**Business Process Service for API Consumers**

There are two types of items in NIMBLE: *Product* and *Logistics Service*. Each one has its own business process flow in terms of available business process definitions.

*Business process flow for products:*

Request for Information => PPAP => Negotiation => Order => Fulfilment

*Business process flow for logistics services:*

Transport Service Information Request => Transport Service Negotiation => Transport Service Plan

Currently, there are six different business process templates supported by NIMBLE. For further details about the business processes, the following document would be useful: <https://www.nimble-project.org/wp-content/uploads/2018/06/NIMBLE_D3_4.pdf>

* **Information Request** is used to request detailed information for an item (product/service) traded on NIMBLE before ordering the item.
* **PPAP (Production Part Approval Process)** is the process used to be sure that the supplier meets the buyer’s requirements in terms of design and production processes.
* **Negotiation** is the process which enables a buyer party to negotiate the default terms specified by the manufacturer/service provide and request additional terms.
* **Order** is used for ordering the product after completing the negotiation for terms.
* **Transport Execution Plan** is used to determine details for the transport service provider to grab the products from the manufacturer’s site and start shipment.
* **Fulfilment** is the only process which is initiated by the seller party. It is used to inform the buyer party that the ordered products are shipped.

An open source business process management tool, Camunda, is used to instantiate a given process definition (Information Request in this example). It simply creates a process instance which represents the Information Request process. It also assigns an identifier to all process instances created so that we can access those whenever they are needed. The message exchanged in a particular step of the business process is stored in the database. Each such message is referred as a ***document***.

Each business process in NIMBLE is represented with two sequential activities: a request and a response. At each step a document is exchanged between trading companies. For each document, a ***ProcessDocumentMetadata*** is created, which refers to the actual exchanged document as well as other metadata including but not limited submission date, associated process instance identifier or identifiers of the initiator and responder companies.

Concerning the flow of the business process as presented before, the processes included the flows are related to each other as they are preliminary activities before closing a deal on a product or service. Furthermore, business process flows could be related to each other. For example, after a confirmed order confirmed, a logistics service business process flow might be performed to arrange transport services for that order. From a seller point of view, such business process flows are related as the logistics service related activities are a continuation of the order. In this specific case, some information related to logistics service processes are initialized with the information obtained from the ordering process e.g. the product to be shipped, latest delivery dates and so on.

To handle related processes and process flows, two concepts namely ***CollaborationGroup*** and ***ProcessInstanceGroup*** are introcduced. ProcessInstanceGroups contain the related processes which are performed for the same product/services whereas CollaborationGroups contain all ProcessInstanceGroups which are created for related processes.

**A Simple Scenario**

A simple scenario introduced as a basis to concepts that are already introduced and to be introduced. The scenario involves three companies with buyer, seller and transport service provider roles.

As a first step, the buyer wants to negotiate about a product. Therefore, s/he initiates a negotiation process. Once a new business process is instantiated, a ProcessInstanceGroup referring to the new process and a CollaborationGroup containing the new ProcessInstanceGroup are created automatically. The seller confirms the proposed terms, and as the next step, the buyer initiates an order with the terms and conditions agreed in the negotiation process. The order process instance is created in the same ProcessInstanceGroup with the negotiation. The seller confirms the order also.

Following the order process, the seller looks for a transport service provider to ship the product. After identifying a service on the platform, s/he initiates a negotiation process for the service. A new ProcessInstanceGroup is created referring to the negotiation process including the seller and transport service provider companies. The new ProcessInstanceGroup is associated to the existing CollaborationGroup.

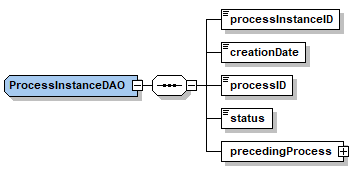
In this scenario, we have exactly two different collaborations. In NIMBLE, a collaboration indicates the union of processes which are inside the same ProcessInstanceGroup. Collaborations might be rated. For this purpose, ***CompletedTask*** objects are used. A CompletedTask is used to represent a completed or cancelled collaboration. It is mainly used to store ratings and reviews regarding to the collaboration as stated above. Canceling a collaboration is possible only if there is no accepted order or transport execution plan in the collaboration.

