ebBP WorkItem-43 Name and NameID

Kenji Nagahashi
April 26, 2004

1. Purpose of this document 1
2. Work Item Summary 1
3. Use case and requirements to consider 2
4. Speculations 3
4.1. Globally-scoped, Package-scoped or Document scoped? 3
4.1.1. Globally-scoped identifier 3
4.1.2. Package-scoped identifier 3
4.1.3. Document-scoped identifier 4
4.2. Is Registry Relevant? 4
4.3. BPSS elements need globally unique identifier? 4
4.4. How to deal with versioning? 5
4.5. name or nameID attribute? 6
4.6. Are IDs painful for humans? 7
5. Proposed Solutions 8
5.1. Solution 0 – Use both Package-scoped name and Globally-unique nameID attributes for references 8
5.2. Solution 1 – use Package-scoped nameID attribute for references 8
5.3. Solution 2 – use Document-scoped nameID attribute for references 9
5.4. Solution 2 amended – use Document-scoped nameID attribute for reference, with mandatory name attribute 9
5.5. Solution 3 – use Package-scoped name attribute for reference 9
5.6. Solution 4 – use Document-scoped name attribute for reference 10
5.7. Solution 5 – use Document-scoped nameID attribute for references and globally-unique externalID for references from external documents 10

1. Purpose of this document

This document is a summary of ebBP TC discussion on work item 43: name and nameID. The author reviewed all relevant e-mails and trys to resolve issues raised (In section 4: Speculations). No new requirements are raised in this document. Finally the author lists several possible solutions, including the solution made by other TC members during discussions, with analysis on impact on other specification related to BPSS. For proposed solutions to work item 43, see section 5.

2. Work Item Summary

In the BPSS Schema 1.01 and 1.1, elements have name and nameID attribute so that it can be referenced from other elements – either from elements in the same BPSS instance, or from elements in other (possibly non-BPSS) documents.

Example 1:

```
<BusinessDocument name="ABC" nameID="ABC_ID">
…
<BusinessTransaction name="Trans1" nameID="Trans1_ID">
  <RequestingBusinessActivity name="ReqBizA" nameID="ReqBizA_ID" …>
  <DocumentEnvelope businessDocument="ABC" businessDocumentIDREF="ABC_ID" …/>
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
```

For BusinessDocument element, name attribute is required. nameID attribute is optional.
For DocumentEnvelope element, businessDocument attribute is required, and businessDocumentIDREF is optional.
When you use both businessDocument and businessDocumentIDREF attribute:
1) businessDocument attribute must have the value of name attribute of a BusinessDocument element.
2) businessDocument attribute must have the value of nameID attribute of a BusinessDocument element.
3) BusinessDocument element in 1) and 2) must be the same one.

- As nameID is optional, it is acceptable to use name attribute only (and its referencing counterpart). But it is not explicit that its value must uniquely identify the element, because name attribute is of type xsd:string. And in this case, we don't need nameID attributes at all.
- It is not possible to use nameID alone, because name attribute is mandatory.
- If you use both, you have to maintain consistency between two references. It is error-prone to have two attributes of equivalent function and it imposes extra burden on BPSS authors (or authoring tools developers).

And as a separate but related topic, BPSS 1.1 defines nameID attribute to be of type "GUID" which is derived from xsd:string. This name "GUID" implies, at least for some people, its value must be guaranteed to be globally unique identifier (This might not be an intention of 1.1 specification authors.) Ensuring global uniqueness requires special scheme, like 128bit identifier used by Windows to identify COM objects, or DCE. Problem with this kind of globally unique identifiers is that we don't have any systematic way of resolving those identifiers.

This leads to following questions.

- Do we need two referencing schemes (name and nameID)?
- Does a type of nameID really need to be "globally unique identifier"?

3. Use case and requirements to consider

- Elements in BPSS description must be uniquely identifiable from outside of the BPSS instance. This is a requirement from eBA.
- BPSS has a concept of package. Referencing scheme must not interfere with packaging.
- BPSS has 'include'. It must be possible to reference elements in the included BPSS instance from elements in including BPSS. Point is that two distinct BPSS instances may have the same name/nameID for element in each instance.

