

2

4

6

8

E-Commerce Integration Meta-Framework – Proof of Concept Documentation (ECIMF-POC)

10

CEN/ISSS/WS-EC/ECIMF

12

Draft, version 0.3
November 28, 2001

14

2	1. PURPOSE AND SCOPE.....	3
	2. BUSINESS CONTEXT MATCHING	3
4	2.1. CREATING THE BUSINESS CONTEXT MODELS	3
	2.2. CHECKING THE BUSINESS CONTEXT MATCHING	5
6	3. PROCESS MEDIATION.....	5
	3.1. CREATE BUSINESS PROCESS MODELS	5
8	3.1.1. <i>Identify the Business Transactions.....</i>	6
	4. SEMANTIC TRANSLATION	7
10	4.1. ACQUIRE THE SOURCE ONTOLOGIES	7
	4.2. SELECT THE KEY CONCEPTS.....	7
12	4.3. CREATE THE MAPPING RULES	8
	5. SYNTAX MAPPING.....	10
14	6. GENERATION OF MANIFEST	11
	7. IMPLEMENTATION: ECIML-COMPLIANT AGENT	13
16	7.1. DESCRIPTION AND LIMITATIONS	14
	7.2. RESULTS.....	14
18	8. SUMMARY.....	14

1. Purpose and scope

This document presents a step-by-step example of how the ECIMF can be used to prepare a set of recipes for interoperability between two e-commerce partners.

One partner, referred to as a Customer, produces Hi-Fi equipment of various sorts, and needs to ship them to the merchants. The other partner, referred to as Shipping Agency, offers services of shipping goods.

The Customer uses RosettaNet Implementation Framework 2.0 (RNIF) as his e-commerce interface, whereas the Shipping Agency uses EDI (EDIFACT D99.A).

This example follows the steps outlined in the Frameworks Integration Guidelines (see ECIMF-GM document).

2. Business Context Matching

In this step, two Business Context models are built and compared, in order to check whether they can match the expectations of the other business partner.

2.1. Creating the Business Context Models

The diagrams below have been built using REA modeling elements, here expressed as UML stereotypes.

(NOTE: they present only a subset of the full diagram! E.g. there should be a Resource:Payload and Resource:Labor which is transformed or used by the Events...)

Figure 1 presents the business context diagram for the shipping agency. Here are the key elements of that diagram:

- The agency expects the payment first, and only then delivers the service
- The roles of ShippingAgent and Cashier are split into two different entities (persons, divisions ...)
- ShippingAgent and Cashier collaborate with each other in order to satisfy the business rules (payment needs to be fulfilled first, and only then the shipment takes place)
- Both ShippingAgent and Cashier collaborate with the Customer.