**Data Model**

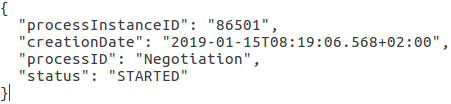
Although Business Process services use the Universal Business Language (UBL) standard to represent documents exchanged in business processes, additional data entities are utilized to manage other aspects of business processes.

**Process Instance**

ProcessInstances represent an individual business process instance.



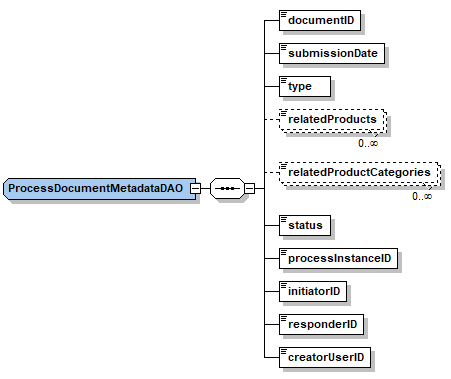
* ***processInstanceID*** is the identifier of the process. It is generated by Camunda when the process instance is started.
* ***processID*** is the type of the process. The followings are the possible values:
  + **Item\_Information\_Request**
  + **Ppap**
  + **Negotiation**
  + **Order**
  + **Transport\_Execution\_Plan**
  + **Fulfilment**
* ***creationDate*** *is the date when the process is started*
* ***status*** status of the process instance. Possible values are the followings:
  + ***STARTED***it indicates that process instance is started (in other words, request is sent)
  + ***CANCELLED*** it indicates that process instance is canceled before getting a response for the request
  + ***COMPLETED*** it indicates that process instance is completed by getting a response for the request



**Figure 1**: Process instance example for negotiation

**Process Document Metadata**

For each step in a business process (i.e. request / response), a ProcessDocumentMetadata is created. A ProcessDocumentMetadata instance contains several details for the step as elaborated below.



* ***documentID*** the identifier of the document which is exchanged as a message in an individual step of a business process
* ***submissionDate*** the date when the process request/response is sent
* ***type*** type of the document. Some possible values are ORDER,QUOTATION, RECEIPTADVICE, PPAPRESPONSE. It is important to notice that there are two types of documents (one for request and one for response) which are involved in an individual business process (e.g. Order, PPAP, etc.) together.
* ***relatedProducts*** A business process does relate to a product/service. This field holds the names of the products.
* ***relatedProductCategories*** categories of the products for which the process document metadata is created.
* ***status*** status of the ProcessDocumentMetadata. Possible values are the followings:
  + ***APPROVED***: it indicates that the request is responded (and accepted if applicable).
  + ***WAITINGRESPONSE***: it indicates that the request is sent, but the response is not sent yet
  + ***DENIED*** : it indicates that the respond is sent but the request is rejected
* ***processInstanceID*** the identifier of the process instance which is created by Camunda
* ***initiatorID*** the identifier of the party initiating the process
* ***responderID*** the identifier of the party responding the process
* ***creatorUserID*** the identifier of the user who either starts a business process or responds to an existing one

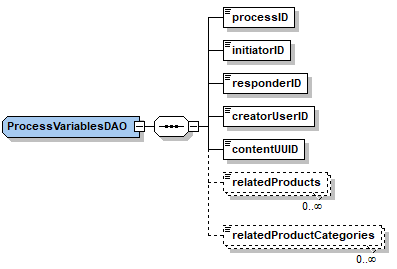


**Figure 2:** ProcessDocumentMetadata of negotiation request (type: REQUESTFORQUOTATION) vs ProcessDocumentMetadata of negotiation response (type: QUOTATION)

This figure’s important to understand ProcessDocumentMetadata concept. Since each process document metadata has a reference to the exchanged document, each of them will have a different *documentID*. *processInstanceID* will remain same because these two documents are related to the same process instance. Although *creatorUserID* of the document (the user who starts the process, continues the process or updates the process) can change, *initiatorID* and *responderID* will be the same since they actually represent the seller and buyer parties.

**Process Variables**

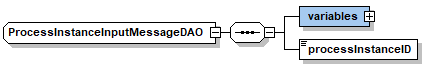
This is the object that encapsulating the information to start/continue a business process. Those are the data which are required by Camunda to start or continue a process instance.



