



EESSI – RINA

BUC Technical Specifications

Architecture & Design Specifications

Table of Contents

1	Introduction	6
1.1	Scope	6
1.2	Structure of the document.....	6
1.3	Reference and Applicable documents	7
1.4	Audience	7
1.5	Definitions, Acronyms and Abbreviations	7
2	RINA BUC Example	9
3	BUC XML File Structure	11
3.1	Case.....	11
3.1.1	Case Context and Parameters	11
3.1.2	Case Actions.....	11
3.1.3	Case Documents	12
4	Case Parameters Details	19
4.1	isML	19
4.1.1	Bilateral BUCs.....	19
4.1.2	Multilateral BUCs.....	19
4.2	removeMeOnly	19
5	Document Parameters Details.....	21
5.1	hasParticipantSelection and isML	21
5.1.1	First schema - default send	22
5.1.2	Second schema - send to sublist	22
5.1.3	Third schema - send to chosen singleton	22
5.1.4	Fourth schema - reply sent to sender only.....	23
5.2	isStarter.....	23
5.2.1	Monostarter BUCs	23
5.2.2	Multistarter BUCs	23
5.3	isBulk	23
5.4	allowsAttachments	24
5.5	hasCancel, hasMultipleVersions, hasReject and hasClarify	26
5.5.1	On the sending side.....	26
5.5.2	On the receiving side	26
5.5.3	Example.....	27
5.6	recreateAfterCancel and recreateAfterSend	27
5.7	hasBusinessValidation.....	27
5.8	recreateAfterDeletion.....	28
5.9	receiveAlwaysEnabled	28
5.10	canBeSentWithoutBulk	28
6	Handlers	29
6.1	Trigger Handlers	30
6.1.1	Create Action Trigger Handler	33
6.1.2	Remove Action Trigger Handler	34
6.1.3	Suspend Action Trigger Handler	34
6.1.4	Reactivate Action Trigger Handler	35
6.1.5	Custom Action Trigger Handlers	35
6.2	Common Action Handlers	39
6.2.1	Case Action Handlers	41
6.2.2	Document Action Handlers	44
6.3	Custom Handlers.....	62
6.3.1	Send U027 Document Action handler, UB_BUC_04.	63
6.4	Condition Handlers	64
7	Events.....	66
8	Practical use of Testing Framework	67
8.1	XML Test File	67

8.2	Corresponding JAVA Class	67
8.3	Structure of the XML Test File	68
8.3.1	Header.....	69
8.3.2	Body.....	69
8.4	Example.....	73

Table of Tables

Table 1:	Table 1: Reference documents list.....	7
Table 2:	Acronyms and Abbreviations list	8

Document Control Information

Document Control	Value
Project Title	Electronic Exchange of Social Security Information (EESSI)
Document Name	EESSI – RINA - BUC Technical Specifications
Document Category	Architecture & Design Specifications
Revision	-
Component Version	-
Last Publication Date Project Milestone	18/12/2020 EESSI-2020
Document Status	Final
Sensitivity (TLP) Distribution terms	Traffic Light Protocol (TLP) = "GREEN"  The distribution of this document is done strictly in line with the Traffic Light Protocol (TLP) established by the European Commission's note AC 790/15 REV for the EESSI project documentation. In line with the note AC 790/15 REV, this document is labelled as TLP = "Green". Therefore, it can be circulated widely within the EESSI community. However, the document or the information herein may not be published or posted on the Internet, nor released outside of the EESSI community.
Connected/Embedded Files	None
Authors	European Commission, DG EMPL F5, EESSI RINA
Revised by	European Commission, DG EMPL F5, EESSI QA/QC
Approved by	European Commission, DG EMPL F5, EESSI PMO team

Document history

Project Milestone	Changes/Corrections
	Description
EESSI-2020	Initial Document

1 Introduction

1.1 Scope

This document aims at:

- Describing the content of BUC XML Files and Test XML files.
- Allowing the reader to define his/her own BUC from scratch using the model described
- In order to fulfill these goals, bringing together business concepts and more technical concepts from the BUC engine itself.
- Describing the concepts from top to bottom, starting from big picture and going into more details
- Giving lots of detailed examples with their meaning
- Inviting the user to look at complete BUC XML Files and Test XML Files (provided with the release) and to understand them, helped with the support of the present guidelines.

Note: For deeper understanding and for self-training, please check the released BUC XML Files and Test XML Files.

1.2 Structure of the document

The document is organized as follows:

- Chapter 1: Lists the scope of the document, underlined principles for the specifications, related documents, the audience, definitions, and glossary and how to navigate through the document.
- Chapter 2: Introduces BUC XML Files through a complete example.
- Chapter 3: Presents the detailed structure of BUC XML Files.
- Chapter 4: Goes into more details on the case parameters.
- Chapter 5: Goes into more details on the document parameters.
- Chapter 6: Details the different sorts of handlers that may be used in a BUC XML File to encode the flow with maximal flexibility.
- Chapter 7: Details the practical usage of the Testing Framework and the structure of Test XML Files used in order to test the BUC XML Files.

1.3 Reference and Applicable documents

#	Artefact/Document	Location
R1	EESSI Business and Technical Glossary	https://citnet.tech.ec.europa.eu/CITnet/confluence/display/EESSITCS/EESSI+Glossary+Draft

Table 1: Reference documents list

1.4 Audience

- Writers of BUC XML Files,
- Developers of new BUCs,
- Managers (support, development team) of BUCs,
- Testers of BUC XML Files,
- Writers of Test XML Files (they can be the testers).

1.5 Definitions, Acronyms and Abbreviations

Please see the EESSI Business and Technical Glossary [R1].

In addition, please find below some RINA related acronyms used in this document.

Abbreviation	Full Form
BUC	Business Use Case: is a single set of rules and steps describing a process as defined in the BUC-Guidelines, which originate from the EU regulation EC 883/2004
SED	Structured Electronic Document - is a unit being exchanged within a BUC. Can be also used only as Document.
BUC-Model	Defines the rules of the modelling of a BUC as XSD schemas
BUC-Engine	Implementation of the common module, which runs the BUC implementations.
BUC-Definition	A formal definition of the BUC processing rules (as an XML, or a BPMN diagram).
BUC-Implementation	An implementation of a particular BUC for the BUC-Engine to run, which encapsulates the BUC-Definition files together with custom implementation.
Extension Point	Is an abstract definition of specific BUC processing parts, which can be overridden to provide custom implementation.

Action	in the context of a BUC-Engine represents a single executable unit within the BUC-Definition. E.g. SET-PARTICIPATNS. An Action may be defined for a CASE context of a DOCUMENT context. In the later one, it SHOULD be accompanied by the document type.
ActionHandler	The implementation of the business logic of an Action. It follows a common interface and is registered to handle a particular set of Actions
Event	A notification, which is triggered on specific places during the Action processing.
EventListener	A listener, which defines an implementation, that is executed, when a particular Event is triggered. It can be registered to listen for a specific Event.
Trigger	Every action may trigger various operations like creating a new action or removing an existing action. This triggers are specified within the BUC model.
TriggerHandler	The implementation of a business logic to handle a trigger.
Extension Point ID	to be able to extend even a very specific functionality we need to specify an ID for the extension points. These IDs must refer to the particular context: 1. ActionHandler - ID: BUC type, Action Type, Document Type (optional) 2. Event - ID: BUC type, Action Type, location (BEFORE or AFTER action execution), action result type (SUCCESS or FAILURE) 3. ActionHandler - ID: BUC type, Action Type, Document Type (optional)
Plugin	Is an implementation of a plugin interface, that is used as a business operation within the engine, but the implementation is provided by the user of the BUC-Engine (e.g. DocumentSender)

Table 2: Acronyms and Abbreviations list

2 RINA BUC Example

This section illustrates a full size BUC XML File.

Here is an extract of the implementation of the counterparty side of the BUC H_BUC_05, version model 4.2. The file is named cp_h_buc_05.xml.

It is the file defining the stateful sequence of possible actions performed on SEDs exchanged from and to counterparties in this BUC.

The rules expressed in this file are based on the requirements for the BUC H_BUC_05 (counterparty side).

```
<?xml version="1.0"?>
<case name="H_BUC_05" version="4.2" role="CP"
      xmlns="http://ec.europa.eu/eessi/rina/buc">

  <context>
    <parameters>
      <parameter key="removeMeOnly">true</parameter> <!-- default is false -->
    </parameters>
  </context>

  <action>
    <!-- empty -->
  </action>

  <document>
    <document type="H061">
      <parameters>
        <parameter key="isStarter">true</parameter>
        <parameter key="hasClarify">true</parameter>
        <parameter key="hasMultipleVersions">true</parameter>
        <parameter key="hasCancel">true</parameter>
      </parameters>
      <triggers>
        <!-- for starter only, local close -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="CASE_LOCAL_CLOSE" documentType="ANY" />
        <!-- for starter only, X005 -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_CREATE" documentType="X005" />
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_RECEIVE" documentType="X005" />
        <!-- for starter only, X006 -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_CREATE" documentType="X006" />
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_RECEIVE" documentType="X006" />
        <!-- for starter only, X007 -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_CREATE" documentType="X007" />
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_RECEIVE" documentType="X007" />
        <!-- for starter only, X009 -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_RECEIVE" documentType="X009" />
        <!-- for starter only, business exception -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_RECEIVE" documentType="X050" />
        <!-- for starter only, change participants -->
        <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
                           actionType="DOC_RECEIVE" documentType="X100" />
        <!-- for starter only, subprocess H001 -->
```

```
<createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
    actionType="DOC_CREATE" documentType="H001" />
<createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"
    actionType="DOC_RECEIVE" documentType="H001" />
<!-- requests and replies -->
<createActionTrigger onAction="DOC_RECEIVE" onResult="SUCCESS"
    actionType="DOC_RECEIVE_REPLY" documentType="H062" />
<createActionTrigger onAction="DOC_RECEIVE" onResult="SUCCESS"
    actionType="DOC_CREATE_REPLY" documentType="H062" />

</triggers>
</document>

<document type="H062">
  <parameters>
    <parameter key="hasMultipleVersions">true</parameter>
    <parameter key="hasCancel">true</parameter>
    <parameter key="recreateAfterCancel">true</parameter>
    <parameter key="allowsAttachments">true</parameter>
    <parameter key="hasClarify">true</parameter>
  </parameters>
</document>

</document>
</case>
```

The available SED names can be recognized in the structure (H061 and H062, but also X005, X006, X007...). Each SED definition might specify parameters and triggers:

- Parameters - properties of the documents with specific meaning, e.g. `hasCancel=true` defines, that *"this document can be invalidated"*.
- Triggers - specify a function to be performed as a result of an action on the document, e.g. *"at reception of the document, create the action to add a participant (X005)"*.

The example is presented without more details. A more detailed explanation will be provided in the following sections.

3 BUC XML File Structure

This section describes the structure of a BUC xml definition file used in the BUC-Engine. It will go through every element of the xml structure.

3.1 Case

The header of the case defines the name of the BUC (H_BUC_05), its model version (4.2) and the role (counterparty).

A case contains 3 sections: <context>, <actions> and <documents>

```
<case name="H_BUC_05" version="4.2" role="CP"
      xmlns="http://ec.europa.eu/eessi/rina/buc">
  <context>
    ...
  </context>

  <actions>
    ...
  </actions>

  <documents>
    ...
  </documents>
</case>
```

3.1.1 Case Context and Parameters

Currently, the <context> section contains only the <parameters> section.

It is a list of parameters related to the case itself (case-level parameters, or case parameters)

```
<context>
  <parameters>
    ...
  </parameters>
</context>
```

The possible case parameters are: isML, removeOnlyMe, as defined in the file ECaseParameterType.java:

```
public enum ECaseParameterType {
    @XmlEnumValue("isML")
    @XmlEnumValue("removeMeOnly")
```

3.1.2 Case Actions

CURRENTLY UNUSED

As most of the case actions are common, they do not need to be specified. They are implemented in the BUC engine itself.

Anyway, there is the possibility to add some case actions if they are needed in some existing (to be updated) or future BUC.

```
<actions>
  <!-- empty -->
</actions>
```

3.1.3 Case Documents

The `<documents>` section contains one or many `<document>` sections

```
<documents>
  <document type="H061">
    ...
  </document>
  <document type="H062">
    ...
  </document>
  ...
</documents>
```

The `<document>` section contains 2 sections: `<parameters>` and `<triggers>`

These are document parameters and document triggers related to the document in question

```
<document type="H061">
  <parameters>
    ...
  </parameters>
  <triggers>
    ...
  </triggers>
</document>
```

3.1.3.1 Case Documents Parameters

3.1.3.1.1 List of parameters

The possible document parameters are:

isML, allowsAttachments, isStarter, isBulk, hasReject, hasClarify, hasBusinessValidation, hasParticipantSelection, recreateAfterDeletion, hasMultipleVersions, hasCancel, recreateAfterCancel, recreateAfterSend, receiveAlwaysEnabled, canBeSentWithoutBulk

They are defined in the file `EDocumentParameterType.java`:

```
public enum EDocumentParameterType {
    @XmlEnumValue("isML")
    @XmlEnumValue("allowsAttachments")
    @XmlEnumValue("isStarter")
    @XmlEnumValue("isBulk")
    @XmlEnumValue("hasReject")
    @XmlEnumValue("hasClarify")
    @XmlEnumValue("hasBusinessValidation")
    @XmlEnumValue("hasParticipantSelection")
    @XmlEnumValue("recreateAfterDeletion")
    @XmlEnumValue("hasMultipleVersions")
    @XmlEnumValue("hasCancel")
    @XmlEnumValue("recreateAfterCancel")
    @XmlEnumValue("recreateAfterSend")
    @XmlEnumValue("receiveAlwaysEnabled")
    @XmlEnumValue("canBeSentWithoutBulk")
```

3.1.3.1.2 Example

Here is an example of `<parameters>` section, taken from a real definition:

```
<parameters>
  <parameter key="isStarter">true</parameter>
  <parameter key="hasClarify">true</parameter>
  <parameter key="hasMultipleVersions">true</parameter>
  <parameter key="hasCancel">true</parameter>
</parameters>
```

3.1.3.1.3 Stateful behaviour

Some of the document parameters (not all) describe the stateful behaviour of RINA (the flow of successive actions): `hasReject`, `hasCancel`, `hasClarify`, `hasMultipleVersions`, `recreateAfterSend`, `recreateAfterCancel`, `recreateAfterDeletion`, `receiveAlwaysEnabled`

For example, **on the sending side**:

`<parameter key="hasClarify">true</parameter>` tells that after the document is sent, a Clarify message X012 can be received and subsequently a Reply To Clarify message X013 can be sent later.