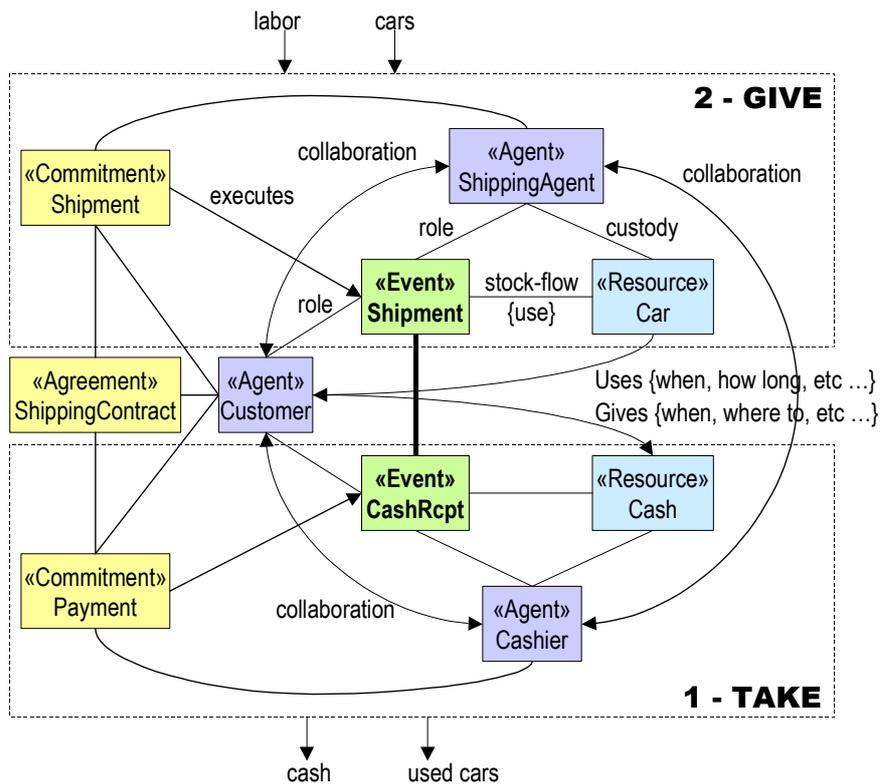


Figure 1 Business Context model as seen by the shipping agency.

2

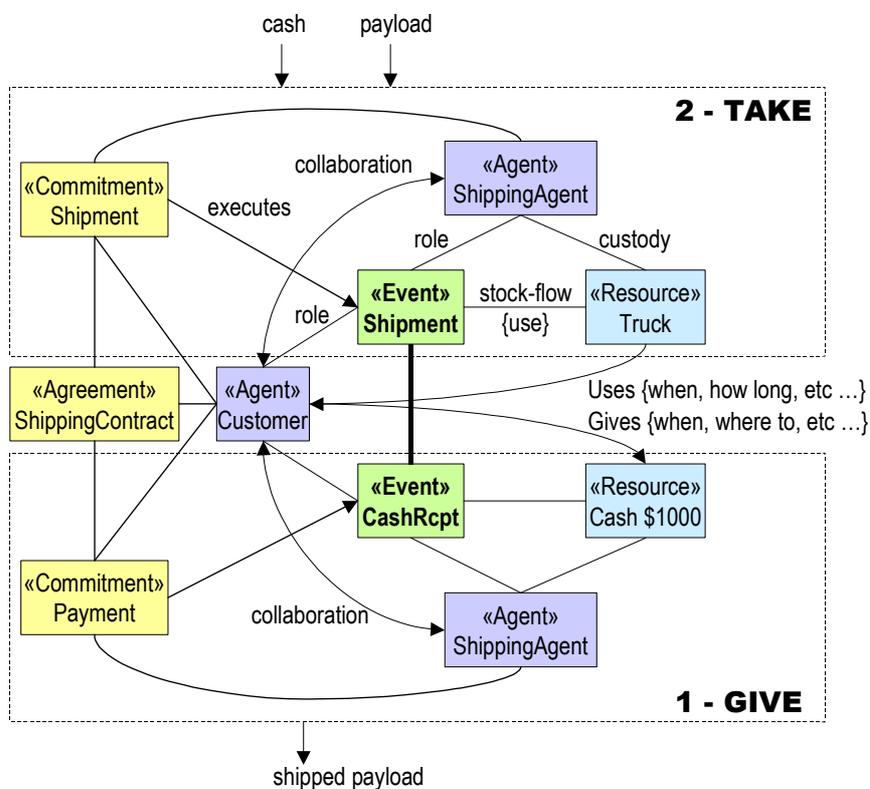


Figure 2 Business Context model as seen by the customer.

4

6

Now, for the customer the business context can be represented as shown on Figure 2. The key elements are:

- Customer expects first to give cash, then receive a service
- Customer wants to deal with the same entity for both events
- Customer has some specific demands on the kind of car, and the amount of cash.

2.2. Checking the Business Context Matching

From the diagrams above it is clear that in order for these two partners to be able to collaborate – in the traditional or in the electronic way – the following criteria have to be met (which ECIMF calls “business context matching rules”):

- *#1: Partners need to play complementary roles:* which is here the case. Note: although the Customer has a limited view of the Shipping Agency organizational structure (he wants to deal with just the ShippingAgent), it still has to be determined if he is able to deal with two separate persons/entities, which is required by the Shipping Agency (ShippingAgent and Cashier).
- *#2: Expected resources need to be equivalent:* in this case, parties need to agree on the exact kind of transportation used, and the exact amounts of money to be paid. They need to also agree on several additional properties of using the transportation (when, how long, from where, etc ...) and providing the payment (when, where to, what currency etc...).
- *#3: Timing constraints need to be mutually satisfiable:* in this case, the Customer is able to satisfy the requirement of the Shipping Agency that he needs first to pay. Further timing constraints may show up when analyzing the collaboration patterns between the parties.
- *#4: Transaction boundaries need to be preserved:* in this case, there are two transactions: payment and shipment, possibly consisting of several lower-level technical transactions. All supporting communication between the partners needs to be aligned in such a way that it preserves these boundaries for each of them.