* ***processID*** is the type of the process. The followings are the possible values:
  + **Item\_Information\_Request**
  + **Ppap**
  + **Negotiation**
  + **Order**
  + **Transport\_Execution\_Plan**
  + **Fulfilment**
* ***initiatorID*** the identifier of the party initiating the process
* ***responderID*** the identifier of the party responding the process
* ***contentUUID*** the identifier of the exchanged document
* ***relatedProducts*** names of the products which the process is related to
* ***relatedProductCategories*** categories of the products which the process is related to
* ***content*** the serialized form of the exchanged document
* ***creatorUserID*** the identifier of the user who starts a business process, responds to an existing one or updates the process instance

**Process Instance Input Message (Piim)**

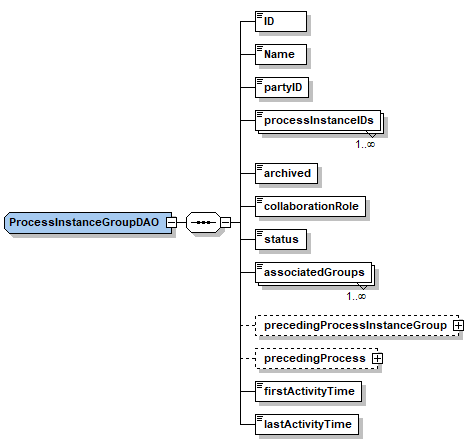
In addition to process variables, we also need *processInstanceID* to continue a business process. This is why we have this object. By checking the given process instance id, services can decide which process instance this response belongs to.



* ***variables*** the process variables object (For more information,refer to ‘2.3 Process Variables’)
* ***processInstanceID*** the identifier of the process instance which the response belongs to

**Process Instance Group**

This is the model which contains all process instances related to the same product.

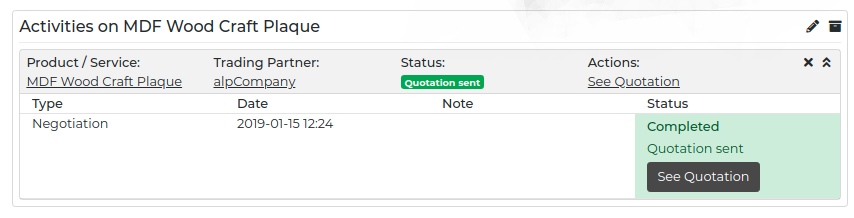


* ***id*** the identifier of the ProcessInstanceGroup
* ***name*** the name of the ProcessInstanceGroup. This field is set according to the name of the product which the process instance group is created for
* ***partyID*** the identifier of the party owning the process instance group
* ***processInstanceIDs*** the identifiers of the process instances which the ProcessInstanceGroup holds
* ***archived*** whether the process instance group is archived or not
* ***collaborationRole*** role of the party owning the process instance group in the collaboration. SELLER and BUYER are two possible values
* ***associatedGroups*** identifiers of the process instance groups which are created for trading partners
* ***firstActivityTime*** the date when the process instance group is created
* ***lastActivityTime*** the date when the last activity such as sending a request or response is performed
* ***precedingProcessInstanceGroup*** the previous ProcessInstanceGroup which the current ProcessInstanceGroup is derived from
* ***precedingProcess*** the process instance object which represent the object (order) used to initiate transport negotiation request

When a process is started from scratch (with no previous processes), two different ProcessInstanceGroups will be created. One is for the initiating party and the other one is for the owner of the product, i.e., target trading partner. If the user continues with other processes (for example, PPAP after Item Information Request), those processes are also added to the existing ProcessInstanceGroup.