`<parameter key="hasMultipleVersions">true</parameter>` tells that the document can be updated and resent after every send.

`<parameter key="hasCancel">true</parameter>` tells that after the document is sent, an Invalidate message X008 can be sent that will cancel the document.

And **on the receiving side**:

`<parameter key="hasClarify">true</parameter>` tells that after the document is received, a Clarify message X012 can be sent, and that a Reply To Clarify message X013 can be received later.

`<parameter key="hasMultipleVersions">true</parameter>` tells that after the document is received, updates on that document may be received.

`<parameter key="hasCancel">true</parameter>` tells that after the document is received, an Invalidate message X008 can be received that will cancel the document.

More details will be provided in a specific section (5) about Parameter details.

3.1.3.2 Case Documents Triggers

This section describes the triggers associated to documents (SEDs).

3.1.3.2.1 Stateful behaviour

Case Document Triggers are also called Action Triggers. The difference between an Action and a Trigger is, that a Trigger cannot be invoked from the outside of the BUC-Engine like an Action. A Trigger is fired always as a direct consequence of a document action. They provide another way of describing the flow of actions. Action triggers are invoked after an action has been executed, e.g after a document has been saved, sent, and before the status of the document is updated.

3.1.3.2.2 Trigger types

Action triggers may be of different types. From file Triggers.java:

```
public class Triggers {  
    @XmlElement({  
        @XmlElement(name = "createActionTrigger", type = CreateActionTrigger.class),  
        @XmlElement(name = "removeActionTrigger", type = RemoveActionTrigger.class),  
        @XmlElement(name = "suspendActionTrigger", type = SuspendActionTrigger.class),  
        @XmlElement(name = "reactivateActionTrigger", type = ReactivateActionTrigger.class)  
    })  
}
```

3.1.3.2.3 BUC extract

```
<triggers>  
  <!-- for starter only, local close -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="CASE_LOCAL_CLOSE" documentType="ANY" />  
  <!-- for starter only, X005 -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_CREATE" documentType="X005" />  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="X005" />  
  <!-- for starter only, X006 -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_CREATE" documentType="X006" />  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="X006" />  
  <!-- for starter only, X007 -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_CREATE" documentType="X007" />  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="X007" />  
  <!-- for starter only, X009 -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="X009" />  
  <!-- for starter only, business exception -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="X050" />  
  <!-- for starter only, change participants -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="X100" />  
  <!-- for starter only, subprocess H001 -->  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_CREATE" documentType="H001" />  
  <createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
    actionType="DOC_RECEIVE" documentType="H001" />  
  <!-- requests and replies -->  
  <createActionTrigger onAction="DOC_RECEIVE" onResult="SUCCESS"  
    actionType="DOC_RECEIVE_REPLY" documentType="H062" />  
  <createActionTrigger onAction="DOC_RECEIVE" onResult="SUCCESS"  
    actionType="DOC_CREATE_REPLY" documentType="H062" />  
</triggers>
```

3.1.3.2.4 Trigger format

```
<{ACTION_TRIGGER}  
  onAction="{ON_ACTION}"  
  [onParentDocumentType="{SED_TYPE_ANY_NONE}"]  
  [onlyStarter="{TRUE_FALSE}"]  
  onResult="{ON_RESULT}"
```

```
[onCondition="{STRING}"]  
  actionType="{ACTION_TYPE}"  
  documentType="{SED_TYPE_ANY}"  
  [sameParentDocument="{TRUE_FALSE}"]  
  [delay="{SETTIMER}"]  
/>
```

(**Bold** is optional)

where:

{ACTION_TRIGGER}: possible action triggers are:
| createActionTrigger: trigger to create an action
| removeActionTrigger: trigger to remove an action
| suspendActionTrigger: trigger to suspend an action
| reactivateActionTrigger: trigger to reactivate an action

{ON_ACTION} from file EactionType.java: possible actions are:
| DOC_CREATE: on creation
| DOC_UPDATE: on update
| DOC_SEND_PARTICIPANTS: on sending to subset of participants
| DOC_SEND: on sending to all involved participants
| DOC_CANCEL: on invalidate
| DOC_CANCEL_RECEIVE: on reception of invalidate
| DOC_DELETE: on deletion
| DOC_RECEIVE: on reception
| DOC_RECEIVE_REPLY: on reception of the reply
| DOC_UPDATE_RECEIVE: on reception of update
| CASE_LOCAL_CLOSE: on local close

{SED_TYPE_ANY_NONE}: sedType (f.i. P2000, A0001, U001...) | ANY | NONE

{SED_TYPE_ANY}: sedType (f.i. P2000, A0001, U001...) | ANY

{TRUE_FALSE}: true | false

{ON_RESULT}: SUCCESS | FAILURE

{ACTION_TYPE} from file EActionType.java. Possible actionType are:
| ANY: any action, used with removeActionTrigger or suspendActionTrigger
| DOC_CREATE: create
| DOC_CREATE_REPLY: create reply to
| DOC_UPDATE: update
| DOC_SEND: send
| DOC_DELETE: delete
| DOC_RECEIVE receive
| DOC_RECEIVE_REPLY: receive reply to
| CASE_ARCHIVE: archive case

{SETTIMER}: value in format "(-?[1-9][0-9]*)\\s*(s|S|m|M|h|H|d|D)"
s|S: second; m|M: minute; h|H: hour; d|D: day
examples "30m": 30 minutes, "1h": 1 hour, "45s": 45 seconds, "365d": 365 days

Optional:

- [onParentDocumentType="{SED_TYPE_NONE_ANY}"]: if omitted, equivalent to onParentDocumentType="ANY" (means: whatever the parent type)

- [onlyStarter="{TRUE_FALSE}"]: if omitted, equivalent to onlyStarter="false" (means: we don't check if the document is a starter)
- [onCondition="{STRING}"]: if omitted, equivalent to condition always true (means: activate trigger unconditionally)
- [sameParentDocument="{TRUE_FALSE}"]: if omitted, equivalent to no parent involved
- [delay="{SETTIMER}"]: if omitted, applies immediately (no delay)

The meaning of the components will be explained through examples below.

3.1.3.2.5 Examples

Fabricated example to illustrate all elements

```
<document type="P9000">
  <triggers>
    <removeActionTrigger
      onAction="DOC_SEND"
      onParentDocumentType="P8000"
      onlyStarter="false"
      onResult="SUCCESS"
      onCondition="C"
      actionType="DOC_CREATE"
      documentType="X012"
      sameParentDocument="true"
      delay="5s"
    />
  </triggers>
</document>
```

Decrypts as: when the action DOC_SEND is executed successfully on a document of type P9000 acting as a reply (child) of a document of type P8000, and not acting as a starter, and only when the condition identified by the string "C" returns true, then after a delay of 5 seconds, remove the action DOC_CREATE on the document of type X012 whose parent is the same parent (the P8000 document).

In brief, means: on successful sending of P9000, remove the Clarify from the parent P8000 under condition "C", after 5 seconds.

Actual examples from RINA

```
<document type="DA003">
  <triggers>
    <createActionTrigger
      onAction="DOC_SEND"
      onlyStarter="true"
      onResult="SUCCESS"
      actionType="CASE_LOCAL_CLOSE"
      documentType="ANY"/>
  </triggers>
</document>
```

Meaning: when DA003, as starter, is sent successfully, activate Local Close (documentType being mandatory, we put "ANY")

```
<document type="DA054">
  <triggers>
    <removeActionTrigger
      onAction="DOC_SEND"
```

```
onResult="SUCCESS"
onParentDocumentType="DA053"
actionType="DOC_CREATE"
documentType="X011"
sameParentDocument="true" />

<removeActionTrigger
onAction="DOC_SEND"
onResult="SUCCESS"
onParentDocumentType="DA053"
actionType="DOC_UPDATE"
documentType="X011"
sameParentDocument="true" />

<removeActionTrigger
onAction="DOC_SEND"
onResult="SUCCESS"
onParentDocumentType="DA053"
actionType="DOC_SEND"
documentType="X011"
sameParentDocument="true" />
</triggers>
</document>
```

Meaning: when DA054 is sent successfully, as reply of DA053, then remove the create, update and send actions on the Reject (X011) of that DA053.

```
<document type="A008">
<triggers>
<createActionTrigger
onAction="DOC_SEND"
onlyStarter="true"
onResult="SUCCESS"
actionType="DOC_CREATE"
documentType="X001"
delay="30d"/>
</triggers>
</document>
```

Meaning: when A008 is sent successfully as a starter, create the Close action after a delay of 30 days.

```
<document type="U020">
<triggers>
<removeActionTrigger
onAction="DOC_CANCEL"
onResult="SUCCESS"
actionType="DOC_CREATE"
documentType="U026" />
<removeActionTrigger
onAction="DOC_CANCEL"
onResult="SUCCESS"
actionType="DOC_UPDATE"
documentType="U026" />
<removeActionTrigger
onAction="DOC_CANCEL"
onResult="SUCCESS"
actionType="DOC_SEND"
documentType="U026" />
</triggers>
</document>
```

Meaning: when U020 is invalidated successfully, remove the create, update and send actions on U026.

```
<document type="U021">
<triggers>
<createActionTrigger
onAction="DOC_CANCEL"
```

```
onParentDocumentType="U029"  
onResult="SUCCESS"  
actionType="DOC_CREATE"  
documentType="U023"  
sameParentDocument="true" />  
</triggers>  
</document>
```

Meaning: when U021, as a reply to U029, is invalidated, create task reply U023 to the same U029.

```
<document type="DA003">  
  <triggers>  
    <createActionTrigger  
      onAction="DOC_RECEIVE"  
      onResult="SUCCESS"  
      actionType="DOC_CREATE_REPLY"  
      documentType="DA003A"/>  
  </triggers>  
</document>
```

Meaning: at successful reception of DA003, create the action to reply DA003A.

```
<document type="DA003">  
  <triggers>  
    <createActionTrigger  
      onAction="DOC_SEND"  
      onResult="SUCCESS"  
      actionType="DOC_RECEIVE_REPLY"  
      documentType="DA003A"/>  
  </triggers>  
</document>
```

Meaning: when DA003 is sent successfully, create the receiver task waiting for DA003A.

```
<document type="DA002">  
  <triggers>  
    <!-- onCondition (Section 5.1 is Yes) -->  
    <createActionTrigger  
      onAction="DOC_SEND"  
      onResult="SUCCESS"  
      onCondition="cp_aw_buc_01a_SECTION_51_YES"  
      actionType="DOC_CREATE" documentType="DA003"/>  
  </triggers>  
</document>
```

Meaning: when DA002 is sent successfully, and only under condition "cp_aw_buc_01a_SECTION_51_YES", create the action to create DA003.

```
<document type="H121">  
  <triggers>  
    <createActionTrigger  
      onParentDocumentType="NONE"  
      onAction="DOC_SEND"  
      onResult="SUCCESS"  
      actionType="DOC_CREATE"  
      documentType="H121" />  
  </triggers>  
</document>
```

Meaning: when H121 is sent successfully as a document without parent, create the action to create H121 again.

In other words: loop on H121 only if it has no parent.

4 Case Parameters Details

Case parameters have been introduced in section 3.1.1.

The case parameters `isML`, `removeMeOnly` are described in more details in the following sections. They take boolean values (true|false).

4.1 isML

This parameter relates to the number of counterparties.

Its default is true.

The value `isML=false` means that there can be only one counterparty at a time in the case (except during the process of forwarding, where the old and new counterparty may exist during a short period of time before the old counterparty is removed). This defines bilateral cases.

The value `isML=true` means that there can be more than one counterparty at a time in the case (it is possible, not mandatory). This defines multilateral cases.