After additional negotiations, we can state that these two Business Contexts match. These additional requirements identified in this step need to be recorded. (NOTE: how?)

For the sake of this example, we assume that both parties agreed to follow the model presented on Figure 1.

3. Process Mediation

3.1. Create Business Process models

Based on the Business Context models, we determined that the collaborations we are interested in are the following:

- **Payment collaboration task:** involving Customer and Cashier
- **Shipment collaboration task:** involving Customer and ShippingAgent.

Based on that, we should be able to identify concrete business processes existing within each organization, which support these collaborations. Also, it should be possible to identify the business transactions, which involve the electronic

communication between the partners, and sending of electronic business documents.

3.1.1. Identify the Business Transactions

For all collaboration tasks we need to describe two sets of transactions, each according to the framework used by the Agent. As an example, we will analyze in detail the Payment Collaboration Task.

The following table contains the example list of business transactions, together with their business documents, identified for the Customer:

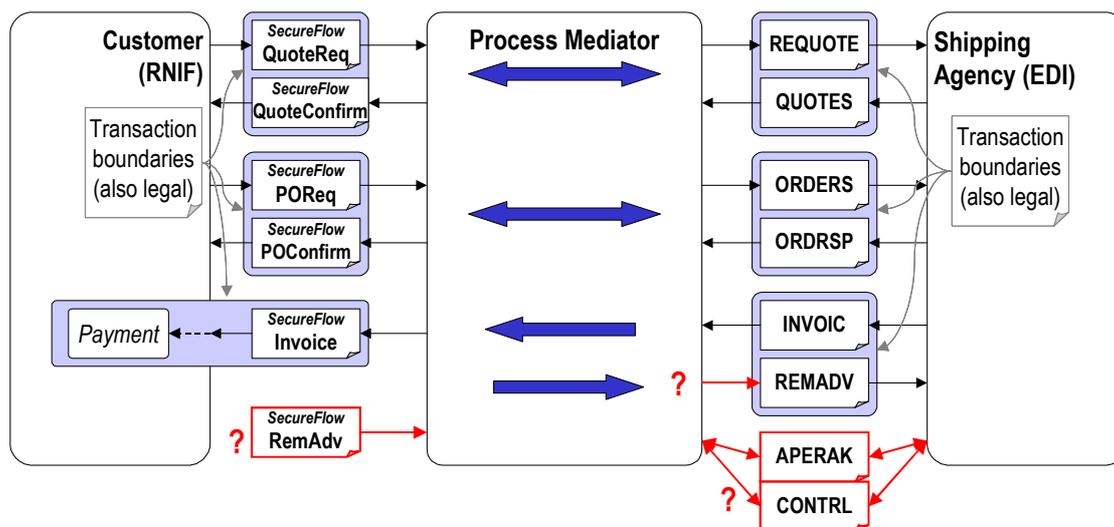
Party	Customer		
Collaboration Task	Payment Collaboration		
Framework	RNIF 2.0		
Transaction name	Initiator / Responder	Request document	Response document
PIP3A1: Request for quote	Initiator	QuoteRequest	QuoteConfirm
PIP3A4: Request Purchase Order	Initiator	PORequest	POConfirm
PIP3C3: Notify of Invoice	Responder	Invoice	
<i>PIP3C6: Notify of remittance advice</i>	<i>Initiator</i>	<i>RemittanceAdvice</i>	
<i>Message delivery control</i>	<i>any</i>	<i>Secure Flow</i>	

In a similar manner, we identify the transactions for the Shipping Agency:

Party	ShippingAgency		
Collaboration Task	Payment Collaboration		
Framework	EDIFACT		
Transaction name	Initiator / Responder	Request document	Response document
Request for quote	Responder	REQUOTE	QUOTES
PIP3A4: Request Purchase Order	Responder	ORDERS	ORDRSP
Notify of Invoice	Initiator	INVOIC	
Notify of remittance advice	Responder	REMADV	
<i>Message delivery control</i>	<i>any</i>	<i>APERAK, CONTRL</i>	

However, at this point we discover that the Customer's system doesn't implement the PIP3C6 – in the RosettaNet framework this is optional. We also discover that RosettaNet uses so called SecureFlows for communication control, whereas EDIFACT uses two messages: APERAK and CONTRL. We need to further study their semantics – see the section on Semantic Translation.

