  structToXml/schema uOrder, xOrder, "order.xsd“             ; convert to XML

  ; Activate (SOAP call-out)
  activate "SHOP_SERVICE".putOrder(xOrder, xConfirmation)  ; IN-XML OUT-XML
  selectcase $procerror
  case "SOAP Fault returned"
    ...
  case ...
    ...
  endselectcase

  ; Transform data
  xmlToStruct/schema uConfirmation, xConfirmation, "confirmation.xsd“
  if (uConfirmation->Status = "OK") uConfirmation->Status = "1"

  ; Set data
  STATUS.ORDER = uConfirmation->Status
  store/e "ORDER"
  commit
end
```

Compuware UNIFACE® Development for the Cloud

SOAP call-in



Component
SOAP call-in
IN (basic)
IN (1-level)
IN (complex)
OUT (basic)
OUT (1-level)
OUT (complex)
External Web Service Consumer
SOAP Request
SOAP Response
To STRUCT
Transform
To Entity
To XML
Transform
To STRUCT
Field or variable
Entity or Occurrence
Component Structure
Field or variable
Entity or Occurrence
Component Structure

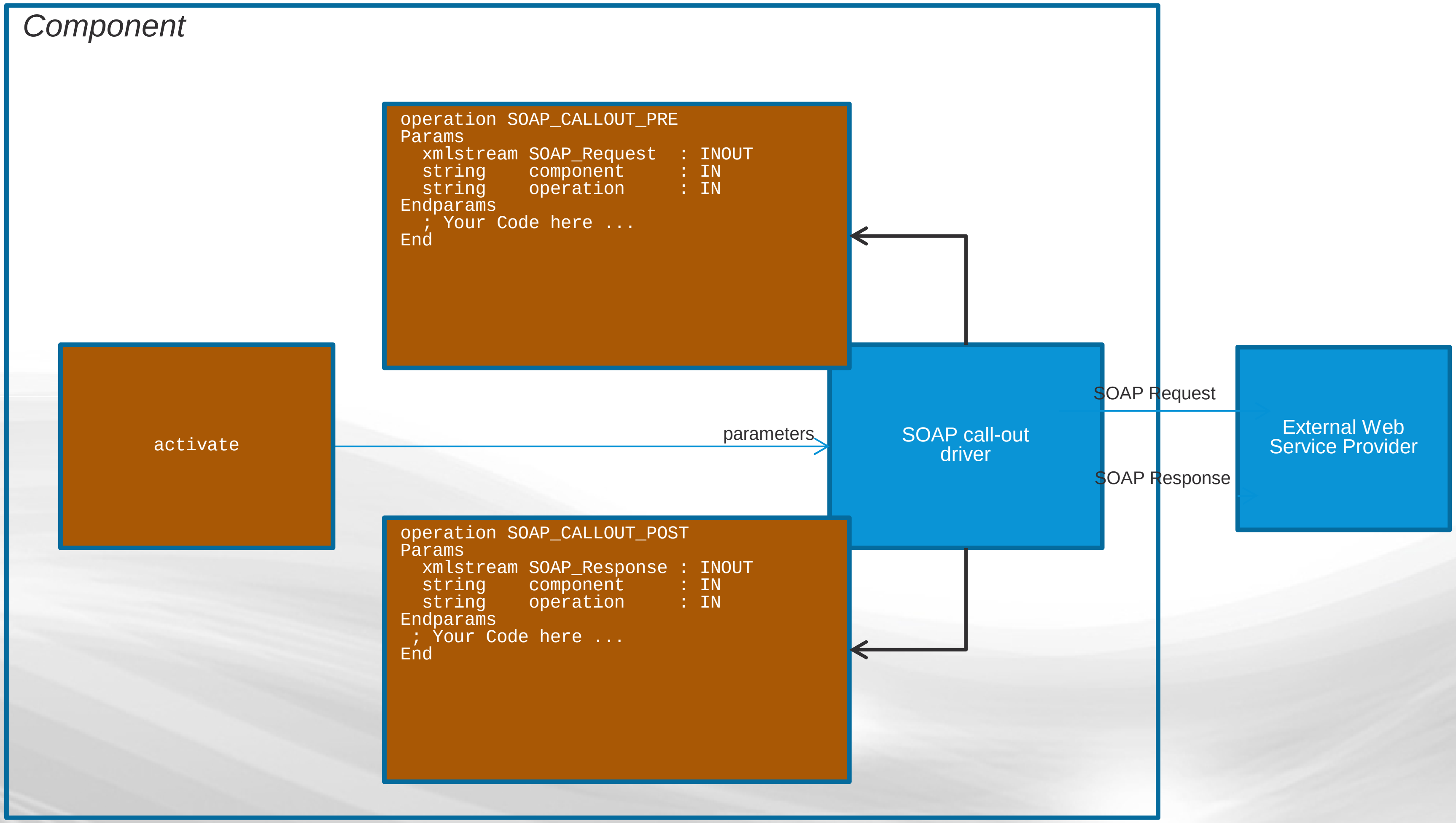
```
operation order
params
  xmlstream xOrder        : IN  ; some XML schema
  xmlstream xConfirmation : OUT ; some XML schema
endparams
variables
  struct uOrder, uConfirmation
endvariables
  ; transform data
  xmltostruct uOrder, xOrder