4. Speculations

4.1. Globally-scoped, Package-scoped or Document scoped?

Package-scoped identifier means that you can tag multiple BPSS elements with the same identifier string, as long as those elements are in the different Package. With document-scoped identifier approach you must make sure particular identifier string is used for single BPSS element in the document. Globally-scoped identifiers are guaranteed to be unique within a realm much larger than one document, e.g. Registry, or World.
IMO, having multiple Packages in a BPSS instance complicated the problem. We could map a concept of Package to BPSS instance (as with targetNamespace in XML Schema).

4.1.1. Globally-scoped identifier

With globally-scoped approach, elements can be referred to from anywhere in the world with single identifier. For this goal, we need special scheme like UUID (used by ebXML Registry). It is BPSS authors responsibility to guarantee uniqueness (Registry will never assign UUID to BPSS elements – see 3.2), and it is usually painful for humans to manually create such identifier.
One problem with this approach is that there's no systematic way to resolve such globally unique identifier into an actual resource. You can't tell where you can find BPSS element, just given a UUID string. First you have to designate BPSS documents which contains elements with UUIDs – this is effectively equivalent of using Document-scoped identifiers. The only globally unique identifier that can practically be resolved is URL.
Another problem is that, if we follow strictly the principle of "global uniqueness", we have to assign distinct identifier to elements in a different version of the (logically) same document. We need some special scheme to maintain continuity between versions.

4.1.2. Package-scoped identifier

If we take Package-scope approach for name and nameID attributes, their values are not guaranteed to be unique within a document. So we need to define a scheme for constructing document-scoped identifier out of Package-scoped identifier and (hierarchical) Package names. For this purpose, BPSS 1.05 defines Java-like dot-delimited notation (7.4.1 Packages and Includes)

> A package defines the namespace of the elements inside it. You cannot have two model elements with the same name within the same package. Model element names can be qualified with the package using the Java notation:
> Business Transaction.Order Entry.Process Purchase Order
> Which means that the Process Purchase Order business transaction is defined within the package Order Entry, which is itself, defined within the Business Transaction package. Note that there is no ambiguity in using spaces within the names of packages or model elements.
> If a model element in package Order Entry needs to name something in a package called Billing, it must include this package to make its elements visible to its own model elements。 Unlike an import, include requires that all model elements from the Billing package be fully qualified. So if we want to designate the Invoice business document within the Order Entry.Process Purchase Order transacition we need to refer to the Billing.Invoice document, assuming it is defined in the Business Transcation.Billing package.

Package-scoped identifiers must be of type (or type derived from) xsd:string, not xsd:ID. This was the original intent in ebBPSS v1.1 to define GUID type for nameID attributes.

4.1.3. Document-scoped identifier

Document-scoped identifier is traditional xsd:ID/IDREF and widely used. For example, ebCPPA uses xsd:ID/IDREF to identify elements such as DeliveryChannel, Transport, and Certificate.
It works well with URL scheme and XLink/XPointer. Benefit of using Document-scope identifier is that processor of referring document need no semantic knowledge of document referred to.
It interacts, however, badly with BPSS's Pacakges. It is also possible to have the same identifier in Include-ing BPSS and Include-d BPSS. We need special scheme to address such situation.

4.2. Is Registry Relevant?

ebXML Registry specification mandates use of UUID for RegistyObject. But name/nameID discussions and GUID discussion have nothing to do with Registry, because we're talking about identification of element internal to BPSS instance, which is not supposed to be a RegistryObject. I assume Registry never assigns UUID to elements in BPSS instance. Even if nameID attribute were a UUID, Registry won't change the value when BPSS is registered into Registry. Registry is irrelevant to this work item. In other words, there's not need to worry that element identifier might be changed by Registry between versions.