4.1.1 Bilateral BUCs

As per requirements, the current bilateral BUCs are: `aw_buc_01a`, `aw_buc_01b`, `aw_buc_02`, `aw_buc_03`, `aw_buc_04a`, `aw_buc_04b`, `aw_buc_04c`, `aw_buc_05`, `aw_buc_06a`, `aw_buc_06b`, `aw_buc_06c`, `aw_buc_07a`, `aw_buc_07b`, `aw_buc_07c`, `aw_buc_07d`, `aw_buc_08`, `aw_buc_09a`, `aw_buc_09b`, `aw_buc_10a`, `aw_buc_10b`, `aw_buc_11`, `aw_buc_12`, `aw_buc_13`, `aw_buc_14`, `aw_buc_15`, `aw_buc_23`, `h_buc_04`, `h_buc_06`, `h_buc_08`, `h_buc_10`, `la_buc_04`, `m_buc_02`, `r_buc_01`, `r_buc_02`, `r_buc_03`, `r_buc_04`, `r_buc_05`, `r_buc_06`, `r_buc_07`, `s_buc_01`, `s_buc_01a`, `s_buc_02`, `s_buc_03`, `s_buc_04`, `s_buc_05`, `s_buc_06`, `s_buc_07`, `s_buc_08`, `s_buc_09`, `s_buc_11`, `s_buc_12`, `s_buc_14`, `s_buc_14a`, `s_buc_14b`, `s_buc_15`, `s_buc_17`, `s_buc_17a`, `s_buc_18`, `s_buc_19`, `s_buc_21`, `s_buc_22`, `s_buc_23`, `s_buc_24`, `ub_buc_01`, `ub_buc_02`, `ub_buc_03`, `ub_buc_04`

```
<context>
  <parameters>
    <parameter key="isML">false</parameter>
  </parameters>
</context>
```

4.1.2 Multilateral BUCs

As per requirements, the current multilateral BUCs are: `fb_buc_01`, `fb_buc_02`, `fb_buc_03`, `fb_buc_04`, `h_buc_01`, `h_buc_02a`, `h_buc_02b`, `h_buc_02c`, `h_buc_03a`, `h_buc_03b`, `h_buc_05`, `h_buc_07`, `h_buc_09`, `la_buc_01`, `la_buc_02`, `la_buc_03`, `la_buc_05`, `la_buc_06`, `m_buc_01`, `m_buc_03a`, `m_buc_03b`, `p_buc_01`, `p_buc_02`, `p_buc_03`, `p_buc_04`, `p_buc_05`, `p_buc_06`, `p_buc_07`, `p_buc_08`, `p_buc_09`, `p_buc_10`, `s_buc_18a`

```
<context>
  <parameters>
    <parameter key="isML">true</parameter>
  </parameters>
</context>
```

4.2 removeMeOnly

The removeMeOnly parameter specifies the behaviour of the "Remove Participant" feature for counterparties.

Its default value is false.

Value removeOnlyMe=true means that the counterparty, in case of use of X006 (Remove Participant), may remove himself only.

It is used only in counterparty side. It has no relevance in case owner side.

```
<context>
  <parameters>
    <parameter key="removeMeOnly">true</parameter>
  </parameters>
</context>
```

As per requirements, the current BUCs with removeOnlyMe=true are: fb_buc_01, h_buc_05, h_buc_10, p_buc_01, p_buc_02, p_buc_03

5 Document Parameters Details

Document parameters have been introduced before (section 3.1.3.1).

This section aims at presenting the parameters in more details.

As a reminder, the list of parameters is: `isML`, `allowsAttachments`, `isStarter`, `isBulk`, `hasReject`, `hasClarify`, `hasBusinessValidation`, `hasParticipantSelection`, `recreateAfterDeletion`, `hasMultipleVersions`, `hasCancel`, `recreateAfterCancel`, `recreateAfterSend`, `receiveAlwaysEnabled`, `canBeSentWithoutBulk`

We present them grouped for better understanding:

`hasParticipantSelection` and `isML` (section 5.1)

`isStarter` (section 5.2)

`isBulk` (section 5.3)

`allowsAttachments` (section 5.4)

`hasCancel`, `hasMultipleVersions`, `hasReject` and `hasClarify` (section 5.5)

`recreateAfterCancel` and `recreateAfterSend` (section 5.6)

`hasBusinessValidation` (section 5.7)

`recreateAfterDeletion` (section 5.8)

`receiveAlwaysEnabled` (section 5.9)

`canBeSentWithoutBulk` (section 5.10)

5.1 `hasParticipantSelection` and `isML`

There parameters are linked, they are presented together.

Value `hasParticipantSelection=false` (default) means that the document is sent to all available participants. In this case, a trigger meant to send the document must use `onAction="DOC_SEND"`.

Value `hasParticipantSelection=true` means that the sender selects subset of participants. In that case, a trigger meant to send the document must use `onAction="DOC_SEND_PARTICIPANTS"`. The idea is that `DOC_SEND_PARTICIPANTS` accepts a payload (subset of participants), while `DOC_SEND` does not.

Value `isML=true` (default) means that the sender can select multiple participants.

Value `isML=false`, if used together with `hasParticipantSelection=true`, means that the sender may select only one participant out of the list of possible participants.

Value `isML=false`, used with `hasParticipantSelection=false`, makes only sense if the document is a reply, and this combination of values expresses that the reply is sent only to the sender of the request.

In summary, 4 combinations are correct. The following sections describe them.

5.1.1 First schema - default send

hasParticipantSelection=false (default)

isML=true (default)

```
<document type="F022">
  <parameters>
    <!-- empty -->
  </parameters>
  <triggers>
    <createActionTrigger
      onAction="DOC_SEND"
      onResult="SUCCESS"
      actionType="DOC_RECEIVE_REPLY"
      documentType="F023"/>
  </triggers>
</document>
<!-- MEANING: send F022 to all -->
```

5.1.2 Second schema - send to sublist

hasParticipantSelection=true

isML=true (default)

```
<document type="F022">
  <parameters>
    <parameter key="hasParticipantSelection">true</parameter>
  </parameters>
  <triggers>
    <createActionTrigger
      onAction="DOC_SEND_PARTICIPANTS"
      onResult="SUCCESS"
      actionType="DOC_RECEIVE_REPLY"
      documentType="F023"/>
  </triggers>
</document>
<!-- MEANING: send F022 to chosen sublist -->
```

5.1.3 Third schema - send to chosen singleton

hasParticipantSelection=true

isML=false

```
<document type="F022">
  <parameters>
    <parameter key="hasParticipantSelection">true</parameter>
    <parameter key="isML">>false</parameter>
  </parameters>
  <triggers>
    <createActionTrigger
      onAction="DOC_SEND_PARTICIPANTS"
      onResult="SUCCESS"
      actionType="DOC_RECEIVE_REPLY"
      documentType="F023"/>
  </triggers>
</document>
<!-- MEANING: send F022 to one (only one) chosen participant -->
```

5.1.4 Fourth schema - reply sent to sender only

The first three schemas are quite obvious. The fourth one is more complex:

hasParticipantSelection=false (default)

isML=false

```
<document type="F022">
  <parameters>
    <parameter key="isML">false</parameter> <!-- default is true -->
  </parameters>
  <triggers>
    <createActionTrigger
      onAction="DOC_SEND"
      onResult="SUCCESS"
      actionType="DOC_RECEIVE_REPLY"
      documentType="F023"/>
  </triggers>
</document>
<!-- MEANING: only if F022 is a reply, because DOC_SEND has no possible payload, so we cannot tell
who the reply has to be sent to if DOC_SEND and only one participant -->
```

5.2 isStarter

This parameter defines the possible starter(s) in the BUC.

The parameter isStarter (default false) tells that the parametrized document is a starter:

- in PO, it says that the document may be sent as the first document in the BUC. Before the starter document is sent, the case remains local. After, it becomes international.
- in CP, it says that the document will instantiate the counterparty case.

There are monostarter BUCs, and multistarter BUCs.

5.2.1 Monostarter BUCs

Monostarter BUCs are BUCs where only one starter is defined. So only one document is found with isStarter=true in PO and CP xml definitions:

```
<parameter key="isStarter">true</parameter>
```

Example: p_buc_01, and many others

5.2.2 Multistarter BUCs

Multistarter BUCs are BUCs where multiple starters are defined. So several documents are found with isStarter=true in PO and CP xml definitions:

Current multistarter BUCs are: fb_buc_03, p_buc_06, s_buc_18, ub_buc_01, ub_buc_02

5.3 isBulk

The isBulk parameter expresses that the document is a bulk document, also known as a master-child document, or a document containing subdocuments.

A BUC containing at least one bulk document is called a "Bulk BUC".

```
<parameter key="isBulk">true</parameter>
```

The current Bulk BUCs are: aw_buc_05, aw15, aw_buc_23, s_buc_19, s_buc_21, s_buc_22, s_buc_23, ub_buc_04

5.4 allowsAttachments

This parameter tells if the document may have attachments included or not.

Default is false.

```
<parameter key="allowsAttachments">true</parameter>
```

The current SEDs that do **not** allow attachments are:

public enum AttachmentNotAllowed {

AW_BUC_01a_DA003A,
AW_BUC_01b_DA003A,
AW_BUC_02_DA003A,
AW_BUC_05_DA010_Master,
AW_BUC_05_DA011,
AW_BUC_05_DA012_Master,
AW_BUC_05_DA012C_Master,
AW_BUC_05_DA012R_Master,
AW_BUC_05_DA014,
AW_BUC_05_DA015,
AW_BUC_05_DA016A,
AW_BUC_05_DA018_Master,
AW_BUC_05_DA019,
AW_BUC_15_DA020_Master,
AW_BUC_15_DA021,
AW_BUC_15_DA022_Master,
AW_BUC_15_DA026_Master,
AW_BUC_23_DA073A_Master,
AW_BUC_23_DA074_Master,
LA_BUC_01_A011,
P_BUC_01_P5000,
P_BUC_01_P7000,
P_BUC_02_P5000,
P_BUC_02_P7000,
P_BUC_03_P5000,

P_BUC_03_P7000,
P_BUC_05_P5000,
P_BUC_05_P7000,
P_BUC_06_P5000,
P_BUC_06_P7000,
P_BUC_07_P11000,
P_BUC_07_P12000,
P_BUC_07_P13000,
P_BUC_08_P12000,
P_BUC_08_P13000,
P_BUC_10_P5000,
P_BUC_10_P7000,
S_BUC_11_S012,
S_BUC_19_S080_Master,
S_BUC_19_S081,
S_BUC_19_S083,
S_BUC_19_S085_Master,
S_BUC_19_S089,
S_BUC_19_S090,
S_BUC_19_S091_Master,
S_BUC_19_S092,
S_BUC_21_S100_Master,
S_BUC_21_S101,
S_BUC_21_S103,
S_BUC_21_S105_Master,
S_BUC_21_S110,
S_BUC_21_S111,
S_BUC_21_S114,
S_BUC_21_S115,
S_BUC_21_S116_Master,
S_BUC_21_S117,
S_BUC_22_S026_Master,
S_BUC_22_S027,
S_BUC_22_S028_Master,

```
S_BUC_22_S032_Master,  
S_BUC_23_S053A_Master,  
S_BUC_23_S054_Master  
}
```

Since the default is false, all SEDs **not** in this list will have to be parametrized with:

```
<parameter key="allowsAttachments">true</parameter>
```

5.5 hasCancel, hasMultipleVersions, hasReject and hasClarify

These parameters have their default value = false.

When used with value true, these parameters must be present **symmetrically** in PO and CP xml definitions. The reason of this requirement is to be found in the meaning of the parameters in sending and receiving sides.

5.5.1 On the sending side

`<parameter key="hasReject">true</parameter>` says that after the document is sent, a Reject message X011 can be received .

`<parameter key="hasClarify">true</parameter>` says that after the document is sent, a Clarify message X012 can be received and subsequently a Reply To Clarify message X013 can be sent later.

`<parameter key="hasMultipleVersions">true</parameter>` says that the document can be updated and resent after every send.

`<parameter key="hasCancel">true</parameter>` says that after the document is sent, an Invalidate message X008 can be sent that will cancel the document.

5.5.2 On the receiving side

`<parameter key="hasReject">true</parameter>` says that after the document is received, a Reject message X011 can be send.

`<parameter key="hasClarify">true</parameter>` says that after the document is received, a Clarify message X012 can be send, and that a Reply To Clarify message X013 can be received later.

`<parameter key="hasMultipleVersions">true</parameter>` says that after the document is received, updates on that document may be received.

`<parameter key="hasCancel">true</parameter>` says that after the document is received, an Invalidate message X008 can be received that will cancel the document.

5.5.3 Example

PO side:

```
<document type="U020"> <!-- sent -->
  <parameters>
    <parameter key="isStarter">true</parameter>
    <parameter key="hasMultipleVersions">true</parameter>
    <parameter key="hasCancel">true</parameter>
    <parameter key="hasReject">true</parameter>
    <parameter key="hasClarify">true</parameter>
    <parameter key="recreateAfterCancel">>false</parameter>
    <parameter key="allowsAttachments">true</parameter>
    <parameter key="isBulk">true</parameter>
  </parameters>
  ...
```

Same on CP side:

```
<document type="U020"> <!-- received -->
  <parameters>
    <parameter key="isStarter">true</parameter>
    <parameter key="hasCancel">true</parameter>
    <parameter key="hasReject">true</parameter>
    <parameter key="hasClarify">true</parameter>
    <parameter key="hasMultipleVersions">true</parameter>
    <parameter key="isBulk">true</parameter>
  </parameters>
  ...
```

5.6 recreateAfterCancel and recreateAfterSend

These parameters have their default = false.

They express that a new occurrence of the document must be created after the previous is cancelled (X008) or sent.

Logically, when the recreate after send is requested (`<parameter key="recreateAfterSend">true</parameter>`), then the recreate after cancel is not necessary (`<parameter key="recreateAfterCancel">true</parameter>`).

Why? Because before being cancelled, a document must be sent. So if the document is available again after sending it, it will for sure be available after its cancellation.

Usually, BUCs specify that `recreateAfterCancel` be true. The value of `recreateAfterSend` is more mixed (sometimes true, sometimes false, depending on BUCs and SEDs).

Previous section (5.3) illustrates a rare example where recreate after cancel is not requested (U020 in UB_BUC_04).

5.7 hasBusinessValidation

This parameter tells that a business validation is linked to the sending of the document.

Default is false.

```
<parameter key="hasBusinessValidation">>false</parameter>
```

The value `hasBusinessValidation=true` will not trigger anything by itself, it is intended to be read by (custom) handlers (see section 6), and to bring additional information to this handler that can test the value of `hasBusinessValidation` and, depending on its value

(true|false), can trigger some validation steps in branches of its algorithm (the actual business validation implementation).

In brief: in order to be useful, the hasBusinessValidation value must be evaluated by a handler that will perform some actual business validation.