It is useful also to picture these collaborations in a common diagram. This is presented on Figure 3. The business transactions are shown here also, as rounded boxes containing the business documents. Areas of potential problems are marked with red color.



2 **Figure 3 Process Mediation for the Payment Collaboration Task.**

4. Semantic Translation

This step of integration helps to discover the underlying data model and the differences in meaning of the concepts used by each e-commerce framework. As it will be demonstrated, these differences will affect the design of both the process mediation and the syntax mapping.

For the sake of the example, let's assume that the customer wants to ship TV-sets from the factory to the shops.

This step will make use of the individual ontologies, a shared vocabulary and external resources in order to map between the key concepts in each of the frameworks.

Please note that generally the mappings are not symmetric, i.e. different rules and possibly different external resources need to be used when translating concepts from Customer to Shipping Agency than the other way around. For this reason, two sets of rules will always be present for each concept.

4.1. Acquire the source ontologies

For the purpose of this example, we acquired necessary concepts from each of the e-commerce frameworks – RNIF and EDIFACT respectively. We also made quite a few assumptions, which in the real case would have to be obtained from the particular IT system implementation, message implementation guidelines, product catalogues, company's procedures etc.

4.2. Select the key concepts

Let's start from the mapping of the two representations of a real-world entity (TV set), which is the subject of the shipment. These representations differ in each framework, because of their different scope.

This entity is represented in the ontology of the Customer as a TV-set – a kind of Hi-Fi equipment, while in the ontology of the Shipping Agency it is represented as a Box – a kind of Payload.

4.3. Create the mapping rules

The table below presents the semantics of the two corresponding concepts – TV-set in the Customer ontology, and Box in the Shipping Agency ontology – and the mappings required between the two representations, whenever they occur in the business documents.

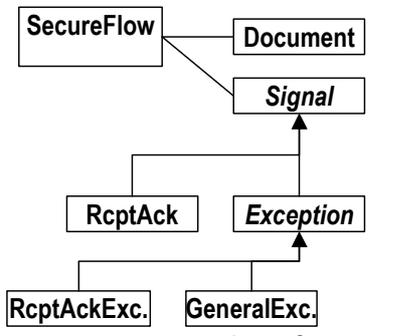
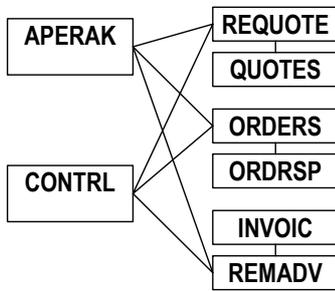
Customer: TV-set Properties	Semantic Translation Mapping Rules	Shipping Agency: Box Properties
Height Width Depth Represent the physical dimensions of the TV set chassis.	Tv_set → Box: dimension values will always be higher, but discrete. Need to be obtained from a cardboard box catalogue (external resource) Box → Tv_set: dimension values will always be lower. Need to be obtained from a TV products catalogue (external resource) using productID	Height Width Depth Represent the physical dimensions of the cardboard box used to ship the electronic equipment of any kind. The values are discrete, because only certain box sizes are available.
Not available (N/A)	Tv_set → Box: needs to be obtained from a product catalogue (external resource) Box → Tv_set: not needed	Weight Represents the weight of the box with the contents.
N/A	Tv_set → Box: needs to be obtained from a product catalogue (external resource) Box → Tv_set: not needed	StackingLevels Represents the number of levels the boxes can be stacked, one on top of the other.
N/A	Tv_set → Box: always set to True. Box → Tv_set: not needed	Fragile Marks the payload as fragile (requiring special care during transportation)
Color	Tv_set → Box: not needed Box → Tv_set: needs to be obtained from a product catalogue (external resource)	N/A
Stereo	Tv_set → Box: not needed Box → Tv_set: needs to be obtained from a product catalogue (external resource)	N/A
UnitPrice	Tv_set → Box: not needed Box → Tv_set: needs to be obtained from a product catalogue (external resource)	N/A
ProductID Product identification (type)	Tv_set → Box: concatenate with the serialNo Box → Tv_set: split into ProductID and serialNo, based on a required serialNo length.	ProductID Product identification (type), including serial number. Primary identification data
SerialNo Serial number. Primary identification data	Tv_set → Box: see rule above Box → Tv_set: see rule above	N/A

There are several interesting observations that can be made based on this example:

- Several external resources need to be consulted in order to prepare the mapping. It is possible to record the fixed values in the translation rules, but it would be more flexible to be able to query these resources dynamically, during run time.