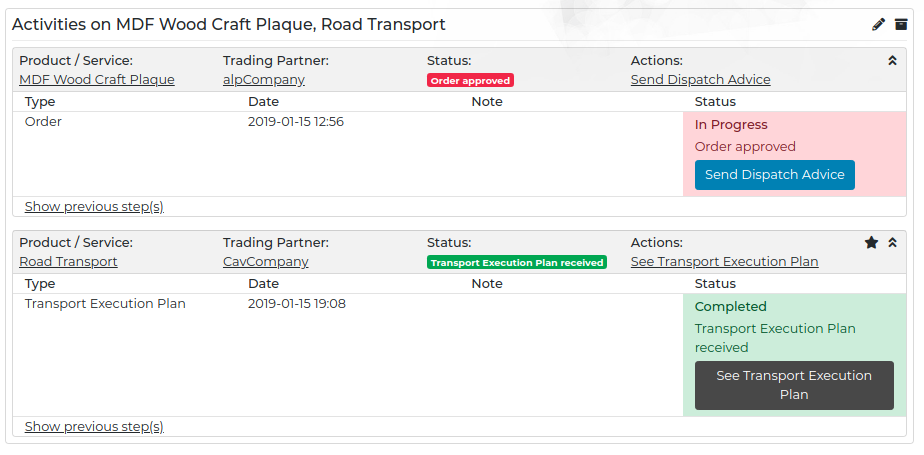
According to the simple scenario described above, after responding the order, the seller initiates a negotiation process on a transport service. At this step, an additional ProcessInstanceGroup will be created to keep processes related to the transport service. Specifically, *precedingProcess* and *precedingProcessInstanceGroup* fields are used to establish a relationship between the preceding and subsequent groups.

The order which is used to initiate transport negotiation request (or in other words, the order for which a transport service is looked for) is referred by the *precedingProcess* field of the latter group. *precedingProcessInstanceGroup* is the ProcessInstanceGroup which contains that order process.

**Figure 3**: Seller’s CollaborationGroup including a single ProcessInstanceGroup

**Collaboration Group**

A CollaborationGroup is intended to keep collaborations of a company with several different companies on an individual order/product. For example, considering the scenario, from a seller’s point of view, a CollaborationGroup would contain collaborations with the buyer and transport service provider.

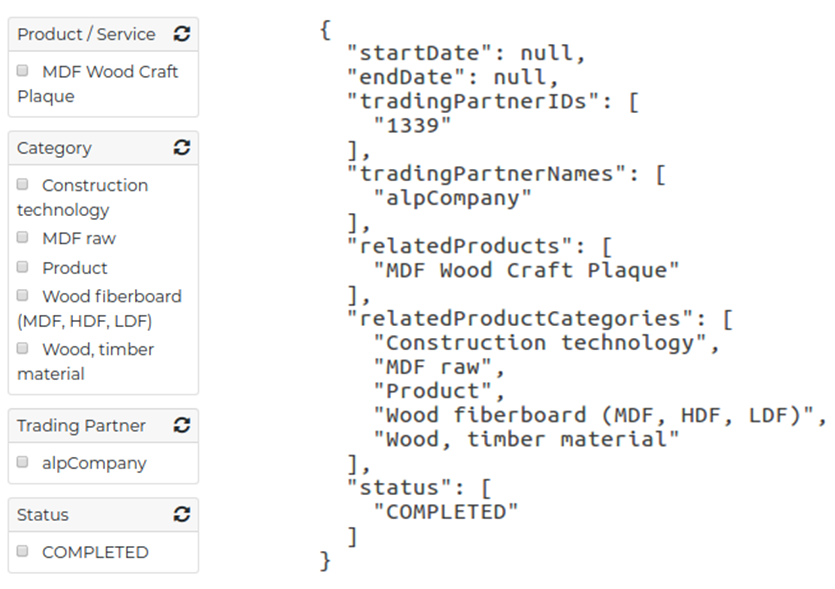
* ***id*** the identifier of the CollaborationGroup.
* ***name*** the name of the CollaborationGroup. If the owner of the CollaborationGroup does not specify a name, it will have a default name which is made of names of groups inside the collaboration group.
* ***archived*** whether the CollaborationGroup is archived or not
* ***associatedCollaborationGroups*** identifiers of the CollaborationGroups of the trading partners. The referred groups contain process instances related to processes included in this group.
* ***associatedProcessInstanceGroups*** identifiers of the ProcessInstanceGroups which this CollaborationGroup contains

***Figure 4:*** *Seller’s CollaborationGroup which contains two different process instance group (one is for MDF Wood Craft Plaque and the other is for Road Transport)*

**Process Instance Group Filter**

It enables us to filter ProcessInstanceGroups according to a set of parameters such as status of the group, related products or product categories. It’s mainly used on the front-end to allow users to filter their own groups.