Used very little, because handlers themselves are powerful enough (see section 6), and this parameter does not bring a lot of additional value.

Was introduced for historical reasons. The old BUC engine, Bonita, was using its own hasBusinessValidation parameter, which was at that time the only way to bring some validation in the processes. With the new BUC engine, flexibility has increased a lot, through the usage of handlers, that can easily extend the validation power.

5.8 recreateAfterDeletion

This parameter tells that when the document is deleted, a new occurrence must be created again.

Default is true.

```
<parameter key="recreateAfterDeletion">true</parameter>
```

Value false is used only in po_p_buc_06.xml and cp_p_buc_06.xml.

5.9 receiveAlwaysEnabled

This parameter tells that a new occurrence of the receiver task linked to the document must be created again after the previous is consumed.

Default is true.

```
<parameter key="receiveAlwaysEnabled">true</parameter>
```

Value false is used very rarely, because usually a document can be received multiple times (think about initial send, re-send after cancellation, loops, updates that may occur in any BUC).

Currently no BUC uses this parameter with value false.

5.10 canBeSentWithoutBulk

This parameter tells that a document defined as a bulk document may sometimes be sent without any subdocument selected.

Default is false.

Must be used in combination with isBulk, otherwise makes no sense at all.

```
<parameter key="isBulk">true</parameter>  
<parameter key="canBeSentWithoutBulk">true</parameter>
```

Is used with U027 in UB_BUC_04.

6 Handlers

In the execution of an action in the BUC-engine, there are three main steps.

- Internal Execution of the action.

In this step any requested validation and any update in case and document metadata is performed. Some new actions or triggers may be generated depending in the type of action, document or case context. Every internal execution of an action is implemented in an *action handler*.

- Trigger processing.

The triggers configured for result *success*, (EResultType.*SUCCESS*), configured in the BUC xml definition for this action and document type and those new added in the internal execution are executed, producing new actions, deletion of existing ones and or update of the status of certaing actions (suspend or reactivate).

For every trigger that is processed, there is a *trigger handler* that implements the actual execution steps for that given trigger.

Triggers defined in buc xml may have a condition. They are only processed if the computation of that condition returns *true*.

The computation of those conditions is implemented in *condition handlers* and they are specific for every buc.

- Firing events

In case of events are defined for this given execution they are triggered, for any possible listener registered for it.

- Deletion of current action.

All executed action is deleted after execution if no exception is thrown.

In case that an exception is thrown at any moment, any previous execution is aborted and there is a rollback in the global transaction. There is a new execution of triggers and firing events where result is *failure*, (EResultType.*FAILURE*).

There are common implementations for triggers and action handlers used by default in any buc, although there custom implementations in specific bucs as special behaviour is required by the specification.

6.1 Trigger Handlers

Triggers are defined in the buc xml definition as result to an action execution and other criteria, conditions, result of the action execution (success, failure), parent document type, etc...

Trigger handler implementations extend the base class. When a trigger must be processed, the specific implementation to be used in every case is determined by a hierarchy in the registered *trigger handler* in the BUC Engine.

In every trigger handler implementation there is an annotation with the scope where that handler implementation must be used. At the execution moment, it will search from the most specific annotation to most generic, with this hierarchy, per trigger type:

`${BUC}_${BUC_VERSION}_${FROM_ACTION}_${FROM_DOCUMENT}_${TO_ACTION}_${TO_DOCUMENT}`

In trigger handler annotation it is specified BUC (including role), version, action where trigger is defined, document type for that action, action type in the trigger, document type in the trigger.

`${BUC}_${BUC_VERSION}_${FROM_ACTION}_${FROM_DOCUMENT}`

In trigger handler annotation it is specified BUC (including role), version, action where trigger is defined, for any action and document type in the trigger.

`${BUC}_${BUC_VERSION}_${FROM_ACTION}%s_${TO_ACTION}`

In trigger handler annotation it is specified BUC (including role), version, action where trigger is defined, action type in the trigger, for any document type or no document type.

`${BUC}_${BUC_VERSION}_${FROM_ACTION}`

In trigger handler annotation it is specified BUC (including role), version, action where trigger is defined, any document type or action type in the trigger.

`${BUC}_${BUC_VERSION}`

In trigger handler annotation it is specified BUC (including role), version, for any action where trigger is defined and any action or document type in the trigger.

`${BUC_VERSION}_${TO_ACTION}_${TO_DOCUMENT}`

In trigger handler annotation it is specified version, for any BUC type, for trigger with action type and document type in the annotation.

The search hierarchy, from first to last, stops then a handler is found and that is used as implementation for that given trigger. If not found in the hierarchy, it will use default trigger handler for that trigger type handler.

For example, default annotation for Create Action Trigger would be just:

```
@OnTrigger(type = CreateActionTrigger.class)
```

Other example,

```
@OnTrigger(type = CreateActionTrigger.class)
@OnCreateActionTrigger(
    actionType = EActionType.DOC_CREATE,
    documentType = EDocumentType.X_006)
```

BUC_VERSION: default in the BUC_PROCESSES version, 4.1 or 4.2

TO_ACTION: DOC_CREATE

TO_DOCUMENT: X006

Level 6 in the hierarchy. Used for any BUC when creating an action Create X006 document.

A more specific case, in this case:

```
@OnTrigger(
    type = CreateActionTrigger.class,
    buc = "cp_p_buc_01",
    fromAction = EActionType.DOC_RECEIVE,
    fromDocument = EDocumentType.P_2000
)
@OnCreateActionTrigger(
    actionType = EActionType.DOC_RECEIVE,
    documentType = EDocumentType.P_3000)
```

There is:

BUC : cp_p_buc_01 (it includes role, PO or CP)

BUC_VERSION: default in the BUC_PROCESSES version, 4.1 or 4.2

FROM_ACTION: DOC_RECEIVE

FROM_DOCUMENT: P2000

TO_ACTION: DOC_RECEIVE

TO_DOCUMENT: P3000

This is level 1 in the hierarchy...

Trigger Handler attributes:

On Action: Action that activates the trigger.

On Result: Result of action that activates the trigger, success, failure.

Only Starter: Trigger is only processed if referring to starter document in the case.

Document Id: Document id in the context of the trigger.

On Condition: If condition is present, trigger is processed if condition is computed as true.

On Parent Document Type: Trigger is applied to documents having same parent document type as current document Id.

Same Parent Document Id: Trigger is applied to documents having same parent document (ID), as the current document Id.

When a trigger is processed, the handler is called with the current execution context. Every trigger processing executes the function *onTrigger*.

```
public void onTrigger(Action action, Trigger trigger, ExecutionContext executionContext)
```

Action handler are implemented per **trigger action type**. It uses a corresponding annotation in the implementation class, example:

`@MetaInfServices(TriggerHandler.class)` – To remark it is a Trigger Action Handler.

`@OnTrigger(type = CreateActionTrigger.class)`

6.1.1 Create Action Trigger Handler

```
@MetaInfServices(TriggerHandler.class)
@OnTrigger(type = CreateActionTrigger.class)
```

This trigger handler creates new actions, according to the trigger attributes. It adds these specific attributes.

Action Type: the type of the new created action.

Document Type: type of the document in the action. Null, *ANY* or *NONE* for case actions.

Delay: delay in the availability of the action.

6.1.1.1 Case Actions

Document type is null, ANY or NONE: a case action is created of action type.

6.1.1.2 Document Actions

6.1.1.2.1 Receive document actions

EActionType.*DOC_RECEIVE*

EActionType.*DOC_RECEIVE_REPLY*

Receive document actions do **not** have document id. Just document type. If trigger for action type *receive reply* then parent document id is added to the new action. If any message for that document type (and parent if related set id) is received, the action will be executed.

EActionType.*DOC_UPDATE_RECEIVE*

Receive update document does have document id, and it is executed for that document id in case of new message is received.

6.1.1.2.2 Create Document actions

EActionType.*DOC_CREATE*

EActionType.*DOC_CREATE_REPLY*

EActionType.*DOC_CREATE_CHILD*

In case of document creation trigger, if no document id is provided, a new document metadata with new document id is created. In case of *create reply* or *create child* the parent document id and type is also provided. Difference is if new created document is as a reply of received document or as a follow up of a sent document, i.e., a cancellation. The status of this new document is EMPTY.

6.1.2 Remove Action Trigger Handler

```
@MetaInfServices (TriggerHandler.class)  
@OnTrigger(type = RemoveActionTrigger.class)
```

Processing this trigger handler removes existing actions, according to the trigger attributes. It adds these specific attributes.

Action Type: the type of the new created action.

Document Type: type of the document in the action. Null, *ANY* or *NONE* for case actions.

Parent Document Type: type of the parent document in the action. It may be Null, *ANY* or *NONE*.

It deletes all actions that meet the criteria specified in trigger attributes: action type, document type, parent document type, document id, same parent document id, etc...

6.1.3 Suspend Action Trigger Handler

```
@MetaInfServices (TriggerHandler.class)  
@OnTrigger(type = SuspendActionTrigger.class)
```

Processing this trigger handler sets to DISABLED status all actions, according to the trigger attributes. It adds these specific attributes.

Action Type: the type of the new created action.

Document Type: type of the document in the action. Null, *ANY* or *NONE* for case actions.

Parent Document Type: type of the parent document in the action. It may be Null, *ANY* or *NONE*.

It sets to status DISABLED all actions that meet the criteria specified in trigger attributes: action type, document type, parent document type, document id, same parent document id, etc...

6.1.4 Reactivate Action Trigger Handler

```
@MetaInfServices (TriggerHandler.class)
@OnTrigger (type = ReactivateActionTrigger.class)
```

Processing this trigger handler sets to ACTIVE status all actions, according to the trigger attributes. It adds these specific attributes.

Action Type: the type of the new created action.

Document Type: type of the document in the action. Null, ANY or NONE for case actions.

Parent Document Type: type of the parent document in the action. It may be Null, ANY or NONE.

It sets to status ACTIVE all actions that meet the criteria specified in trigger attributes: action type, document type, parent document type, document id, same parent document id, etc...

6.1.5 Custom Action Trigger Handlers

There may be custom implementation of any of the previous action trigger handlers that are processes replacing the default behaviour in the context they are defined.

There is a custom trigger handler in the *common* project, in order to implement a custom behaviour during the creation of Create action for document X006. In other BUCs there are other triggers that may require information from that case context in order to fulfil case requirements.

Here some examples of them. There are more in different BUC implementations.

6.1.5.1 Create X006 Create Action Trigger Handler

```
@MetaInfServices (TriggerHandler.class)
@OnTrigger (type = CreateActionTrigger.class)
@OnCreateActionTrigger (
    actionType = EActionType.DOC_CREATE,
    documentType = EDocumentType.X_006)
```

Custom implementation of creation of action Create for document type X006.

When processed, this trigger will create a document metadata for new X006 in status EMPTY, including all existing case participants. If number of participants in the case is 2, the action will be created in status SUSPENDED, otherwise it will be in status ACTIVE.

6.1.5.2 Remove X008 Actions Sending U024 Trigger Handler

This trigger is in BUC UB_BUC_04.

BUC xml definition, in document type U024:

```
<!-- CUSTOM REQUESTED: at DOC_SEND of U024, remove DOC_CREATE, DOC_UPDATE and DOC_SEND on X008 from  
all previously sent U021 and U023 -->  
<removeActionTrigger onAction="DOC_SEND" onResult="SUCCESS" actionType="ANY" documentType="X008"  
> <!-- This is the custom trigger (SendU024TriggerHandler.java)-->  
  
@OnTrigger(  
    type = RemoveActionTrigger.class,  
    buc = "cp_ub_buc_04",  
    fromAction = EActionType.DOC_SEND,  
    fromDocument = EDocumentType.U_024)  
@OnRemoveActionTrigger(  
    actionType = EActionType.ANY,  
    documentType = EDocumentType.X_008)
```

When U024 document is sent, the trigger is remove X008 actions. Custom implementation performs deletion of X008 actions, create, update, send for cancellation document, for document types U021 and U023.

P3000 document type triggers.

P3000 documents require special treatment, as they are specific per country that receives it. So, it must be created of specific type for every receiving country institution. These specific document types are encoded as P3000_*CountryCode*, being *CountryCode* the international country code, i.e., FR for France, EL for Greece, FI, for Finland, etc.. so, corresponding P3000 document types would be P3000_FR, P3000_EL, P3000_FI, etc...

In the BUC xml definitions, its set only as P3000 document, and custom trigger handles manage that *conversion* for document type.

This special treatment is required in order to create *Create* actions and *Receive* actions for P3000 documents.

P3000 documents are present in several Pension BUCs, P_BUC_01, P_BUC_02, P_BUC_03, P_BUC_10 . Where triggers like the ones below can be found.

6.1.5.3 Create P3000 documents Create Action

```
@OnTrigger(  
    type = CreateActionTrigger.class,  
    buc = "po_p_buc_01",  
    fromAction = EActionType.DOC_SEND,  
    fromDocument = EDocumentType.P_2000  
)  
@OnCreateActionTrigger(  
    actionType = EActionType.DOC_CREATE,  
    documentType = EDocumentType.P_3000)
```

This trigger creates P3000 Create actions for every different country in the counter parties in the case, when P2000 document is sent. In the BUC xml definition it has the attribute *onlyStarter=true*, so, it is only executed the first time.

```
<createActionTrigger      onAction="DOC_SEND"      onlyStarter="true"      onResult="SUCCESS"  
actionType="DOC_CREATE"documentType="P3000" />
```

It gets all different country codes in counter parties, i.e., BE, ES, PT. and it creates one different Create P3000 document action per country, so, in this case, Create P3000_BE, Create P3000_ES, Create P3000_PT.

The receiver of every P3000 is/are only the institutions in the specific country of every P3000.