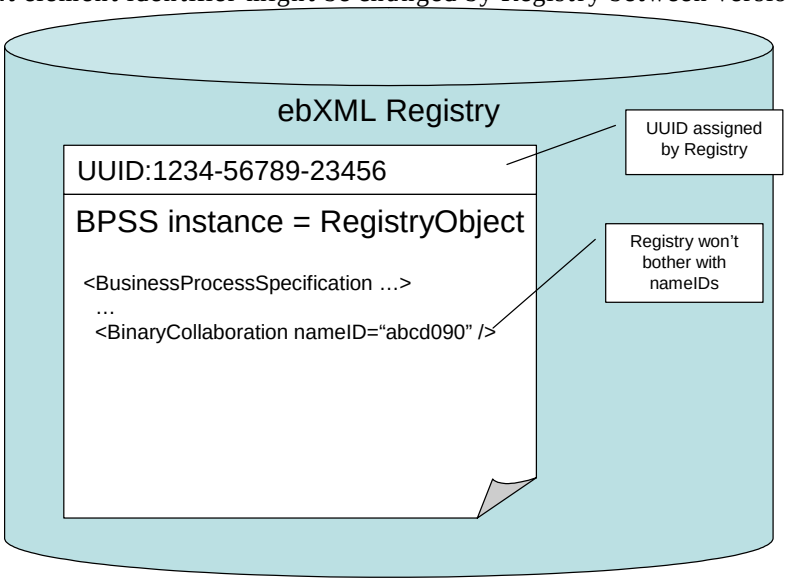

Figure 1: BPSS instance in a Registry

4.3. BPSS elements need globally unique identifier?

IMO, we don't need globally unique identifier for BPSS elements.
As eBA states "Elements in BPSS description must be uniquely identifiable from outside of the BPSS instance", it may seem reasonable to tag each BPSS element with globally unique identifier like UUID. But this approach has a flaw. In order for such GUID to work, it must be possible to resolve those GUIDs. While ebXML Registry does provide resolution service from UUID to RegistryObject, it is very unlikely for BPSS elements to be a RegistryObject (see section 3.2 above). We cannot rely on Registry to directly locate BPSS elements by UUID.
What is really likely to happen is that we register BPSS instance as a RegistryObject, and then reference BPSS elements in it relative to the instance. This is a common way of referencing elements in a document and also works with ordinary file systems or web environment. In this approach, we don't need globally unique identifier for each element. We only need element identifier local to the document.

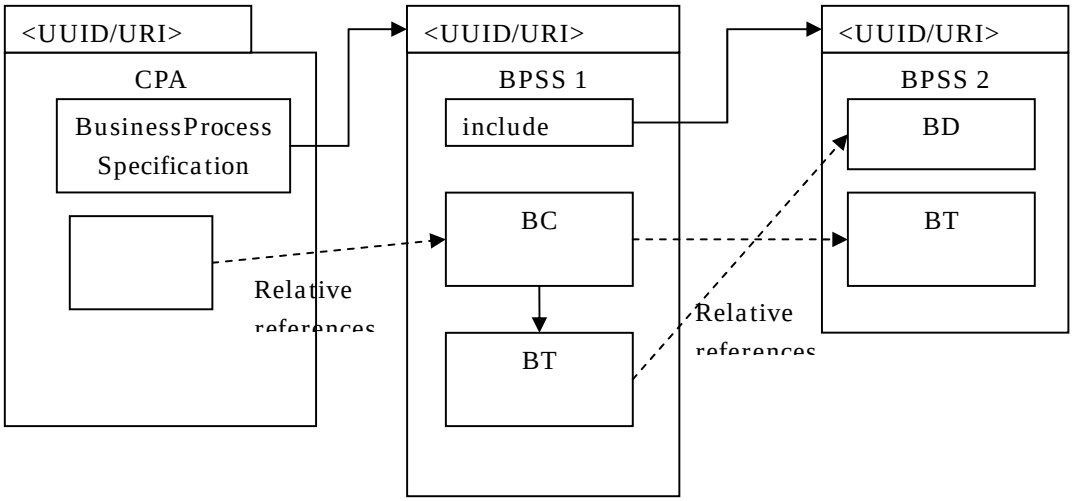

Figure 2: global document id + local element id

I suspect the type name "GUID" brought some confusion to the discussion. Some people (including me) took it for "globally unique identifier", like UUID used by Registry. This might not be an original intent----it's unclear. If we are not mandating real "globally unique identifier", we should not use type name "GUID".

4.4. How to deal with versioning?

Some discussion touched a versioning issue. Point was that we need both dynamic binding and static binding of business process description.

- Dynamic Binding: Business Process Specification A references (reuses) "the newest version" of Business Process Specification B.
- Static Binding: Business Process Specification A references (reuses) a "specific version" of Business Proces Specificatoin B.

This can be easily achieved with "global document id +local element id" scheme above, *if* global document identifiers support versioning (Figure 3). Since versioning requires support from content management systems (such as ebXML Registry) and it has universal usage for any resources maintained in Registry. BPSS should not define its own, local, versioning scheme.

