1.1.1. Interaction Life-line

An Interaction completes normally when the message exchange completes successfully. An Interaction completes abnormally when:

- An application signals an error condition during the management of a request or within a party when processing the request
- The time-to-complete timeout, identifying the timeframe within which an Interaction MUST complete, occurs after the Interaction was initiated but before it completed
- Other types of errors, such as Protocol Based Exchange failures, Security failures, Document Validation errors

Interaction Syntax

The syntax of the *interaction* construct is:

```
<interaction  name="ncname"
              channelVariable="qname"
              operation="ncname"
              ~~time-to-complete="xsd:duration"?~~
              align="true"|"false"?
              initiate="true"|"false"? >


   <participate  relationship="qname"
                 fromRole="qname" toRole="qname" />

   <exchange  name="ncname"
              informationType="qname"? | channelType="qname"?
              action="request"|"respond" >

     <send      variable="XPath-expression"?
                recordReference=”list of ncname”?
                causeException=”true”|”false”? />

     <receive   variable="XPath-expression"?
                recordReference=”list of ncname”?
                causeException=”true”|”false”? />
   </exchange>*

   <timeout  time-to-complete="xsd:duration"|time-to-complete="xsd:deadline"
             fromRoleRecordReference=”ncname”?
             toRoleRecordReference=”ncname”? />?

   <record  name="ncname"
            when="before"|"after"|”timeout”
            causeException=”true”|”false”? >

     <source  variable="XPath-expression"? | expression=”Xpath-expression”? />
     <target  variable="XPath-expression" />
   </record>*
</interaction>
```

The attribute name is used for specifying a name for each Interaction element declared within a Choreography.

The channelVariable attribute specifies the Channel Variable containing information of a party, being the target of the Interaction, which is used for determining where and how to send and receive information to and into the party. The Channel Variable used in an Interaction MUST be available at the two Roles before the Interaction occurs.

At runtime, information about a Channel Variable is expanded further. This requires that the messages in the Choreography also contain correlation information, for example by including:

- A protocol header, such as a SOAP header, that specifies the correlation data to be used with the Channel, or
- Using the actual value of data within a message, for example the "Order Number" of the Order that is common to all the messages sent over the Channel

In practice, when a Choreography is performed, several different ways of doing correlation may be employed which vary depending on the Channel Type.

The operation attribute specifies the name of the operation that is associated with this Interaction. The specified operation belongs to the interface, as identified by the role and behavior elements of the Channel Type of the Channel Variable used in this Interaction.

~~The optional time-to-complete attribute identifies the timeframe within which an Interaction MUST complete after it was initiated.~~

The optional align attribute when set to "true" means that the Interaction results in the common understanding of both the information exchanged and the resulting observable information creations or changes at the ends of the Interaction as specified in the fromRole and the toRole. The default for this attribute is "false".

An Interaction activity can be marked as a Choreography initiator when the optional initiate attribute is set to "true". The default for this attribute is "false".

Within the participate element, the relationship attribute specifies the Relationship Type this Interaction participates in and the fromRole and toRole attributes specify the requesting and the accepting Role Types respectively. The Role Type identified by the toRole attribute MUST be the same as the Role Type identified by the role element of the Channel Type of the Channel Variable used in the interaction activity.

The optional exchange element allows information to be exchanged during an Interaction. Within this element, the attribute name is used for specifying a name for it.

Within the exchange element, the optional attributes informationType and channelType identify the Information Type or the Channel Type of the information that is exchanged between the two Roles in an Interaction. If none of these attributes are specified, then it is assumed that either no actual information is exchanged or the type of information being exchanged is of no interest to the Choreography definition.

Within the exchange element, the attribute action specifies the direction of the information exchanged in the Interaction:

- When the action attribute is set to "request", then the information exchange happens fromRole to toRole
- When the action attribute is set to "respond", then the information exchange happens from toRole to fromRole

Within the exchange element, the send element shows that information is sent from a Role and the receive element shows that information is received at a Role respectively in the Interaction:

- The send and the receive elements MUST only use the WS-CDL function getVariable within the variable attribute

- The optional Variables specified within the send and receive elements MUST be of type as described in the informationType or channelType attributes

- When the action element is set to "request", then the Variable specified within the send element using the variable attribute MUST be defined at the fromRole and the Variable specified within the receive element using the variable attribute MUST be defined at the toRole

- When the action element is set to "respond", then the Variable specified within the send element using the variable attribute MUST be defined at the toRole and the Variable specified within the receive element using the variable attribute MUST be defined at fromRole

- The Variable specified within the receive element MUST not be defined with the attribute silent set to “true”

- Within the send or the receive element(s) of an exchange element, the recordReference attribute contains a list of references to record element(s) in the same Interaction. The same record element MAY be referenced from different send or the receive element(s) within the same Interaction thus enabling re-use

- Within the send or the receive element(s) of an exchange element, the causeException attribute when set to “true”, specifies that an Exception will be caused at the respective Roles. In this case, the informationType of the exchange element MUST be of Exception Type