6.1.5.4 Create P3000 document Receive Action

```
@OnTrigger(  
    type = CreateActionTrigger.class,  
    buc = "cp_p_buc_01",  
    fromAction = EActionType.DOC_RECEIVE,  
    fromDocument = EDocumentType.P_2000  
)  
@OnCreateActionTrigger(  
    actionType = EActionType.DOC_RECEIVE,  
    documentType = EDocumentType.P_3000)
```

In the BUC xml definition, it is set that it must create a DOC_RECEIVE action for document type P_3000 when a P2000 document is received. In the BUC xml definition it has the attribute *onlyStarter=true*, so, it is only executed the first time.

```
<createActionTrigger onAction="DOC_RECEIVE" onlyStarter="true" onResult="SUCCESS"  
actionType="DOC_RECEIVE" documentType="P3000" />
```

However, in case of P3000 documents, this document type is not expected to be received, instead, the receiver expects receiving P3000_CountryCode, being *CountryCode* the country code of the receiver.

Processing this trigger creates an action DOC_RECEIVE for document type P3000_CountryCode,

6.2 Common Action Handlers

Action handlers implement certain execution when an action of the given type and for a given document type is executed. Typically, they produce some updates on case and document metadata and they update the list of actions available in a case, creating new ones and/or removing or disabling some of the existing ones.

They extend the base class `AbstractActionHandler` and must implement the abstract function `executeInternal` and, if required override the function `validatebusinessRules`.

There are a set of action handlers that are common to most BUC and document types, but there are exceptions that have its own implementation in the corresponding BUC.

Action handler are implemented per **action type** and **document type**. It uses a corresponding annotation in the implementation class, example:

```
@MetaInfServices(ActionHandler.class) - To remark it is an Action Handler.  
@OnAction(type = EActionType.CASE_SET_PARTICIPANTS)  
@OnAction(type = EActionType.DOC_CREATE)
```

When an action must be processed, the specific implementation to be used in every case is determined by a hierarchy in the register of action *handlers* in the BUC Engine.

In every action handler implementation there is an annotation with the scope where that handler implementation must be used. At the execution moment, it will search from the most specific annotation to most generic, with this hierarchy

`_${BUC}_${BUC_VERSION}_${ACTION}_${DOCUMENT}`

BUC, including role, version, action type and document type

`_${BUC}_${BUC_VERSION}_${ACTION}`

BUC, including role, version and action type

`_${BUC_VERSION}_${ACTION}_${DOCUMENT}`

Version, action type and document type

`{BUC_VERSION}_${ACTION}`

Version, action type.

When an implementation of the handler is found, following that hierarchy, this is executed. BUC Version is default in the BUC_Processes, 4.1, 4.2, not specified in annotation.

Examples,

```
@OnAction(type = EActionType.CASE_CREATE) level 4 in the hierarchy.
```

```
@OnAction(buc = "cp_ub_buc_04", type = EActionType.DOC_UPDATE_RECEIVE, document = EDocumentType.U_020) Level 1 in the hierarchy, BUC, version, action type and document type.
```

Action handlers implement the abstract class `AbstractActionHandler<T extends Object>` completing the abstract methods and, if required, overriding the inheritable methods. Then abstract class implements the interface `ActionHandler<T extends Object>` with to functions, that are the basic steps in all execution:

- Validation of Business Rules

Performs validation of business rules given the context of the execution. If rule is not validated a Business Validation Exception is thrown with given message. If no failure execution continues to next step.

Not all action handlers have business validation, and there are some specific implementations of action handlers in several buc types in order to add custom validations.

- Internal execution.

An action handler must implement the function *executeInternal* that calls the updates on metadata and the computation of new possible actions, creating, deleting or updating actions directly or creating new action triggers to be processed, besides the ones defined in the xml buc definition.

```
protected abstract void executeInternal(Action action, T payload,  
ExecutionContext executionContext, CaseParticipantDO whoAmI) throws Exception;
```

After execution of every action, it is deleted. If required, another action of same type (for same document) is recreated.

Actions are created, removed, suspended, or reactivated either directly, calling *actionServices*, or by adding new action triggers to the action in execution that will perform the creation, update, or deletion of actions in the case.

When the execution of the handler finishes, any new trigger generated during the execution of handlers and any other trigger specified in the buc xml configuration related to this document type and action are processed.

6.2.1 Case Action Handlers

Case action handlers manage actions that are not linked to a specific document or document type.

Case actions are normally not defined in buc xml definition, as they are generated by default execution of action and trigger handlers.

6.2.1.1 Create Case Action Handler

```
@OnAction(type = EActionType.CASE_CREATE)
```

This action is executed by default when a case is created as Process Owner not forwarded.

Execution:

It updates case metadata with buc definition parameters isML (is multilateral case), and removeMeOnly (if participant can only remove itself).

Actions:

- Create trigger Create SetParticipants action
- Create trigger Create DeleteCase action

6.2.1.2 Set Participants Action Handler

```
@OnAction(type = EActionType.CASE_SET_PARTICIPANTS)
```

This action sets the initial participants in the case. Before sending any document.

Execution:

It updates case metadata with buc definition parameters isML (is multilateral case), and removeMeOnly (if participant can only remove itself).

Actions:

- Create trigger Create UpdateParticipants action
- Create trigger Create Document for starter document/s in the case.
- Create Initial Document Metadata for this/these starter document/s.

Validation:

Validation rules for participants, 1 sender PO, not repeated participants.

6.2.1.3 Update Participants Action Handler

```
@OnAction(type = EActionType.CASE_UPDATE_PARTICIPANTS)
```

This action allows updating initial participants in the case before any document is sent.

Execution:

It updates case metadata with buc definition parameters isML (is multilateral case), and removeMeOnly (if participant can only remove itself).

Actions:

- Create trigger Create UpdateParticipants action

Validation:

Validation rules for participants, 1 sender PO, not repeated participants.

6.2.1.4 Delete Case Action Handler

```
@OnAction(type = EActionType.CASE_DELETE)
```

This action deletes current case and all related data. Currently only possible if no document has been sent.

Execution:

Deletion of case metadata, related documents, comments, etc...

Actions:

No new actions are generated

6.2.1.5 Local Close Action Handler

```
@OnAction(type = EActionType.CASE_LOCAL_CLOSE)
```

This action closes the case.

Execution:

- Case is set to status CLOSED
- All actions are set to status SUSPENDED, except actions of type: DOC_READ, DOC_READ_PARTICIPANTS, DOC_RECEIVE and DOC_UPDATE_RECEIVE.

Actions:

- Create trigger Create action CASE_LOCAL_REOPEN
- Create trigger Create action CASE_ARCHIVE

6.2.1.6 Local Reopen Action Handler

```
@OnAction(type = EActionType.CASE_LOCAL_REOPEN)
```

This action reopens the case

Execution:

- Case is set to status OPEN
- All actions are set to the status before execution of LOCAL CLOSE.

Actions:

- Create trigger Create action CASE_LOCAL_CLOSE
- Create trigger Remove action CASE_ARCHIVE

6.2.1.7 Archive Case Action Handler

```
@OnAction(type = EActionType.CASE_ARCHIVE)
```

This action archives a case.

Execution:

- Case is set to status ARCHIVED
- All *read* and *receive* actions and local reopen action are suspended.

Actions:

- Create trigger Create action CASE_RESTORE
- Suspend Case Local Reopen.

6.2.1.8 Restore Case Action Handler

```
@OnAction(type = EActionType.RESTORE_CASE)
```

This action restores an archived case to status closed

Execution:

- Case is set to status CLOSED
- All *read* and *receive* actions and local reopen action are activated.

Actions:

- Create trigger Create action CASE_ARCHIVE
- Reactivate Case Local Reopen.
- Reactivate all read and receive actions

6.2.1.9 Request Approval Action Handler

```
@OnAction(type = EActionType.REQUEST_APPROVAL)
```

This action generates a notification to case supervisor or authorised clerk to request sending document approval (an unauthorised clerk to be authorised in the case).

Execution:

No execution inside BUC-Engine

Actions:

No action

6.2.2 Document Action Handlers

Actions linked to a specific document according to the document type.

There are common handlers used for actions in documents, then there are some specific handlers for certain action and certain document types. These override the default ones.

Moreover, in some specific bucs, there may be other custom handlers that override handler for specific actions, document types in that buc type.

Common handlers

6.2.2.1 Read Document/Subdocument/Participants Action Handler

`DOC_READ, DOC_READ_PARTICIPANTS, READ_SUBDOCUMENT`

These handlers are not implemented, as BUC Engine is not called when these are executed. These actions exist just to allow reading a document, then the action exists and is enabled. As these actions are not executed, they are not deleted either by reading the document, subdocument, etc., and they do not need to be recreated.

6.2.2.2 Create Document Action Handler

`@OnAction(type = EActionType.DOC_CREATE)`

This action is executed when document is saved the first time.

Execution:

Set document to status NEW

Actions:

All actions are related to current document

- Create trigger Create Update Document action
- Create trigger Create Delete Document action

If document allows attachments:

- a) Create trigger Create Add Attachment action
- b) Create trigger Create Remove Attachment action

If document is not bulk or is bulk and can be sent without subdocuments:

- If has not Participant selection:
 - c) Create trigger Create Delete Document action
- If has Participant selection
 - d) Create trigger Create Delete Document action
 -
- If Document is bulk:
 - e) Create trigger Create Add Subdocument action
 - f) Create trigger Create Import Subdocument action

6.2.2.3 Update Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE)
```

This action is executed when document is updated.

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- Create trigger Create Update Document action
- If Status is SENT
 - a) Create trigger Create Send Document action

6.2.2.4 Delete Document Action Handler

```
@OnAction(type = EActionType.DOC_DELETE)
```

This action deletes the current document

Execution:

Set document to status NEW

Actions:

No actions are generated

6.2.2.5 Send Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND)
```

Send document action handler, send participants action handler and other sending document handlers extend the common class `GenericSendDocumentActionHandler<T>` where the implementation to generate conversation messages and generic setting up new actions is implemented.

In some cases there are specific implementations of some of the functions in the generic class, according to specific desired behaviour.

In this action specific execution is:

Execution:

Set document in status SENT.

Create a message for every receiver, add to conversation and call sender plugin.

Process sensitive data if not committed.

Actions:

- Create trigger Remove Delete Document action
- Create trigger Remove Update Document action

If document is bulk:

- Create trigger Remove Add Subdocument action
- Create trigger Remove Remove Document action

If Document has multiple versions

- Create trigger Create Update Document action

If Document has **not** multiple versions

- Create trigger Remove Update Subdocument action

If document has attachments:

- Create trigger Remove Add Attachment action
- Create trigger Remove Remove Attachment action

If document has Cancel

- Create trigger Create X008 Document action

If document has Clarify

- Create trigger Receive X012 Document action

If document has Reject

- Create trigger Receive X011 Document action

If document is the first sent document in the case:

- Create trigger Remove Update Participants action
- Create trigger Remove Delete Case action
- Create trigger Create Read Participants action

Validation:

If document is bulk, check if it can be sent without subdocuments. If false, check that the document contains, at least, one subdocument.

6.2.2.6 Send Participants Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND_PARTICIPANTS)
```

This handler extends the common class `GenericSendDocumentActionHandler<T>` where the implementation to generate conversation messages and generic setting up new actions is implemented.

It has exactly same behaviour and validations as previous one, adding validation for selected participants.

Validation:

In order to send the document it checks:

- At least 1 sender and 1 receiver
- They are valid ones.
- Sender is the case owner.
- If case is bilateral, only 1 receiver allowed.
- No participants are repeated

6.2.2.7 Receive Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE)
```

This handler manages the reception of a message of a SED document into the case. Document content is stored before calling this action.

It updates the document metadata in the case and also the case metadata if sensible person flag is enabled.

They it behaves differently if it is a starter document, normal document in a case, or a forwarded document.

Normal Reception:

- Create trigger create Read Action
- Create trigger create receive action for same document type and parent document.
- If document has Cancel, create Trigger Receive X008 related document.
- Process sensitive data if not committed
- If document has Clarify create trigger Create X012 related document.
- If document has Reject create trigger Create X011 related document
- If document has multiple version, create Trigger Receive Update for this document.
- If document is starter, first in the case, set up participants in case metadata. Set as valid starter and as starter document type.
- If document is of type of starter document, if starter document in the case is cancelled, set this new received one as starter document in the case. Reactivate Add Participant and Forward Participant actions if suspended.

All DOC_RECEIVE triggers in the xml document are processed after the handler execution is done.

Reception of forwarded document:

In this case, the received document must be treated as if owned and sent by receiving case. As it will manage the document from that moment on.