Here I assumed that versioning is done per document basis----if we are to put multiple versions of elements within one BPSS instance, we would need new syntax to describe versions in BPSS. To me, this is unnecessary complication.

ebCPPA's ProcessSpecification element has "version" attribute to identify the version of BPSS instance.

Anyway, we don't have good requirements description for versioning. It would be too early to consider versioning issues in the context of this work item.

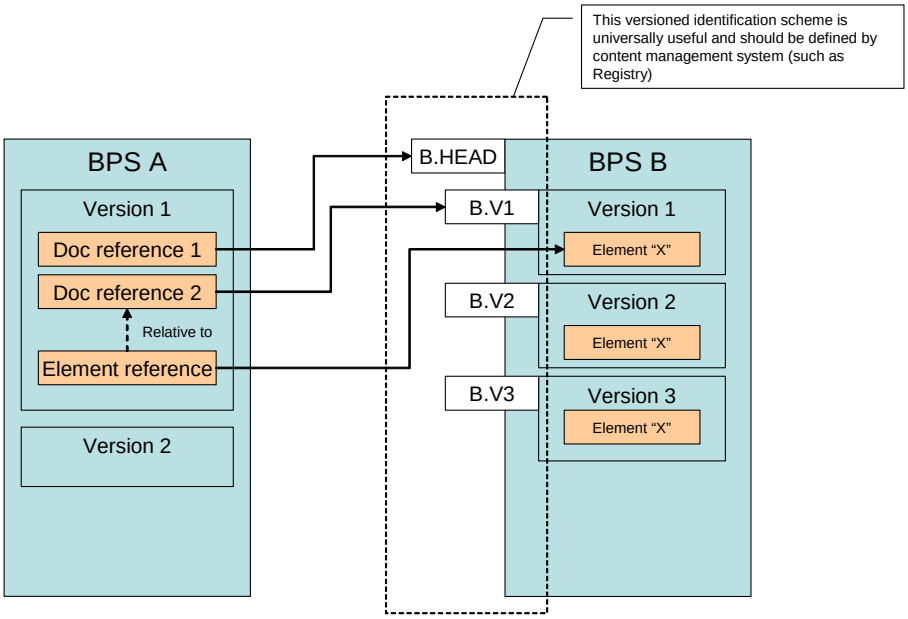

Figure 3: versioned reference

4.5. name or nameID attribute?

Following table lists references from ebCPPA to ebBPSS.

CPA element	Reference to BPSS element
PartyInfo	
CollaborationRole	
Role	
@name	Value taken from a name attribute of one of a BinaryCollaboration's Role elements described in the ebBPSS document.
@xlink:href	URI that identifies the element or attribute within ebBPSS document that defines the role in the context of BusinessCollaboration.
ServiceBinding	
Service	Uuid attribute specified for the ProcessSpecification element in the ebBPSS document.
CanSend	
ThisPartyActionBinding	
@xlink:href	URI that specifically identifies the RequestingBusinessActivity or RespondingBusinessActivity element
ActionContext	
@binaryCollaboration	Matches the value of the name attribute of the BinaryCollaboration element.
@businessTransactionActivity	Matches the value of the name attribute of the BusinessTransactionActivity element.
@requestOrResponseAction	Matches the value of the name attribute of either the RequestingBusinessActivity or

	RespondingBusinessActivity element.
CollaborationActivity	
@name	Matches the value of the name attribute of the CollaborationActivity element.

In summary, ebCPPA uses two major ways of referencing BPSS elements:

1. URI (URL with ID value as a fragment
2. name attribute of elements

Thus if we make name attribute optional, it will impact ebCPPA specification in many places.

On the other hand, there will be no impact when we obsolete nameID attributes. Although there are two cases where ebCPPA uses nameID as part of reference (xlink:href), they will not be affected by removing nameID attribute if name attribute is defined to be an element identifier.

4.6. Are IDs painful for humans?

You can use human friendly string for IDs, such as “PurchaseOrderBC” when you create BPSS manually. Tools may generate human-unfriendly strings for IDs, but IMO, it is more painful to have two reference mechanisms than to have only one.