- The request exchange MUST NOT have causeException attribute set to “true”

- When two or more respond exchanges are specified, one respond exchange MAY be of normal informationType and all others MUST be of Exception Type. There is an implicit choice between two or more respond exchanges

- If the align attribute is set to "false" for the Interaction, then it means that the:

    - Request exchange completes successfully for the requesting Role once it has successfully sent the information of the Variable specified within the send element and the Request exchange completes successfully for the accepting Role once it has successfully received the information of the Variable specified within the receive element

  - Response exchange completes successfully for the accepting Role once it has successfully sent the information of the Variable specified within the send element and the Response exchange completes successfully for the requesting Role once it has successfully received the information of the Variable specified within the receive element

- If the align attribute is set to "true" for the Interaction, then it means that the:

  - Interaction completes successfully if the Request and the Response exchanges complete successfully and all referenced records complete successfully

  - Request exchange completes successfully once both the requesting Role has successfully sent the information of the Variable specified within the send element and the accepting Role has successfully received the information of the Variable specified within the receive element

  - Response exchange completes successfully once both the accepting Role has successfully sent the information of the Variable specified within the send element and the requesting Role has successfully received the information of the Variable specified within the receive element

Within the OPTIONAL timeout element, the time-to-complete attribute identifies the timeframe within which an Interaction MUST complete after it was initiated. Alternatively, the deadline attribute identifies the deadline before an Interaction MUST complete. The fromRoleRecordReference and toRoleRecordReference attributes identifies references to record element(s) in the same Interaction that will take effect when a timeout occurs.

The optional element record is used to create or change one or more Variables using another Variable or an expression. Within this element, the attribute name is used for specifying a name for it. Within the record element, the source and target elements specify these recordings of information at the send and receive ends of the Interaction:

- When the action element is set to "request", then the recording(s) specified within the source and the target elements occur at the fromRole for the send and at the toRole for the receive
- When the action element is set to "response", then the recording(s) specified within the source and the target elements occur at the toRole for the send and at the fromRole for the receive

Within the record element, the when attribute specifies if a recording happens before or after a send or “before” or “after” a receive of a message at a Role in a Request or a Response exchange or when a “timeout”, time-to-complete or deadline has expired at a role.. When the when attribute is set to “timeout”, the record element indicates the record to be performed when a timeout occurs. If two or more record elements have the same

value in their when attribute and are referenced within the recordReference attribute of a send or a receive element, then they are performed in the order in which they are specified.
The following rules apply for the information recordings when using the record element:

- The source MUST define either a variable attribute or an expression attribute:
    - When the source defines an expression attribute this MUST contain expressions, as defined in Section 2.4.3. The resulting type of the defined expression MUST be compatible with the target Variable type
    - When the source defines a Variable, then the source and the target Variable MUST be of compatible type
    - When the source defines a Variable, then the source and the target Variable MUST be defined at the same Role
- When the attribute variable is defined it MUST use only the WS-CDL function getVariable
- The target Variable MUST NOT be defined with the attribute silent set to "true"
- One or more record elements MAY be specified and performed at one or both the Roles within an Interaction
- A record element MUST NOT be specified in the absence of an exchange or the timeout element
- At most one record element MAY be specified where the when attribute is set to "timeout" for fromRole and toRole respectively
- The attribute causeException MAY be set to "true" in a record element if the target Variable is an Exception Variable
- When the attribute causeException is set to "true" in a record element, the corresponding Role gets into Exception state
- When two or more record elements are specified for the same Role in an Interaction with target Variables of Exception Type, one of the Exception recordings MAY occur. An Exception recording has an non-observable predicate condition, associated implicitly with it, that decides if an Exception occurs
- If the align attribute is set to "false" for the Interaction, then it means that the Role specified within the record element makes available the creation or change of the information specified within the record element immediately after the successful completion of each record
- If the align attribute is set to "true" for the Interaction, then it means that
    - Both Roles know the availability of the creation or change of the information specified within the record element only at the successful completion of the Interaction

- o If there are two or more record elements specified within an Interaction, then all record operations MUST complete successfully for the Interact to complete successfully. Otherwise, none of the Variables specified in the target attribute will be affected

The example below shows a complete Choreography that involves one Interaction performed from Role Type “Consumer” to Role Type “Retailer” on the Channel "retailer-channel" as a request/response exchange:

- The message “purchaseOrder” is sent from the “Consumer” to the “Retailer” as a request message
- The message “purchaseOrderAck” is sent from the “Retailer” to the “Consumer” as a response message
- The Variable “consumer-channel” is made available at the “Retailer” using the record element
- The Interaction happens on the “retailer-channel”, which has a Token Type “purchaseOrderID” used as an identity element of the channel. This identity element is used to identify the business process of the “Retailer”
- The request message “purchaseOrder” contains the identity of the “Retailer” business process as specified in the tokenLocator for ”purchaseOrder” message
- The response message “purchaseOrderAck” contains the identity of the “Consumer” business process as specified in the tokenLocator for “purchaseOrderAck” message
- The “consumer-channel” is sent as a part of “purchaseOrder” Interaction from the “Consumer” to the “Retailer” on “retailer-channel” during the request. Here the record element makes available the “Consumer-channel” at the “Retailer” Role. If the align attribute was set to "true" for this Interaction, then it also means that the “Consumer” knows that the “Retailer” now has the contact information of the “Consumer”. In another example, the “Consumer” could set its Variable "OrderSent" to "true" and the “Retailer” would set its Variable "OrderReceived" to "true" using the record element
- The exchange “badPurchaseOrderAckException” specifies that an Exception of “badPOAckType” Exception Type could occur at both parties

```
<package name="ConsumerRetailerChoreography” version="1.0"
  <informationType name="purchaseOrderType" type="pons:PurchaseOrderMsg"/>
  <informationType name="purchaseOrderAckType" type="pons:PurchaseOrderAckMsg"/>
  <informationType name=”badPOAckType”  type=”xsd:string” exceptionType=”true”/>

  <token name="purchaseOrderID" informationType="tns:intType"/>
  <token name="retailerRef" informationType="tns:uriType"/>
```

```
<tokenLocator tokenName="tns:purchaseOrderID"
              informationType="tns:purchaseOrderType" query="/PO/orderId"/>
<tokenLocator tokenName="tns:purchaseOrderID"
              informationType="tns:purchaseOrderAckType" query="/PO/orderId"/>

<roleType name="Consumer">
  <behavior name="consumerForRetailer" interface="cns:ConsumerRetailerPT"/>
  <behavior name="consumerForWarehouse" interface="cns:ConsumerWarehousePT"/>
</roleType>
< roleType name="Retailer">
  <behavior name="retailerForConsumer" interface="rns:RetailerConsumerPT"/>
</roleType>

<relationshipType name="ConsumerRetailerRelationship">
  <role type="tns:Consumer" behavior="consumerForRetailer"/>
  <role type="tns:Retailer" behavior="retailerForConsumer"/>
</relationshipType>

<channelType name="ConsumerChannel">
  <role type="tns:Consumer"/>
  <reference>
    <token type="tns:consumerRef"/>
  </reference>
  <identity>
    <token type="tns:purchaseOrderID"/>
  </identity>
</channelType>

<channelType name="RetailerChannel">
  <passing channel="ConsumerChannel" action="request" />
  <role type="tns:Retailer" behavior="retailerForConsumer"/>
  <reference>
    <token type="tns:retailerRef"/>
  </reference>
  <identity>
    <token type="tns:purchaseOrderID"/>
  </identity>
</channelType>

<choreography name="ConsumerRetailerChoreography” root="true">
  <relationship type="tns:ConsumerRetailerRelationship"/>
  <variableDefinitions>
    <variable name="purchaseOrder" informationType="tns:purchaseOrderType"
              silent="true" />
    <variable name="purchaseOrderAck"
              informationType="tns:purchaseOrderAckType" />
    <variable name="retailer-channel" channelType="tns:RetailerChannel"/>
    <variable name="consumer-channel" channelType="tns:ConsumerChannel"/>
    <variable name=”badPurchaseOrderAck”
              informationType=”tns:badPOAckType” role="tns:Consumer"/>
    <variable name=”badPurchaseOrderAck”
              informationType=”tns:badPOAckType” role="tns:Retailer"
              silent="true" />
  </variableDefinitions>

  <interaction name=”createPO”
               channelVariable="tns:retailer-channel"
               operation="handlePurchaseOrder" align="true"
               initiate="true">
    <participate relationship="tns:ConsumerRetailerRelationship"
                 fromRole="tns:Consumer" toRole="tns:Retailer"/>

    <exchange name="request”
```

```
                informationType="tns:purchaseOrderType" action="request">
        <send variable="cdl:getVariable(“tns:purchaseOrder”, “”, “”)" />
        <receive variable="cdl:getVariable(“tns:purchaseOrder”, “”, “”)"
                 recordReference="record-the-channel-info" />
      </exchange>

      <exchange name="response”
                informationType="purchaseOrderAckType" action="respond">
        <send variable="cdl:getVariable(“tns:purchaseOrderAck”, “”, “”)" />
        <receive variable="cdl:getVariable(“tns:purchaseOrderAck”, “”, “”)" />
                 recordReference=" recordBadPurchaseOrder " />
      </exchange>

      <exchange name="badPurchaseOrderAckException”
                informationType="badPOAckType" action="respond">
        <send variable="cdl:getVariable(“tns:badPurchaseOrderAck”, “”, “”)"
                causeException=”true” />
        <receive variable="cdl:getVariable(“tns:badPurchaseOrderAck”, “”, “”)"
                causeException=”true” />
      </exchange>

      <record name="record-the-channel-info" when="after">
        <source variable="cdl:getVariable(“tns:purchaseOrder, “”,
                                          “PO/CustomerRef”)"/>
        <target variable="cdl:getVariable(“tns:consumer-channel”, “”, “”)"/>
      </record>

    </interaction>
  </choreography>
</package>
```