- Remove any possible existing actions for this document type and same parent document id (also if no parent document).
- Create Read Action
- Process sensitive data if not committed

- If document has multiple versions, create update action
- If document has recreate after send, Create Document of same Type trigger is created.
- If document has cancel, create trigger Create X008 related document
- If document has Clarify create trigger Receive X012 related document.
- If document has Reject create trigger Receive X011 related document
- If document is starter, first in the case, set up participants in case metadata, remove sender and set current participant as sender and Case Owner, START_FORWARD. Set as valid starter and as starter document type.
- All DOC_RECEIVE triggers of the document type are removed from the action, and replaced by DOC_SEND and DOC_SEND_PARTICIPANT triggers

6.2.2.8 Receive Document Update Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE_RECEIVE)
```

This handler manages the reception of a message that is an update of an already received SED document into the case. Document content is stored before calling this action.

These documents must "have" multiple versions.

It updates the document metadata in the case and also the case metadata if sensible person flag is enabled.

They it behaves differently if it is a starter document, normal document in a case, or a forwarded document.

Execution:

- Process sensitive data if present in the message

Actions:

- Create trigger Create Receive Update Delete Document action

6.2.2.9 Add Attachment Action Handler

```
@OnAction(type = EActionType.ADD_ATTACHMENT)
```

This action is executed when an attachment is added to the document

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- Create trigger Create Add Attachment action
 - If Status is SENT
 - a) Create trigger Create Send Document action

6.2.2.10 Add Attachment Action Handler

```
@OnAction(type = EActionType.REMOVE_ATTACHMENT)
```

This action is executed when an attachment is removed from the document

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- Create trigger Create Remove Attachment action
- If Status is SENT
 - a) Create trigger Create Send Document action

6.2.2.11 Add Subdocument Action Handler

```
@OnAction(type = EActionType.ADD_SUBDOCUMENT)
```

This action is executed when a subdocument is added to the document

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- Create trigger Create Add Subdocument action
- If first subdocument added:
 - a) Create Trigger Create Update Subdocument Action
 - b) Create Trigger Create Remove Subdocument Action
 - c) Create Trigger Create Read Subdocument Action
- If Status is SENT
 - d) Create trigger Create Send Document action

6.2.2.12 Remove Subdocument Action Handler

```
@OnAction(type = EActionType.REMOVE_SUBDOCUMENT)
```

This action is executed when a subdocument is deleted from the document

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- If there is still more than 1 subdocument in the document
 - a) Create trigger Create Remove Subdocument action
- If no more subdocuments in the document
 - a) Create Trigger Remove Update Subdocument Action
 - b) Create Trigger Remove Read Subdocument Action
- If Status is SENT
 - c) Create trigger Create Send Document action

6.2.2.13 Import Subdocuments Action Handler

```
@OnAction(type = EActionType.IMPORT_SUBDOCUMENT)
```

This action is executed when a subdocument is added to the document

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- Create trigger Create Add Subdocument action
- If first subdocument added:
 - a) Create Trigger Create Update Subdocument Action
 - b) Create Trigger Create Remove Subdocument Action
 - c) Create Trigger Create Read Subdocument Action
- If Status is SENT
 - d) Create trigger Create Send Document action

6.2.2.14 Update Subdocument Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE)
```

This action is executed when a subdocument is deleted from the document

Execution:

If document is in status SENT, it sets to status ACTIVE

Actions:

All actions are related to current document

- Create Trigger Create Update subdocument action

Specific handlers for document type, common to all BUC types

For further clarification about these document use cases, please refer to corresponding business specifications.

Handlers for X008 document. Cancellation document.

6.2.2.15 Send X008 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_008)
```

Sends the document X008, Cancellation of the parent document.

Execution:

Set status of document X008 to SENT. Set status of parent document as CANCELLED.

Actions:

- Remove all X008 edition actions, DOC_UPDATE, DOC_DELETE.

- Cancel all edition or send actions for parent cancelled document or its other child documents.
- Add all triggers for action type DOC_CANCEL of parent document in triggers to be processed.
- If parent document is starter document, set valid starter as false and disable add/forward participant actions (create/update/send X005 and X007 documents).

6.2.2.16 Receive X008 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_008)
```

Reception of the document X008, Cancellation of the parent document.

Execution:

Set status of document X008 to Received. Set status of parent document as CANCELLED.

Actions:

- Create trigger create read document action for X008.
- Cancel all edition or send actions for parent cancelled document or its other child documents.
- Add all triggers for action type DOC_CANCEL of parent document in triggers to be processed.
- If parent document is starter document, set valid starter as false and disable add/forward participant actions (create/update/send X005 and X007 documents).

Handlers for X011, X012, X013, Reject, Clarify, Reply Clarify

6.2.2.17 Receive X011 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_011)
```

Manages reception of document X011, Reject of the parent document.

Actions:

- Create trigger Create Read X011 action
- Recreate DOC_RECEIVE for X011 action in same parent document id.

6.2.2.18 Send X012 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_012)
```

Sends the document X012, Clarify of the parent document.

Execution:

Set status of document X012 to SENT.

Actions:

- Create trigger Create Reply Clarify for X012 parent document action.
- Remove all X012 edition actions.

6.2.2.19 Receive X012 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_012)
```

Manages reception of document X012, Clarify of the parent document.

Actions:

- Create trigger Create Read X012 action
- Create trigger Create Reply X013 for X012 document action.
- Recreate DOC_RECEIVE for X012 action in same parent document id.

6.2.2.20 Receive X013 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_013)
```

Manages reception of document X013, Reply Clarify of the parent document X012.

Actions:

- Create trigger Create Read X013 action

Handlers for X005, X006, X007, Add, Remove, Forward participant

Add Participant.

6.2.2.21 Create X005 Document Action Handler

```
@OnAction(type = EActionType.DOC_CREATE, document = EDocumentType.X_005)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

Actions:

- Create trigger Create Update Document action
- Create trigger Create Read Document action
- Create trigger Create Delete Document action
- Create trigger Create Send Document action

Validations:

- New added participant is not null
- New added participant is not existing participant in the case
- If case owner is counter party, new added participant must be in the same country.

6.2.2.22 Update X005 Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE, document = EDocumentType.X_005)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

Actions:

- Create trigger Create Update Document action

Validations:

- New added participant is not null
- New added participant is not existing participant in the case
- If case owner is counter party, new added participant must be in the same country.

6.2.2.23 Send X005 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_005)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

- New participant is added to case as a counter party participant.
- X005 document is sent to all existing participants in the case.
- All sent documents in the case to be resent to new participant are sent to it.
- New participant is added to all documents that have been or may be sent to the new participant.

Actions:

- Create trigger create X005 document action.
- Remove all X005 edition actions
- If any Create, update, Send action for X006 document is in status SUSPENDED, it is reactivated to status ACTIVE

Validations:

- Same validations as in add or update X005

6.2.2.24 Receive X005 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_005)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

- New participant is added to case as a counter party participant.
- All sent documents in the case to be resent to new participant are sent to it.

- New participant is added to all documents that have been or may be sent to the new participant.

Actions:

- Create trigger read X005 document action.
- Create trigger receive X005 document action.
- If any Create, update, Send action for X006 document is in status SUSPENDED, it is reactivated to status ACTIVE

Remove Participant.

6.2.2.25 Create X006 Document Action Handler

```
@OnAction(type = EActionType.DOC_CREATE, document = EDocumentType.X_006)
```

Actions:

- Create trigger Create Update Document action
- Create trigger Create Read Document action
- Create trigger Create Delete Document action
- Create trigger Create Send Document action

Validations:

- Remove added participant is existing counter party participant in the case
If case has parameter **REMOVE_ME_ONLY** as true, removed participant must be the creator of the X006, not Process Owner.

6.2.2.26 Update X006 Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE, document = EDocumentType.X_006)
```

Actions:

- Create trigger Create Update Document action

Validations:

- Remove added participant is existing counter party participant in the case
If case has parameter **REMOVE_ME_ONLY** as true, removed participant must be the creator of the X006, not Process Owner.

6.2.2.27 Send X006 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_006)
```

Execution:

An existing participant, counter party, is removed from the case.

- Send X006 document to all case participants.

If Removed participant is the sender:

- Set case to REMOVED status.
- Disable all non READ and all RECEIVE actions.
- Create Trigger Create Archive Action.

If removed participant is not the sender:

- Delete removed participant from all document conversations.
Delete removed participant from case participants
- New participant is added to all documents that have been or may be sent to the new participant.

Actions:

- Remove all current X006 edition actions
- *Create Action Create Trigger Create X006 document.*
If there are only 2 participants remaining in the case this action will be in status SUSPENDED.

Validations:

- Remove added participant is existing counter party participant in the case
If case *has parameter REMOVE_ME_ONLY* as true, removed participant must be the creator of the X006, not Process Owner.

6.2.2.28 Receive X006 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_006)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

If Removed participant is the receiver:

- Set case to REMOVED status.
- Disable all non READ and all RECEIVE actions.
- Create Trigger Create Archive Action.

If removed participant is not the receiver:

- Delete removed participant from all document conversations.
Delete removed participant from case participants
- New participant is added to all documents that have been or may be sent to the new participant.

Actions:

- Create trigger read X006 document action.
- Create trigger receive X006 document action.

Forward Participant.

6.2.2.29 Create X007 Document Action Handler

```
@OnAction(type = EActionType.DOC_CREATE, document = EDocumentType.X_007)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

Actions:

- Create trigger Create Update Document action
- Create trigger Create Read Document action
- Create trigger Create Delete Document action
- Create trigger Create Send Document action

Validations:

- New added participant is not null
- New added participant is not existing participant in the case
- New participant must be in the same country as the sender
- Removed participant must be the case owner, current tenant

6.2.2.30 Update X007 Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE, document = EDocumentType.X_007)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

Actions:

- Create trigger Create Update Document action

Validations:

- New added participant is not null
- New added participant is not existing participant in the case
- New participant must be in the same country as the sender.
- Removed participant must be the case owner, current tenant

6.2.2.31 Send X005 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_007)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

- Set case to REMOVED status.

- Disable all non READ and all RECEIVE actions.
- X007 document is sent to all existing participants in the case, including new participant.
- All sent documents in the case to be resent to new forwarded participant are sent to it.

Actions:

- Create Trigger Create Archive Action.

Validations:

- Same validations as in add or update X007

6.2.2.32 Receive X007 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_007)
```

Execution:

Add new chosen participant to the conversation metadata of the document in order to be sent also to it, and to all other existing case participants.

- New participant is added to case as a counter party participant.
- Removed participant is removed from the case and from all document metadata
- All sent documents in the case to be resent to new participant are sent to it.
- New participant is added to all documents that have been or may be sent to the new participant.

Actions:

- Create trigger read X007 document action.

Handlers for Global Close, Global Reopen of a Case

Refer to corresponding Global Case, Reopen use case definition.

6.2.2.33 Create X001 Document Action Handler

```
@OnAction(type = EActionType.DOC_CREATE, document = EDocumentType.X_001)
```

Execution:

Create X001 document. Standard create execution, only content validation is added.

Actions:

- Create trigger Create Update Document action
- Create trigger Create Read Document action
- Create trigger Create Delete Document action
- Create trigger Create Send Document action

Validations:

- If content field `reasonForClosing` is OTHERS, value 99, then field `pleaseProvideMoreDetailsIf990therSelected` must not be empty

6.2.2.34 Update X001 Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE, document = EDocumentType.X_001)
```

Execution:

Update X001 document. Standard create execution, only content validation is added.

Actions:

- Create trigger Create Update Document action

Validations:

- If content field `reasonForClosing` is OTHERS, value 99, then field `pleaseProvideMoreDetailsIf990therSelected` must not be empty

6.2.2.35 Send X001 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_001)
```

Execution:

Sending X001 document. Case is set to status CLOSED, all edition actions are set to disabled.

Actions:

- Sent document ID is stored in case context, prefill, with key `lastProcessedX001DocumentId` in order to recreate X002 document, if needed, if reopening is rejected.
- Create trigger Create Archive Case action
- Create trigger Create X001 document action
- Create Triggers Suspend X001 document edition actions. (for future reopening). If there is REOPEN, it will be set in the buc xml definition, creating X002 document trigger after X001 is sent. It is not in the custom handler.

Validations:

- If content field `reasonForClosing` is OTHERS, value 99, then field `pleaseProvideMoreDetailsIf990therSelected` must not be empty

6.2.2.36 Receive X001 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_001)
```

Execution:

Receiving X001 document. Case is set to status CLOSED, all edition actions are set to disabled.

Actions:

- Sent document ID is stored in case context, prefill, with key `lastProcessedX001DocumentId` in order to recreate X002 document, if needed, if reopening is rejected.
- Create trigger Archive Case action
- Create trigger Read X001 document action
- Create Triggers Suspend X001 document edition actions. (for future reopening). If there is REOPEN, it will be set in the buc xml definition, creating X002 document trigger after X001 is received. It is not in the custom handler.

6.2.2.37 Create X002 Document Action Handler

```
@OnAction(type = EActionType.DOC_CREATE, document = EDocumentType.X_002)
```

Execution:

Create X002 document. Standard create execution, only content validation is added.

Actions:

- Create trigger Create Update Document action
- Create trigger Create Read Document action
- Create trigger Create Delete Document action
- Create trigger Create Send Document action

Validations:

- If content field `reasonForClosing` is OTHERS, value 99, then field `pleaseProvideMoreDetailsIf990therSelected` must not be empty

6.2.2.38 Update X002 Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE, document = EDocumentType.X_002)
```

Execution:

Update X002 document. Standard create execution, only content validation is added.

Actions:

- Create trigger Create Update Document action

Validations:

- If content field `reasonForClosing` is OTHERS, value 99, then field `pleaseProvideMoreDetailsIf990therSelected` must not be empty

6.2.2.39 Send X002 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_002)
```

Execution:

Sending X002 document, asking for case reopening.

Actions:

- Remove all X002 edition actions, (Update, Delete).
- Create trigger Remove Archive Case action
- Create trigger Receive X003 document action

Validations:

- If content field `reasonForClosing` is OTHERS, value 99, then field `pleaseProvideMoreDetailsIf990therSelected` must not be empty

6.2.2.40 Receive X002 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_002)
```

Execution:

Receiving X002 document. Reply with X003 document.

Actions:

- Create Trigger Create X003 Reply document action.
- Create trigger Remove Archive Case action
- Create trigger Read X002 document action

6.2.2.41 Send X003 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_003)
```

Execution:

Sending X003 document, replying about case reopening, agreeing or not.

Actions:

- Remove all X003 edition actions, (Update, Delete).
- Create trigger Receive X004 document action.