  uOrder->$name = "ORDER.SALES"
  if (uOrder->ShipTo != uOrder->BillTo)
    ; Unsupported -> return SOAP Fault
    return -1
  endif
  ; ...
  structtocomponent uOrded
  ; Process...
  ORDER_ID = $uuid
  store
  commit
  ; generate result
  if ($status = 0)
    uConfirmation->OrderStatus = "ordered"
    uConfirmation->OrderId = ORDER_ID.ORDER.SALES
  else
    uConfirmation->OrderStatus = "failed"
    ; ...
  endif
  structtoxml xConfirmation, uConfirmation
  return 0
end
```

Compuware UNIFACE Development for the Cloud

SOAP Header support

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"">
  <soap:Header>
    <transaction
soap:mustUnderstand="1“>234</transaction>
  </soap:Header>
  <soap:Body>
    <OrderForm>
      <Order id="21EC2020">
        <Date>20110801</date>
        <Status>2</status>
        <Total>7650</Total>
        <ShipTo>
          My house
          Dreef 60
          Amsterdam
        </ShipTo>
         <OrderLine id="1234FFFF ">
          <Item> PHB Slater I </item>
          <UnitPrice>3225</UnitPrice>
          <Quantity>2</Quantity>
          <Total>6450</Total>
        </OrderLine>
      </Order>
    </OrderForm>
  </soap:Body>
</soap:Envelope>
```