* ***startDate*** the date when the first activity related the group is performed
* ***endDate*** the date when the last activity related the group is performed
* ***tradingPartnerIDs*** identifiers of the trading partners
* ***tradingPartnerNames*** names of the trading partners
* ***relatedProducts*** the names of products belonging to the process instance group
* ***relatedProductCategories*** the categories of the products belonging to the process instance group
* ***status*** status of the process instance. Possible values are the followings:
  + ***STARTED*:**it indicates that process instance is started (in other words, request is sent)
  + ***CANCELLED*:**it indicates that process instance is canceled before getting a response for the request
  + ***COMPLETED*:**it indicates that process instance is completed by getting a response for the request

****

**Figure 5:** Process instance group filter example

**Completed Task**

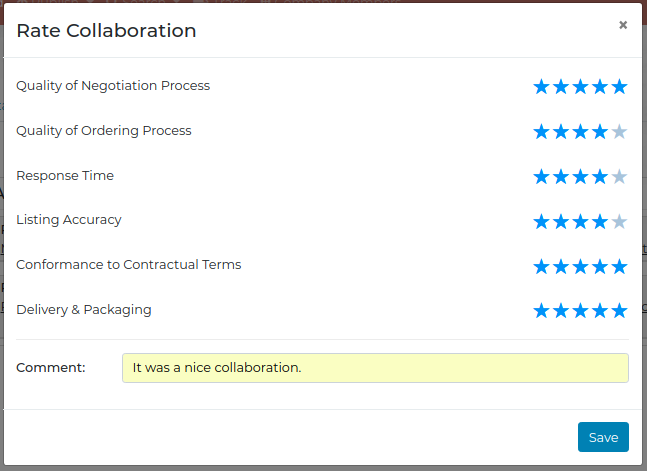
CompletedTasks are used to represent a completed collaboration between parties. Responding a transport execution plan request or sending a receipt advice would complete a collaboration concerning the currently available business process workflows. Moreover, canceling a business process is another way to complete a collaboration. Business process instance which completes the collaboration is associated with the corresponding completed task. In other words, each completed task will have a process instance id to identify them.

* ***associatedProcessInstanceID*** identifier of the process instance which is the last process before completing the collaboration
* ***description*** the reason why this completed task is created. There are two possible values :
  + **Completed**: after sending a receipt advice or responding a transport execution plan
  + **Cancelled**: after canceling the process
* ***evidenceSupplied*** the ratings related to the collaboration
* ***period*** the time passed from the first request to the last response in the collaboration
* ***comment*** the reviews related to the collaboration

**Qualifying Party**

Qualifying party is mainly used to keep a reference to completed collaborations which are represented with CompletedTasks.

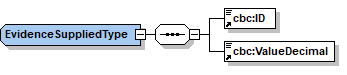
* ***completedTask*** the completed task list of the party
* ***party*** the party representing the owner of the qualifying party

****

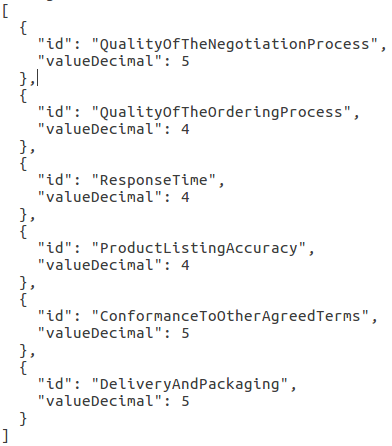
**Figure 6:** Ratings and reviews for a collaboration

**Evidence Supplied Type**

This concept is used to keep ratings and reviews about a collaboration. Rating a collaboration is possible when the collaboration is completed (sending receipt advice or sending transport execution plan) or canceled.