6.2.2.42 Receive X003 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_003)
```

Execution:

Receiving X003 reply document.

Actions:

- Create trigger Read X003 document action
- When all expected X003 are received, create trigger create X004 document action.
- If any received X003 document replies with NO to reopen, then X004 document will have `hasBusinessValidation = TRUE` in its metadata. It forces to reply NO to reopen case.

6.2.2.43 Create X004 Document Action Handler

```
@OnAction(type = EActionType.DOC_CREATE, document = EDocumentType.X_004)
```

Execution:

Create X004 document. Final decision if case is Reopened or not.

Actions:

- Create trigger Create Update Document action
- Create trigger Create Read Document action
- Create trigger Create Delete Document action
- Create trigger Create Send Document action

Validations:

- If document metadata X004 field *hasBusinessValidation* is TRUE, only allowed content for `decisionReopenCaseIndicator` is no, 1.

6.2.2.44 Update X004 Document Action Handler

```
@OnAction(type = EActionType.DOC_UPDATE, document = EDocumentType.X_004)
```

Execution:

Update X001 document. Standard create execution, only content validation is added.

Actions:

- Create trigger Create Update Document action

Validations:

- If document metadata X004 field *hasBusinessValidation* is TRUE, only allowed content for `decisionReopenCaseIndicator` is no, 1.

6.2.2.45 Send X004 Document Action Handler

```
@OnAction(type = EActionType.DOC_SEND, document = EDocumentType.X_004)
```

Execution:

Sending X004 document. If decision is YES, Case is set to status OPEN, all edition actions are set to ACTIVE, except for the ones for document X006 in case of having only 2 participants.

Actions:

If decision to reopen is NO

- Create trigger create Archive Case action.
- Recreate action Create X002 for reply for last processed X001 document

If decision to reopen is YES

- Case is set to status OPEN
- Reactivate actions as described in execution.
- Create trigger create X001 document action.

Validations:

- If document metadata X004 field *hasBusinessValidation* is TRUE, only allowed content for `decisionReopenCaseIndicator` is NO, 1.

6.2.2.46 Receive X004 Document Action Handler

```
@OnAction(type = EActionType.DOC_RECEIVE, document = EDocumentType.X_004)
```

Execution:

Receiving X004 document. If decision is YES, Case is set to status OPEN, all edition actions are set to ACTIVE, except for the ones for document X006 in case of having only 2 participants.

Actions:

- Create trigger Read X004 document action

If decision to reopen is NO

- Create trigger create Archive Case action.
- Recreate action Create X002 for reply for last processed X001 document

If decision to reopen is YES

- Case is set to status OPEN
- Reactivate actions as described in execution.
- Create trigger create X001 document action.

6.3 Custom Handlers

In some BUC it is required to modify the standard behaviour as described before. It works similarly as explained for custom handler in administrative documents, X008, Close,

Reopen, etc... The custom implementation is local to every buc, and using annotation it is linked to specific trigger.

```
@OnAction(buc = "role buc type", type = EActionType.actionType, document = EDocumentType.documentType)
```

There are many of them, and two of them are described as an example.

Custom handlers must implement the function *executeInternal* from abstract class `AbstractActionHandler<T extends Object>` and, if required, override the function *validateBusinessRules*. In many cases, custom handlers extend other more generic ones, like default *sendDocumentActionHandler*, and they delegate the implementation of certain functions, like *executeInternal* in that parent class.

```
protected abstract void executeInternal(Action action, T payload, ExecutionContext executionContext, CaseParticipantDO whoAmI) throws Exception;

void validateBusinessRules(final Action action, T payload, final ExecutionContext executionContext, final CaseParticipantDO whoAmI) throws BusinessValidationException;
```

6.3.1 Send U027 Document Action handler, UB_BUC_04.

This action handler is executed when U027 document is sent in this BUC, as per BUC annotation:

```
@MetaInfServices(ActionHandler.class)
@OnAction(buc = "cp_ub_buc_04", type = EActionType.DOC_SEND, document = EDocumentType.U_027)
```

Implementation class is

```
public class SendU027DocumentActionHandler extends SendDocumentActionHandler
```

This custom handler is required in order to introduce a business validation, it extends *SendDocumentActionHandler* so it does not *reimplement* *executeInternal* function, and it just overrides *validateBusiness*

In this case, overriding the validation rules for sending bulk documents, allows sending U027 document without subdocuments.

It only enforces having, at least, 1 subdocument if content is *partially accepted*, that is interpreted as business rule field *chargingInterestAccepted* has value 02.

```
@Override
public void validateBusinessRules(Action action, Void payload, ExecutionContext executionContext, CaseParticipantDO whoAmI)
    throws BusinessValidationException {

    log.debug("Validating business rules for action[{}] in the context [{}]", action, executionContext);

    // @formatter:off
    boolean isPartiallyAccepted = documentContentService.checkFieldValue(
        executionContext.getDocumentId(),
        executionContext.getCaseId(),
        isPartiallyAcceptedTest,
        "U027_Master", "ChargingInterestAccepted", "chargingInterestAccepted", "value"
    );
    // @formatter:on
```

```
        if (isPartiallyAccepted) {  
            if (documentMetadataService.countSubDocuments(executionContext.getDocumentId(),  
executionContext.getCaseId()) == 0)  
                throw new BusinessValidationException(ERROR_MESSAGE_NOT_RELATED_SUBDOCUMENTS);  
        }  
    }  
}
```

6.4 Condition Handlers

In BUC xml specification conditions may be added to document triggers, with the tag `onCondition`. If so, they are computed and that trigger is processed if condition is true.

```
<document type="DA002">  
  <parameters>  
    <!-- Check hasParticipantSelection - no need if isML(context)=false (in current BUC,  
isML(context)=false) -->  
    <parameter key="allowsAttachments">true</parameter> <!-- default is false -->  
  </parameters>  
  <triggers>  
    <!-- onCondition (Section 5.1 is Yes) -->  
    <createActionTrigger onAction="DOC_SEND" onResult="SUCCESS"  
      onCondition="cp_aw_buc_01a_SECTION_51_YES" actionType="DOC_CREATE"  
      documentType="DA003"/>  
  </triggers>  
</document>
```

A condition must have an unique identification, when the trigger is to be processed, if it has a condition, the handler implementing that condition will be executed, if it returns true the trigger will be processed.

A handler of conditions in a BUC must contain the annotation:

```
@MetaInfServices(ConditionTrigger.class)  
@BucCondition(buc = "buc type")
```

In the class it is defined. All conditions in a BUC are implemented in the same class.

There will be a function per condition to be implemented. There maybe be many in the same class.

For every function implementing a condition, it will have the annotation

```
@OnCondition(name = "condition ID")
```

So the handler for the condition in the example above, `onCondition="cp_aw_buc_01a_SECTION_51_YES"` would have as annotation:

```
@MetaInfServices(ConditionTrigger.class)  
@BucCondition(buc = "aw_buc_01a")  
public class AwBuc01aConditions extends AbstractConditionTrigger {  
    ...  
}
```

And in the function implementing the computation:

```
@OnCondition(name = "cp_aw_buc_01a_SECTION_51_YES")  
public Boolean isSection51Yes(final ExecutionContext executionContext) {...}
```

Condition handlers must implement abstract class `AbstractConditionTrigger`

7 Events

Events may be defined in order to be triggered during the execution of an action.

The steps during the execution of an actions are:

1. Fire event *BEFORE* for result *SUCCESS* for this action
 2. Execute *action handler*.
 3. Process *action triggers* for result *SUCCESS*.
 4. If no exception:
 - a) Fire event *AFTER* for result *SUCCESS* for this action
 - b) Delete action
- If exception:
- a) Process *action triggers* for result *FAILURE*.
 - b) Fire event for result *FAILURE*.

Any implementation of a listener of an event must include the `ListensForEvent` annotation:

```
@ListensForEvent(order = <EventOrder>, type = <EActionType>, result = <EResultType>, document = <EDocumentType>, buc = <buc type>)
```

Example:

```
@ListensForEvent(result = EResultType.SUCCESS, order = eu.ec.dgempl.eessi.rina.buc.api.event.EventOrder.BEFORE, type = EActionType.DOC_CREATE, document = EDocumentType.H_070, buc = "p_buc_01")
```

Where `EventOrder` is one of `EventOrder.BEFORE` or `EventOrder.AFTER`, `EActionType` is the type of the action it is listening, `result` if execution is successful or not, `document` refers to document type and `buc` is the name of the executing buc.

Version is the default where the event is defined, no matter if defined in specific buc or in buc-commons code. The version is the same.

In order to find the right handler for the fired event, there is a hierarchy in the registry. When a handler is found, search stops and that handler is executed.

`${BUC}_${BUC_VERSION}_${ACTION}_${DOCUMENT}_${RESULT}_${ORDER}`

Buc name, buc version, action type, document type, result

`${BUC}_${BUC_VERSION}_${ACTION}_%s_${RESULT}_${ORDER}`

Buc name, buc version, action type, result

`ANY_${BUC_VERSION}_${ACTION}_%s_${RESULT}_${ORDER}`

Buc version, action type, result

8 Practical use of Testing Framework

8.1 XML Test File

Please refer also to EESSI 2020 - RINA - Business Use Case Engine Architecture for the testing framework architecture.

In order to test the BUC XML Files, tests must be constructed. They can be written in plain java, using the API of the BUC Engine directly. However, the Testing Framework is provided to ease the writing of these tests.

Within the Testing Framework, the tests are written in XML files. These files are called the XML Test Files.

They contain statements that call the BUC Engine API, so the tester not familiar with this API can write his own tests using only the definition of these statements and their practical usage.

The present chapter 7 aims at covering these statements by introducing some examples of tests covering the following features:

- creating the case
- setting up participants
- creating documents of different types : normal documents (like P2000), request/reply documents (like P8000/9000)
- updating documents
- sending documents to all participants, to a subset of them
- sending a specific content
- sending administrative documents like X008
- asserting actions
- receiving documents
- receiving documents with a specific content
- reading documents

8.2 Corresponding JAVA Class

Each XML Test File is linked to a JAVA class via the annotations @Test and @BucTestCase used in the class.

The annotation @Test says that the next method defines a test. The annotation @BucTestCase defines the id of the test and its related xml test file (the file that we define in section 7).

An example of such a JAVA class is shown here (Hbuc10_FlowTest).

```
public class HBuc10_FlowTest extends AbstractFlowTest {
```

```
@Test
@BucTestCase(id = "h_buc_10_main_flow", file = "/test-flows/h_buc_10_main_flow.xml")
public void testMainFlow() {

    log.info("Finished flow test - h_buc_10_man_flow");

}

/**
 * Payload provider method, which is used to get payloads for the actions
 * from the test
 *
 * @param testId
 * @param payloadId
 * @return
 * @throws DataSourceException
 */
@PayloadProvider
public Object getPayloadFor(final String testId, final String payloadId) throws
DataSourceException {

    if (payloadId.equals("participants")) {

        List<ConversationParticipantDO> participants = new ArrayList<>();
        participants.add(getTestDataFactory().getConversationParticipantFor("BE:BE001",
EDocumentRole.SENDER));
        participants.add(getTestDataFactory().getConversationParticipantFor("ES:ES001",
EDocumentRole.RECEIVER));

        return participants;

    }

    return null;
}
}
```

In the previous example, the annotation `@BucTestCase` defines

- the id of the test as: "h_buc_10_main_flow".
- the xml test file as: "/test-flows/h_buc_10_main_flow.xml".

In this class, another annotation is used: `@PayloadProvider`. The method annotated by this annotation defines the possible participant lists that can be used as payloads in the tests defined in the class. Each participant is given a role (sender or receiver).

In the previous example, the payloadId "participants" contains:

- the sender BE:BE001 and
- the receiver ES:ES001.

As a consequence this mechanism associates the test id "h_buc_10_main_flow" with the payloadId "participants".

8.3 Structure of the XML Test File

This section defines the components of the XML Test File with the help of real examples.

8.3.1 Header

The header describes the name of the test and the namespace of the test document:

```
<bucTestCase name="h_buc_10_main_flow"
            xmlns="http://ec.europa.eu/eessi/rina/buc/test">
```

8.3.2 Body

The body of the test contains an ordered list of statements. The different types of statements are explored in the sections below.

8.3.2.1 createCase

The createCase statement creates the case, with an case id, a case type, version and the id of the institution representing the case owner / process owner. Example:

```
<createCase caseId="caseId1"
            bucType="h_buc_10"
            bucVersion="4.2"
            processOwnerId="BE:BE001"/>
```

8.3.2.2 caseAction

The first action in a case is to set up the list of participants.

In the following example:

```
<caseAction caseId="caseId1"
            type="CASE_SET_PARTICIPANTS"
            payloadId="participants"/>
```

- the caseId references the Id of the case (same as in createCase),
- the type defines the case action. In this case, it sets the participants involved in the case,
- these participants are defined by the payloadId, discussed ins section 8.2:

```
if (payloadId.equals("participants")) {

    List<ConversationParticipantDO> participants = new ArrayList<>();
    participants.add(getTestDataFactory().getConversationParticipantFor("BE:BE001",
EDocumentRole.SENDER));
    participants.add(getTestDataFactory().getConversationParticipantFor("ES:ES001",
EDocumentRole.RECEIVER));

    return participants;

}
```

8.3.2.3 documentAction

After the case is created and the list of participants is set up, the documentAction statement will execute actions on documents : create, update, send, add attachment, add subdocument if it is bulk...

8.3.2.3.1 Create document without parent

```
<documentAction caseId="caseId1"
                type="DOC_CREATE"
                documentType="H130"
                documentId="H130id"/>
```

This example shows the creation (DOC_CREATE) of a document of type H130. The document is given an Id (H130id) that uniquely identifies the document in the whole test.

8.3.2.3.2 Create document with a parent

A document may sometimes be created a reply of a request document. For instance, document H002 is a reply to document H001 in H_BUC_01. The parameter `parentDocumentId` is used for that purpose.