Also name attribute may not always be user-friendly. Following is the excerpt from section 8.5 Scoped Name Reference of BPSS 1.05:

> Plain text attributes do not have this capability and may result in duplicate names. To unambiguously identify a referenced element using plain text attribute in the referencing attribute it is strongly recommended that XPath syntax be used. However, this is not enforced in the Schema.
>
> …
>
> Example of named elements and references:

```
<Package name=”ebXMLOrdering”>
  <BinaryCollaboration name=”OrderCollaboration” nameID=”b112”>
    <Role name=”buyer” nameID=”r224”/>
    <Role name=”seller” nameID=”r225”/>
  </BinaryCollaboration>
</Package>
…
<!—the XPath approach →
<Performs
Role=’//Package[@name=”OAGOrdering”]/BinaryCollaboration[@name=”OrderCollaboration”]/Role[@name=”buyer”]’/>
```

> It is not required to use the full path specification as shown above, other forms of XPath expression could be used as long as they resolve to a single reference. For example if buyer was unique to the document then the XPath could have been:

```
<Performs Role=’//Role[@name=”buyer”]’/>
```

> Relative paths are also allowed for example:

```
<BusinessTransactionActivity fromRole=’../Role[@name=”buyer”]’ …/>
```

This XPath approach is functionally equivalent of Package-qualified identifier (see3.1.2) and much more painful than (possibly user-friendly) plain ID strings.

5. Proposed Solutions

5.1. Solution 0 – Use both Package-scoped name and Globally-unique nameID attributes for references

As specified in BPSS v1.1:

- Elements have mandatory name attribute and optional nameID attribute, and both used for element reference – need to maintain consistency if both of these attributes are used.
- Name attribute is of type xsd:string and Package-scoped.
- NameID attribute is optional and need to be globally unique if specified.
- Two ways of referring elements in other Package described using name attribute value: 1) Java-like qualification, and 2) XPath

Example:

```
BPSS
<Package name="Documents">
<BusinessDocument name="ABC" nameID="634eb12a-7962-4928-a8de-125bd516f321">
…
<Package name="Process">
<BusinessTransaction name="Trans1" nameID="992fc9d4-4c4f-4796-9d24-8a1cae65b25d">
  <RequestingBusinessActivity name="ReqBizA" nameID="854100f1-4881-4a50-90fa-b753a78c59e9" …>
  <DocumentEnvelope businessDocument="Documents.ABC"
                    businessDocumentIDREF="634eb12a-7962-4928-a8de-125bd516f321" …/>
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
CPPA
<!-- may not work -->
<ActionContext binaryCollaboration="…" businessTransaction="Trans1" …/>
```

This is incompatible with ebCPPA 2.0a, because ebCPPA expects that name attribute is Document-scoped.

5.2. Solution 1 – use Package-scoped nameID attribute for references

- Make name attribute optional. It is a descriptive text for an element and never used for referring elements. No constraints on its value. Remove all attributes referring name attribute value.
- Make nameID attribute mandatory. Its type is (or derived from) xsd:string, not xsd:ID. Value of nameID attribute is required to be unique within a package.
- Since nameIDs are Package-scoped, we need to specify Package qualified identifier scheme for reference from external document (see 4.1.2). Clearly state this qualified identifier must be used when external document refer BPSS elements (also applicable to reference to elements in the "Include"-ed document).

Example:

```
BPSS
<Package name="Documents">
<BusinessDocument name="ABC" nameID="ABC">
…
<Package name="Process">
<BusinessTransaction name="Transaction 1" nameID="Trans1">
  <RequestingBusinessActivity name="Requesting Business Activity" nameID="ReqBizA" …>
  <DocumentEnvelope businessDocumentIDREF="Documents.ABC" …/>
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
CPPA
<ActionContext binaryCollaboration="…" businessTransaction="Process.Trans1" …/>
```

Impact: As name attribute is optional and not guaranteed to be unique within document-scope any more, ebCPPA needs changes. ebCPPA have to use qualified identifier instead of name attribute values.

5.3. Solution 2 – use Document-scoped nameID attribute for references

This is a variant of Solution 1, where

- Type of nameID attribute is xsd:ID. This means nameID is Document-scoped, irrelevant to package structure.

Since nameID is Document-scoped, we don't need qualified identifier syntax, but we may need extra care to make nameID unique within a document (such as embedding package name in nameID).