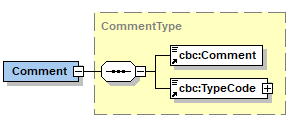
* ***id*** name of the criteria which is rated such as ‘ResponseTime’,’ProductListingAccuracy’ or ‘DeliveryAndPackaging’
* ***valueDecimal*** value of the rating (over 5)



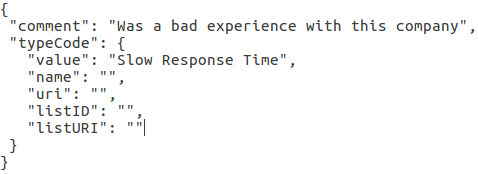
**Figure 7:** Ratings example

**2.11 Comment Type**

CommentType is used to represent the reviews of the buyer/seller about a collaboration. After completing a collaboration (sending receipt advice or sending transport execution plan), the buyer can comment on the collaboration or after canceling the process, both parties can provide a verbal, descriptive explanation why the collaboration is canceled.



* ***comment*** comment about the collaboration
* ***typeCode*** the type code field keeps enumerated values for the reasons of unsuccessful collaborations such as slow response time, rejected delivery terms or undervalued offer.



**Figure 8:** Reviews example

**REST API**

Although the REST API of the business process offers varying types of services, the most notable endpoints are the *start[[1]](#footnote-1)* and *continue[[2]](#footnote-2)* endpoints, which allow starting and continuing a business process respectively as their names imply. Other group services are related to management of ***CollaborationGroups[[3]](#footnote-3)***, ***ProcessInstanceGroups[[4]](#footnote-4)*** and ***ProcessInstances[[5]](#footnote-5)***. Trust-related services let creation and retrieval of ratings for collaboration activities. Statistics services[[6]](#footnote-6) provide retrieval of several type of statistics including but not limited to negotiation times, response times or business process counts. While contract service[[7]](#footnote-7) provides management ***Contract*** instances, contract-generator service[[8]](#footnote-8) provides capabilities to context-aware contract bundles and terms & conditions. EPC service[[9]](#footnote-9) includes operations to manage EPC codes associated to order processes. Lastly, document service[[10]](#footnote-10) enables retrieval of the documents that are exchanged throughout the business processes.

The latest released version of the REST API of the business service can also be found at: http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/

Below, the REST API components are summarized along with the services included in them:

**Start and Continue API (http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/start-controller)**

This controller has a single service to start a business process between two trading partners.

* Starting a business process service: http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/start-controller/startProcessInstanceUsingPOST

**Continue API (http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/continue-controller)**

This controller has a single service to continue a business process that has been already started

* Continuing a business process service: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/continue-controller/continueProcessInstanceUsingPOST](http://localhost:8081/swagger-ui.html#!/continue-controller/continueProcessInstanceUsingPOST)

**Collaboration Groups Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/collaboration-groups-controller**](http://localhost:8081/swagger-ui.html#/collaboration-groups-controller)**)**

This API offers services to manage ***CollaborationGroups***.

* Retrieving collaboration groups: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/collaboration-groups-controller/getCollaborationGroupsUsingGET](http://localhost:8081/swagger-ui.html#!/collaboration-groups-controller/getCollaborationGroupsUsingGET)
* Retrieving a specific collaboration group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/collaboration-groups-controller/getCollaborationGroupUsingGET](http://localhost:8081/swagger-ui.html#!/collaboration-groups-controller/getCollaborationGroupUsingGET)
* Updating the name of the collaboration group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/collaboration-groups-controller/updateCollaborationGroupNameUsingPATCH](http://localhost:8081/swagger-ui.html#!/collaboration-groups-controller/updateCollaborationGroupNameUsingPATCH)
* Deleting a collaboration group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/collaboration-groups-controller/deleteCollaborationGroupUsingDELETE](http://localhost:8081/swagger-ui.html#!/collaboration-groups-controller/deleteCollaborationGroupUsingDELETE)
* Archiving a collaboration group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/collaboration-groups-controller/archiveCollaborationGroupUsingPOST](http://localhost:8081/swagger-ui.html#!/collaboration-groups-controller/archiveCollaborationGroupUsingPOST)
* Restoring a collaboration group: http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/collaboration-groups-controller/restoreCollaborationGroupUsingPOST

**Process Instance Group Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/process-instance-group-controller**](http://localhost:8081/swagger-ui.html#/process-instance-group-controller)**)**

This API includes services to manage ***ProcessInstanceGroups***.