```
<documentAction caseId="caseId1"
  type="DOC_CREATE"
  documentType="H002"
  documentId="H002id"
  parentDocumentId="H001id" />
```

Again the `parentDocumentId` must uniquely identify the parent document in question.

8.3.2.3.3 Send document without content

```
<documentAction caseId="caseId1"
  type="DOC_SEND"
  documentType="H130"
  documentId="H130id"/>
```

This example illustrates the sending (DOC_SEND) of the document of type H130 and with Id H103id (reminder: the document is uniquely defined by use of its Id). The document is sent without any particular payload. It means all participants defined in section 8.3.2.2 are involved.

8.3.2.3.4 Send document with content

```
<documentAction caseId="caseId1"
  type="DOC_SEND"
  documentType="H130"
  documentId="H130id"
  content="/documents-content/H130-condition-0.json"/>
```

The same example as the previous one. However, this time the document is sent with a particular content, defined in json format. It may be useful to give a particular content in order to test, for example, a condition associated to the DOC_SEND trigger, based on the content of the document sent.

In order to illustrate this more convincingly, check for example the real BUC XML File `cp_aw_buc_01a.xml` where this trigger is defined:

```
<document type="DA002">
  <triggers>
    <!-- onCondition (Section 5.1 is Yes) -->
    <createActionTrigger
      onAction="DOC_SEND"
      onResult="SUCCESS"
      onCondition="cp_aw_buc_01a_SECTION_51_YES"
      actionType="DOC_CREATE" documentType="DA003"/>
    </triggers>
  </document>
One corresponding test (named aw_buc_01a_DA002_true.xml) contains the statement:
<documentAction caseId="caseId2"
  type="DOC_SEND"
  documentType="DA002"
  documentId="DA002-1"
  content="/documents-content/DA002-condition-1.json"/>
```

The referenced json file `="/documents-content/DA002-condition-1.json` contains:

```
{
  "DA002": {
    "sedPackage": "Sector Components/AWOD/DA002",
    "sedGVer": "4",
    "sedVer": "0",
    "Person": {
      "PersonIdentification": {
        "familyName": "Crockett",
        "forename": "Jones",
        "dateBirth": "1958-05-04",
        "sex": {
          "value": [
            "01"
          ]
        }
      },
      ...
    },
    ...
  },
  ...
},
...
}
...
}
...
}
...
}
...
}
```

In other words, the statement :

```
<documentAction caseId="caseId2"
  type="DOC_SEND"
  documentType="DA002"
  documentId="DA002-1"
  content="/documents-content/DA002-condition-1.json"/>
```

will test the trigger:

```
<document type="DA002">
  <triggers>
    <!-- onCondition (Section 5.1 is Yes) -->
    <createActionTrigger
      onAction="DOC_SEND"
      onResult="SUCCESS"
      onCondition="cp_aw_buc_01a_SECTION_51_YES"
      actionType="DOC_CREATE"
      documentType="DA003"/>
    </triggers>
  </document>
```

through the following logic:

During the send (DOC_SEND) of DA002, the condition "cp_aw_buc_01a_SECTION_51_YES" will evaluate (see section 6.4 for more implementation details about that evaluation) if DA002 contains the value highlighted in green equal to "1". The test is built so the condition returns true (the json file contains "1"). Then the trigger DOC_SEND will create the action DOC_CREATE DA003.

In the test, an <assertActionExists> can then used to assert if the document DA003 may actually be created (see section 8.3.2.5).

8.3.2.3.5 Send document with participant payload

```
<documentAction caseId="caseId2"  
                type="DOC_SEND_PARTICIPANTS"  
                documentType="F026"  
                documentId="F026id"  
                payloadId="replyParticipants" />
```

This example illustrates the sending (DOC_SEND_PARTICIPANTS) of the document of type F026 and with Id F026id (reminder: the document is uniquely defined by use of its Id, so the documentId F026id must have been previously defined by a documentAction of type DOC_CREATE). The document is sent with a particular payload (replyParticipants). It means only the participants defined in the payloadId "replyParticipants" are involved.

Check the related JAVA code associated with the test (released file FbBuc01MainFlowTest.java related to FB_BUC_01), where "replyParticipants" is defined:

```
public class FbBuc01MainFlowTest extends AbstractFlowTest {  
    @Test  
    @BucTestCase(id = "fb_buc_01_main_flow", file = "/test-flows/fb_buc_01_main_flow.xml")  
    public void testMainflow() {}  
  
    @PayloadProvider  
    public Object getPayloadFor(final String testId, final String payloadId) throws  
        DataSourceException {  
  
        if (testId.equals("fb_buc_01_main_flow") && payloadId.equals("participants")) {  
            List<ConversationParticipantDO> participants = new ArrayList<>();  
  
            participants.add(getTestDataFactory().getConversationParticipantFor("BE:BE001",  
EDocumentRole.SENDER));  
            participants.add(getTestDataFactory().getConversationParticipantFor("ES:ES001",  
EDocumentRole.RECEIVER));  
            participants.add(getTestDataFactory().getConversationParticipantFor("FR:FR001",  
EDocumentRole.RECEIVER));  
  
            return participants;  
        }  
  
        if (testId.equals("fb_buc_01_main_flow") && payloadId.equals("replyParticipants")) {  
            List<ConversationParticipantDO> participants = new ArrayList<>();  
  
            participants.add(getTestDataFactory().getConversationParticipantFor("ES:ES001",  
EDocumentRole.SENDER));  
            participants.add(getTestDataFactory().getConversationParticipantFor("BE:BE001",  
EDocumentRole.RECEIVER));  
            participants.add(getTestDataFactory().getConversationParticipantFor("FR:FR001",  
EDocumentRole.RECEIVER));  
  
            return participants;  
        }  
  
        return null;  
    }  
}
```

8.3.2.4 receiveAction

After a document is sent, it must be explicitly received by the other case(s). These cases are defined by the "receiveAction" statement. They are given a caseId, that uniquely represents that case in the test flow, a receiverId, that must match one of the receiver participants (see payload) the message was sent to, a documentId (the unique identifier

of the document that was sent and is now received by the receiver defined here), and possibly a content.

```
<receiveAction caseId="caseId2"
               receiverId="ES:ES001"
               documentId="H130id"
               content="/documents-content/H130-condition-0.json"/>
```

Once again, just like during the sending of the document, the "content" parameter will be used to impose the content of the received document, and be able to test some triggers in some BUC XML File.

8.3.2.5 assertActionExists

The consequence of a previous statement can be asserted true or false thanks to the statement `assertActionExists`:

```
<assertActionExists caseId="caseId2"
                   type="DOC_SEND"
                   documentType="X008"
                   documentId="X008id"
                   parentDocumentId="H131id"/>
```

This example illustrates that in `caseId2`, the document of type `X008`, with `documentId` `X008id` and `parentDocumentId` `H131id` can be sent (`DOC_SEND`) without any particular payload (otherwise `DOC_SEND_PARTICIPANTS` would be used).

In brief, it is asserted that the administrative document `Invalidate` (defined by the Id `X008id`) can be sent for main document identified by `H131id`.

Use `negative="true"` to assert the contrary of the condition:

```
<assertActionExists caseId="caseId2"
                   type="DOC_SEND"
                   documentType="X008"
                   documentId="X008id"
                   parentDocumentId="H131id"
                   negative="true"/>
```

8.4 Example

This section presents some typical full example. Comments inside explain the main ideas.

```
<?xml version="1.0" encoding="UTF-8"?>
<bucTestCase name="h_buc_10_main_flow" xmlns="http://ec.europa.eu/eessi/rina/buc/test">

  <!-- create case -->
  <createCase caseId="caseId1" bucType="h_buc_10" bucVersion="4.2" processOwnerId="BE:BE001"/>

  <!-- asserts that choose participants and delete case are available -->
  <assertActionExists caseId="caseId1" type="CASE_SET_PARTICIPANTS"/>
  <assertActionExists caseId="caseId1" type="CASE_DELETE"/>

  <!-- set participants -->
  <caseAction caseId="caseId1" type="CASE_SET_PARTICIPANTS" payloadId="participants"/>

  <!-- CO creates starter H130 -->
  <assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="H130"/>
  <documentAction caseId="caseId1" type="DOC_CREATE" documentType="H130"
                  documentId="H130-0"/>

  <!-- CO updates draft starter H130 -->
```

```
<assertActionExists caseId="caseId1" type="DOC_UPDATE" documentId="H130-0"/>
<documentAction caseId="caseId1" type="DOC_UPDATE" documentType="H130"
    documentId="H130-0"/>

<!-- CO sends H130 with content from file documents-content/H130-condition-0.json -->
<assertActionExists caseId="caseId1" type="DOC_SEND" documentId="H130-0"/>
<documentAction caseId="caseId1" type="DOC_SEND" documentType="H130"
    documentId="H130-0"
    content="/documents-content/H130-condition-0.json"/>

<!-- asserts for case after starter is sent -->
<assertActionExists caseId="caseId1" type="CASE_UPDATE_PARTICIPANTS"
    negative="true"/>
<assertActionExists caseId="caseId1" type="CASE_DELETE"
    negative="true"/>

<!-- asserts for documents after starter is sent -->
<assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="H001"/>
<assertActionExists caseId="caseId1" type="DOC_RECEIVE" documentType="X007"/>
<assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="X009"/>
<assertActionExists caseId="caseId1" type="DOC_RECEIVE" documentType="X009"/>
<assertActionExists caseId="caseId1" type="DOC_RECEIVE" documentType="X050"/>
<assertActionExists caseId="caseId1" type="DOC_RECEIVE" documentType="X100"/>
<assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="X001"/>

<!-- CP receives H130 with content from file documents-content/H130-condition-0.json -->
<receiveAction caseId="caseId2" receiverId="ES:ES001" documentId="H130-0"
    content="/documents-content/H130-condition-0.json"/>
<assertActionExists caseId="caseId2" type="DOC_CREATE" documentType="H131"
    parentDocumentId="H130-0"/>

<!-- CP creates and sends reply H131 -->
<documentAction caseId="caseId2" type="DOC_CREATE" documentType="H131"
    documentId="H131-1"
    parentDocumentId="H130-0"/>
<assertActionExists caseId="caseId2" type="DOC_SEND" documentId="H131-1"
    parentDocumentId="H130-0"/>
<documentAction caseId="caseId2" type="DOC_SEND" documentType="H131"
    documentId="H131-1"/>

<!-- CO receives reply H131 -->
<receiveAction caseId="caseId1" receiverId="BE:BE001" documentId="H131-1"/>
<assertActionExists caseId="caseId1" type="DOC_READ" documentType="H131"
    documentId="H131-1" />

<!-- Check that no new H130 can be sent again -->
<assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="H130"
    negative="true"/>

<!-- CP sends X008 on H131 -->
<assertActionExists caseId="caseId2" type="DOC_CREATE" documentType="X008"
    parentDocumentId="H131-1"/>
<documentAction caseId="caseId2" type="DOC_CREATE" documentType="X008"
    documentId="X008-1"
    parentDocumentId="H131-1"/>
<assertActionExists caseId="caseId2" type="DOC_SEND" documentType="X008"
    documentId="X008-1"
    parentDocumentId="H131-1"/>
<documentAction caseId="caseId2" type="DOC_SEND" documentType="X008"
    documentId="X008-1"/>

<!-- CO receives X008 on H131 -->
<receiveAction caseId="caseId1" receiverId="BE:BE001" documentId="X008-1"/>
<assertActionExists caseId="caseId1" type="DOC_READ" documentType="X008"
    documentId="X008-1"/>

<!-- CP replies new H131 -->
<assertActionExists caseId="caseId2" type="DOC_CREATE" documentType="H131"
```

```
<documentAction caseId="caseId2" type="DOC_CREATE" documentType="H131"
                                parentDocumentId="H130-0"/>
                                documentId="H131-2"
                                parentDocumentId="H130-0"/>
<assertActionExists caseId="caseId2" type="DOC_SEND" documentId="H131-2"
                                parentDocumentId="H130-0"/>
<documentAction caseId="caseId2" type="DOC_SEND" documentType="H131"
                                documentId="H131-2"/>
<assertActionExists caseId="caseId2" type="DOC_CREATE" documentType="X008"
                                parentDocumentId="H131-2"/>

<!-- CO receives new H131 -->
<receiveAction caseId="caseId1" receiverId="BE:BE001" documentId="H131-2"/>
<assertActionExists caseId="caseId1" type="DOC_READ" documentType="H131"
                                documentId="H131-2" />
<assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="H130"
                                negative="true"/>

<!-- Test of exchange of H001 and reply H002 initiated by CP -->
<assertActionExists caseId="caseId2" type="DOC_CREATE" documentType="H001" />
<assertActionExists caseId="caseId1" type="DOC_RECEIVE" documentType="H001" />

<documentAction caseId="caseId2" type="DOC_CREATE" documentType="H001"
                                documentId="H001"/>
<documentAction caseId="caseId2" type="DOC_SEND" documentType="H001"
                                documentId="H001"/>
<assertActionExists caseId="caseId2" type="DOC_RECEIVE" documentType="H002"
                                parentDocumentId="H001" />

<receiveAction caseId="caseId1" receiverId="BE:BE001" documentType="H001"
                                documentId="H001" />

<assertActionExists caseId="caseId1" type="DOC_CREATE" documentType="H002"
                                parentDocumentId="H001" />
<documentAction caseId="caseId1" type="DOC_CREATE" documentType="H002"
                                documentId="H002"
                                parentDocumentId="H001" />
<documentAction caseId="caseId1" type="DOC_SEND" documentType="H002"
                                documentId="H002"/>

<receiveAction caseId="caseId2" receiverId="ES:ES001" documentType="H002"
                                documentId="H002" />
<documentAction caseId="caseId2" type="DOC_READ" documentType="H002"
                                documentId="H002"/>

</bucTestCase>
```