Example:

```
BPSS
<Package name="Documents">
<BusinessDocument name="ABC" nameID="Documents_ABC">
…
<Package name="Process">
<BusinessTransaction name="Transaction 1" nameID="Process_Trans1">
  <RequestingBusinessActivity name="Requesting Business Activity" nameID="Process_ReqBizA" …>
  <DocumentEnvelope businessDocumentIDREF="Documents_ABC" …/>
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
CPPA
<ActionContext binaryCollaboration="…" businessTransaction="Process_Trans1" …/>
```

Impact: As name attribute is optional and not guaranteed to be unique within document-scope any more, ebCPPA needs changes. ebCPPA have to use nameID attribute values instead of name attribute values to refer to BPSS elements.

5.4. Solution 2 amended – use Document-scoped nameID attribute for reference, with mandatory name attribute

Amendment to solution 2 was proposed to the list by Monica Martin, per discussion with Dale Moberg and Hima Mukkamara, It is to mandate the use of name attribute in addition to solution 2.

Name attribute is used by CPPA 2.0 to reference BPSS elements. By making name attribute mandatory, we can avoid changes to CPPA specification. Name attribute must be unique within a BPSS document, but it is not used by other BPSS elements for reference purpose. Its type is xsd:string and arbitrary text is allowed.

5.5. Solution 3 – use Package-scoped name attribute for reference

- Name attribute is mandatory and of type xsd:string. The value must be unique within a Package.
- Deprecate use of nameID attributes and its referring counterparts. They are never used for referring elements. Let type of nameID attribute be xsd:string to prevent possible validation errors.
- Since name is Package-scoped, we need to specify Package qualified identifier scheme for reference from external document (see 4.1.2). Clearly state this qualified identifier must be used when external document refer BPSS elements (also applicable to reference to elements in the "Include"-ed document).

Example:

```
BPSS
<Package name="Documents">
<BusinessDocument name="ABC">
…
<Package name="Process">
<BusinessTransaction name="Transaction 1">
  <RequestingBusinessActivity name="Requesting Business Activity" …>
  <DocumentEnvelope businessDocument="Documents.ABC" …/>
```

```
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
CPPA
<ActionContext binaryCollaboration="…" businessTransaction="Processs.Transaction 1" …/>
```

Impact: some impact on ebCPPA. Since name attribute is Package-scoped, ebCPPA needs to use qualified identifier (see 3.1.2) to refer to BPSS elements.

5.6. Solution 4 – use Document-scoped name attribute for reference

- Name attribute is mandatory and of type xsd:string (not xsd:ID). The value must be unique within a document, irrelevant of Package structure.
- Deprecate use of nameID attributes and its referring counterparts. They are never used for referring elements. Let type of nameID attribute be xsd:string to prevent possible validation errors.

Example:

```
BPSS
<Package name="Documents">
<BusinessDocument name="Document ABC">
…
<Package name="Process">
<BusinessTransaction name="Process Transaction 1">
  <RequestingBusinessActivity name="Process Requesting Business Activity" …>
  <DocumentEnvelope businessDocument="Document ABC" …/>
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
CPPA
<ActionContext binaryCollaboration="…" businessTransaction="Process Transaction 1" …/>
```

Impact: No implact on ebCPPA, because name attribute is Document-scoped, as expected by ebCPPA.

5.7. Solution 5 – use Document-scoped nameID attribute for references and globally-unique externalID for references from external documents

(Proposed by Martin Roberts – I'm not sure if I capture his idea correctly)

- Name attribute is optional. It is not required to be unique within a BPSS instance.
- nameID attribute is mandatory. Its type is xsd:ID (Document-scoped).
- Those elements that have a possibility of being referenced by external entity have mandatory externalID attribute. Its value must be globally unique.

Example:

```
BPSS
<Package name="Documents">
<BusinessDocument name="ABC" nameID="Documents_ABC" >
…
<Package name="Process">
<BusinessTransaction name="Transaction 1" nameID="Process_Trans1"
      externalID="992fc9d4-4c4f-4796-9d24-8a1cae65b25d">
  <RequestingBusinessActivity name="Process_ReqBizA" nameID="ReqBizA"
      externalID="854100f1-4881-4a50-90fa-b753a78c59e9" …>
  <DocumentEnvelope businessDocumentIDREF="Documents_ABC" …/>
  </RequestingBusinessActivity>
  …
</BusinessTransaction>
CPPA
<ActionContext binaryCollaboration="…" businessTransaction="992fc9d4-4c4f-4796-9d24-8a1cae65b25d" …/>
```

Note that all elements that may be referred by ebCPPA and other docments must have externalID (see section 3.5)
Impact: ebCPPA must change references to name attribute to ones to nameID attribute.