SOAP call-out call-back operations



```
operation SOAP_CALLOUT_PRE
Params
  xmlstream SOAP_Request  : INOUT
  string    component     : IN
  string    operation     : IN
Endparams
  ; Your Code here ...
End
```

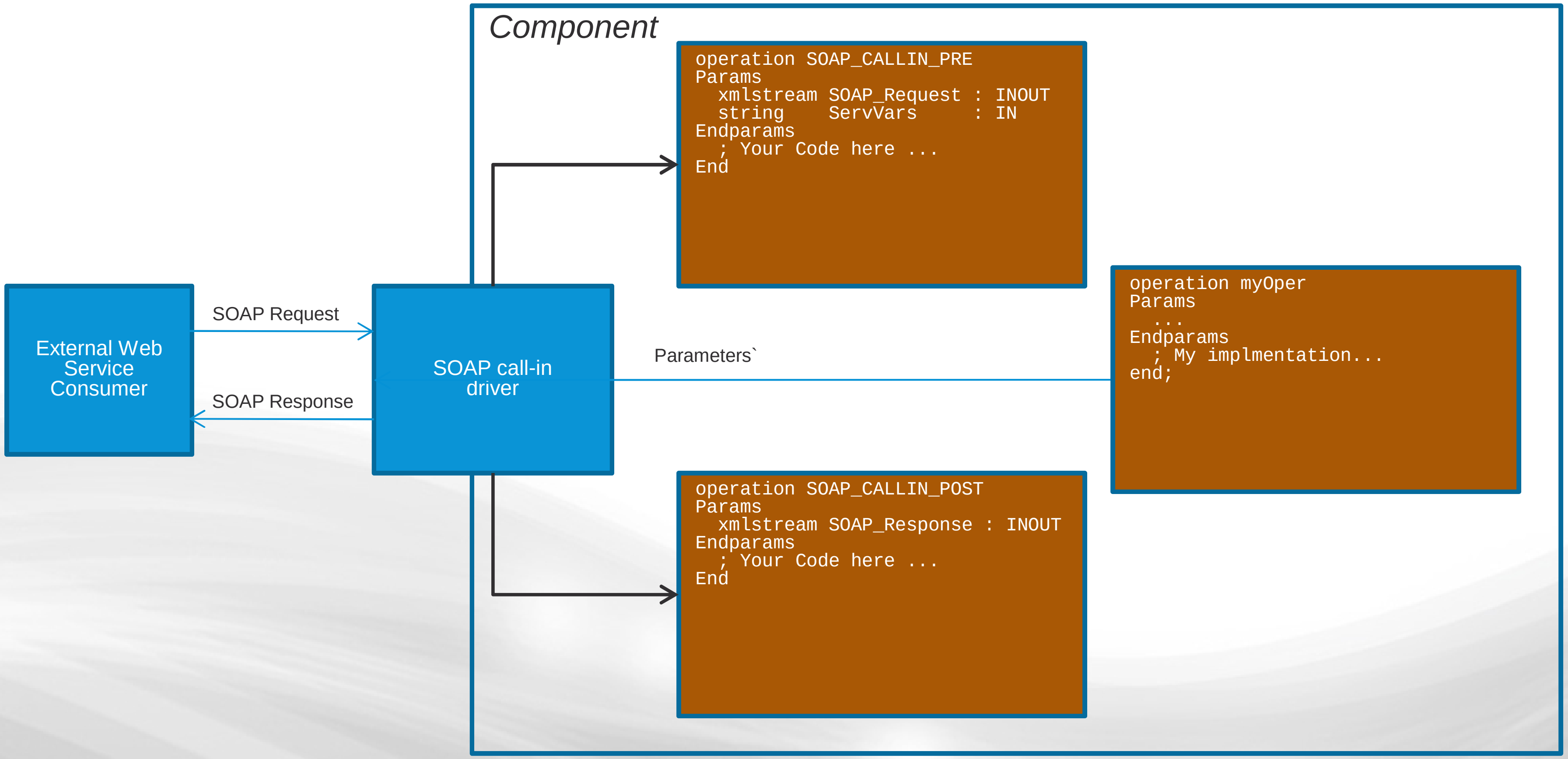
```
operation SOAP_CALLOUT_POST
Params
  xmlstream SOAP_Response : INOUT
  string    component     : IN
  string    operation     : IN
Endparams
 ; Your Code here ...
End
```

SOAP call-out call-back operations

Call-back operation execution sequence:

```
[DRIVER_SETTINGS]
USYS$SOP_PARAMS = callback=svc1,svc2,svc3

; overlaid with:
[SERVICES_EXEC]
MYSOAPCPT = $SOP:COMP1  callback=svc1,svc2
```

SOAP call-out call-back operations



```
operation SOAP_CALLIN_PRE
Params
  xmlstream SOAP_Request : INOUT
  string    ServVars     : IN
Endparams
  ; Your Code here ...
End
```

```
operation myOper
Params
  ...
Endparams
  ; My implmentation...
end;
```

```
operation SOAP_CALLIN_POST
Params
  xmlstream SOAP_Response : INOUT
Endparams
  ; Your Code here ...
End
```

Compuware UNIFACE Development for the Cloud

SOAP call-in call-back operations

Call-back operation execution sequence:

```
[SETTINGS]
$SOAP_CALLIN_CB = svc1, svc2, self,
  svc3
```

Where 'self' refers to the current activated instance

SOAP Fault support

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"">
  <soap:Header>
    <transaction
soap:mustUnderstand="1“>234</transaction>
  </soap:Header>
  <soap:Body>
    <soap:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <myfaultdetails>
          <message>
            My application didn't work
          </message>
          <errorcode>
            1001
          </errorcode>
        <myfaultdetails>
         </detail>
    </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Compuware
UNIFACE
Development for the Cloud

SOAP Fault support

- SOAP Fault automatically returned by SOAP call-in driver for any technical reasons
- Custom SOAP Faults can be returned using Struct and SOAP Call-back triggers
  (Use Proc to replace the SOAP Response with a SOAP Fault)
- Received SOAP Faults by SOAP call-out driver are available to Proc via $procerrorcontext

THANKS & QUESTIONS

Compuware UNIFACE®