* Retrieving a specific process instance group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-group-controller/getProcessInstanceGroupUsingGET](http://localhost:8081/swagger-ui.html#!/process-instance-group-controller/getProcessInstanceGroupUsingGET)
* Cancelling a process instance group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-group-controller/cancelCollaborationUsingPOST](http://localhost:8081/swagger-ui.html#!/process-instance-group-controller/cancelCollaborationUsingPOST)
* Deleting a process instance group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-group-controller/deleteProcessInstanceGroupUsingDELETE](http://localhost:8081/swagger-ui.html#!/process-instance-group-controller/deleteProcessInstanceGroupUsingDELETE)
* Retrieving the order process instance inside a process instance group given a process instance id related to the group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-group-controller/getOrderProcessUsingGET](http://localhost:8081/swagger-ui.html#!/process-instance-group-controller/getOrderProcessUsingGET)
* Retrieving search filters based on trading partners and products related to process instance groups: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-group-controller/getProcessInstanceGroupFiltersUsingGET](http://localhost:8081/swagger-ui.html#!/process-instance-group-controller/getProcessInstanceGroupFiltersUsingGET)

**Process Instance Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/process-instance-controller**](http://localhost:8081/swagger-ui.html#/process-instance-controller)**)**

This API includes services to manage ***ProcessInstances***.

* Updating a document exchanged via a process instance group: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-controller/updateProcessInstanceUsingPATCH](http://localhost:8081/swagger-ui.html#!/process-instance-controller/updateProcessInstanceUsingPATCH)
* Cancelling a process instance: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-controller/cancelProcessInstanceUsingPOST](http://localhost:8081/swagger-ui.html#!/process-instance-controller/cancelProcessInstanceUsingPOST)
* Checking the rating status of a ***CompletedTask***, which is associated to a process instance: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/process-instance-controller/isRatedUsingGET](http://localhost:8081/swagger-ui.html#!/process-instance-controller/isRatedUsingGET)

**Contract Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/contract-controller**](http://localhost:8081/swagger-ui.html#/contract-controller)**)**

This API provides various services to manage contracts and their clauses.

* Retrieving a specific clause: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/getClauseDetailsUsingGET](http://localhost:8081/swagger-ui.html#!/contract-controller/getClauseDetailsUsingGET)
* Updating a clause: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/updateClauseUsingPUT](http://localhost:8081/swagger-ui.html#!/contract-controller/updateClauseUsingPUT)
* Retrieving clauses inside a contract: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/getClausesOfContractUsingGET](http://localhost:8081/swagger-ui.html#!/contract-controller/getClausesOfContractUsingGET)
* Deleting a clause from a contract: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/deleteClauseFromContractUsingDELETE](http://localhost:8081/swagger-ui.html#!/contract-controller/deleteClauseFromContractUsingDELETE)
* Retrieving specific types of clauses of a document including a contract: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/getClauseDetailsUsingGET\_1](http://localhost:8081/swagger-ui.html#!/contract-controller/getClauseDetailsUsingGET_1)
* Adding a data monitoring clause to a document: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/addDataMonitoringClauseToContractUsingPATCH](http://localhost:8081/swagger-ui.html#!/contract-controller/addDataMonitoringClauseToContractUsingPATCH)
* Adding a document clause to a document including a contract: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/addDocumentClauseToContractUsingPATCH](http://localhost:8081/swagger-ui.html#!/contract-controller/addDocumentClauseToContractUsingPATCH)
* Generating a contract considering all the process instances included in the process instances group containing the specified process instance: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-controller/constructContractForProcessInstancesUsingGET](http://localhost:8081/swagger-ui.html#!/contract-controller/constructContractForProcessInstancesUsingGET)

**Contract Generator Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/contract-generator-controller**](http://localhost:8081/swagger-ui.html#/contract-generator-controller)**)**

This API provides two services to generate a contract bundle and terms & conditions text.

* Generating a contract bundle for an order: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-generator-controller/generateContractUsingGET](http://localhost:8081/swagger-ui.html#!/contract-generator-controller/generateContractUsingGET)
* Generating terms and conditions for an order: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/contract-generator-controller/generateOrderTermsAndConditionsAsTextUsingGET](http://localhost:8081/swagger-ui.html#!/contract-generator-controller/generateOrderTermsAndConditionsAsTextUsingGET)

**Document Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/document-controller**](http://localhost:8081/swagger-ui.html#/document-controller)**)**

Document API provides services to retrieve documents exchanged in business processes.

* Getting a document in json format: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/document-controller/getDocumentJsonContentUsingGET](http://localhost:8081/swagger-ui.html#!/document-controller/getDocumentJsonContentUsingGET)
* Getting a document in XML format: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/document-controller/getDocumentXMLContentUsingGET](http://localhost:8081/swagger-ui.html#!/document-controller/getDocumentXMLContentUsingGET)

**Trust Service Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/trust-service-controller**](http://localhost:8081/swagger-ui.html#/trust-service-controller)**)**

Trust API provides services to manage ratings that are available after successful or unsuccessful collaboration activities.

* Getting ratings and reviews: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/trust-service-controller/listAllIndividualRatingsAndReviewsUsingGET](http://localhost:8081/swagger-ui.html#!/trust-service-controller/listAllIndividualRatingsAndReviewsUsingGET)
* Adding ratings and reviews for a party: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/trust-service-controller/createRatingAndReviewUsingPOST](http://localhost:8081/swagger-ui.html#!/trust-service-controller/createRatingAndReviewUsingPOST)
* Getting ratings summary for a party: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/trust-service-controller/getRatingsSummaryUsingGET](http://localhost:8081/swagger-ui.html#!/trust-service-controller/getRatingsSummaryUsingGET)

**EPC Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/epc-controller**](http://localhost:8081/swagger-ui.html#/epc-controller)**)**

EPC service lets users to retrieve EPC codes were used throughout the time and track and trace endpoints that for a specific product.

* Getting EPC codes that were created for a specific product: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/epc-controller/getEPCCodesBelongsToProductUsingGET](http://localhost:8081/swagger-ui.html#!/epc-controller/getEPCCodesBelongsToProductUsingGET)
* Getting track and trace related endpoints for the given EPC code: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/epc-controller/getTTDetailsUsingGET](http://localhost:8081/swagger-ui.html#!/epc-controller/getTTDetailsUsingGET)

**Statistics Controller (**[**http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/statistics-controller**](http://localhost:8081/swagger-ui.html#/statistics-controller)**)**

Statistics service offers services to extract statistics related to business processes for metrics including but not limited to trading volume, process counts, non-ordered products or average response times.

* Getting list of inactive companies: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getInactiveCompaniesUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getInactiveCompaniesUsingGET)
* Getting average response time for a party: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getAverageResponseTimeUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getAverageResponseTimeUsingGET)
* Getting average negotiation time for a party: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getAverageNegotiationTimeUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getAverageNegotiationTimeUsingGET)
* Getting overall statistics for a party: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getStatisticsUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getStatisticsUsingGET)
* Getting list of non-ordered products: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getNonOrderedProductsUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getNonOrderedProductsUsingGET)
* Getting total number of business processes: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getProcessCountUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getProcessCountUsingGET)
* Getting number of business processes requiring an action: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getActionRequiredProcessCountUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getActionRequiredProcessCountUsingGET)
* Getting number of business processes per process type: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getProcessCountBreakDownUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getProcessCountBreakDownUsingGET)
* Getting trading volume: [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#!/statistics-controller/getTradingVolumeUsingGET](http://localhost:8081/swagger-ui.html#!/statistics-controller/getTradingVolumeUsingGET)

**Business Process Service for Platform Providers**

1. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/start-controller [↑](#footnote-ref-1)
2. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/continue-controller [↑](#footnote-ref-2)
3. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/collaboration-groups-controller [↑](#footnote-ref-3)
4. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/process-instance-group-controller [↑](#footnote-ref-4)
5. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/process-instance-controller [↑](#footnote-ref-5)
6. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/statistics-controller [↑](#footnote-ref-6)
7. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/contract-controller [↑](#footnote-ref-7)
8. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/contract-generator-controller [↑](#footnote-ref-8)
9. [http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/epc-controller](http://localhost:8081/swagger-ui.html#/epc-controller) [↑](#footnote-ref-9)
10. http://nimble-staging.salzburgresearch.at/business-process/swagger-ui.html#/document-controller [↑](#footnote-ref-10)