- However, some of the values can be specified explicitly in the rules, and have fixed value (e.g. the `fragile` `Box` property).
- The translation rules are definitely not symmetric.
- There is a property, which uniquely identifies the corresponding physical entity (`Tv_set.serialNo` and `Box.productID`), although it is defined differently and requires processing.
- The properties related to physical dimensions are confusingly homonymous, although in reality their relationship is governed by a complex formula (and requires use of external resources).

Before proceeding to the last step (syntax mapping), let's analyze the message delivery control mechanisms, as these were identified as problematic during the process mediation step.

Customer (RNIF)	Semantic Translation	Shipping Agency (EDI)
 <p>SecureFlow consists of a business document (containing business data), and a responding business signal (acknowledgement).</p>	<p>The RNIF business documents map 1:1 to EDI business messages, e.g.:</p> <p>QuoteRequest ↔ REQUOTE QuoteConfirm ↔ QUOTES PORequest ↔ ORDERS POConfirm ↔ ORDRSP etc ...</p> <p>However, individual data elements can be missing, and will have to be collected from the previous messages, or supplied explicitly in the rules, or obtained from external resources.</p>	 <p>In this particular case, the EDI system uses APERAK and CONTRL messages only to signal exceptions. Acknowledgements are implicit, in the form of response business documents.</p>
<p>ReceiptAck This signal means that the document business data has been accepted for further processing (which implies also well-formedness)</p>	<p>RNIF → EDI: not needed – don't forward.</p> <p>EDI → RNIF: needs to be synthesized from the response document. Possible problems with timing constraints... (ack. too late)</p>	<p>N/A – implementation choice (positive acknowledgements are implicit).</p>
<p>ReceiptAckException This signal means the document was not well-formed (parsing errors). Business data was not considered at all.</p>	<p>The semantics of both messages is identical, which means a 1:1 mapping can be applied, both ways.</p>	<p>CONTRL This message is sent when parsing errors occur. Business data was not considered at all.</p>
<p>GeneralException This signal means that there were errors in the business data processing (though it means implicitly the document was well-formed).</p>	<p>RNIF → EDI: always map to APERAK</p> <p>EDI → RNIF: map only if the APERAK message carries an error status.</p>	<p>APERAK In this implementation, this message is sent only when an error occurs when processing business data (though it means implicitly the document was well-formed).</p>

Again, this analysis brings a couple of interesting observations:

- The differences in the semantics of message flow control mechanisms will affect the implementation of the process mediator, because some messages need to be created, removed, or sent at different times than the originating messages. Conclusion: there is no simple 1:1 mapping between messages, and the process mediator is really needed.
- The business documents map 1:1 in this example. However, as shown on the Figure 3, the RNIF side doesn't produce the `RemittanceAdvice` message, which the EDI side needs for completion of the low-level transaction. This message needs to be either synthesized by the process mediator (by accessing an external resource, such as the payee bank), or the RNIF side needs to implement it.
- The timing constraints for `ReceiptAck` (times defined in RNIF, which define how long the sender has to wait for an acknowledgement before concluding a failure) may be impossible to satisfy in this scenario. The EDI side doesn't produce required `ReceiptAck` signals, and they need to be created based on the response EDI messages – which may be sent too late to satisfy the timing limits defined in RNIF.

After completing this step, we are very well prepared to define the low-level syntax mapping – transformation of the data elements in individual messages.

5. Syntax mapping

According to the layered ECIMF model, the syntax mapping – i.e. the translation between the individual data elements – is the lowest layer of interoperability, and it is affected by the rules defined in all the higher layers.

Let's take for example a fragment of mapping between the `PurchaseOrderRequest` and `ORDERS`. Figure 5 shows the fragments of each message and the mapping links between the data elements.

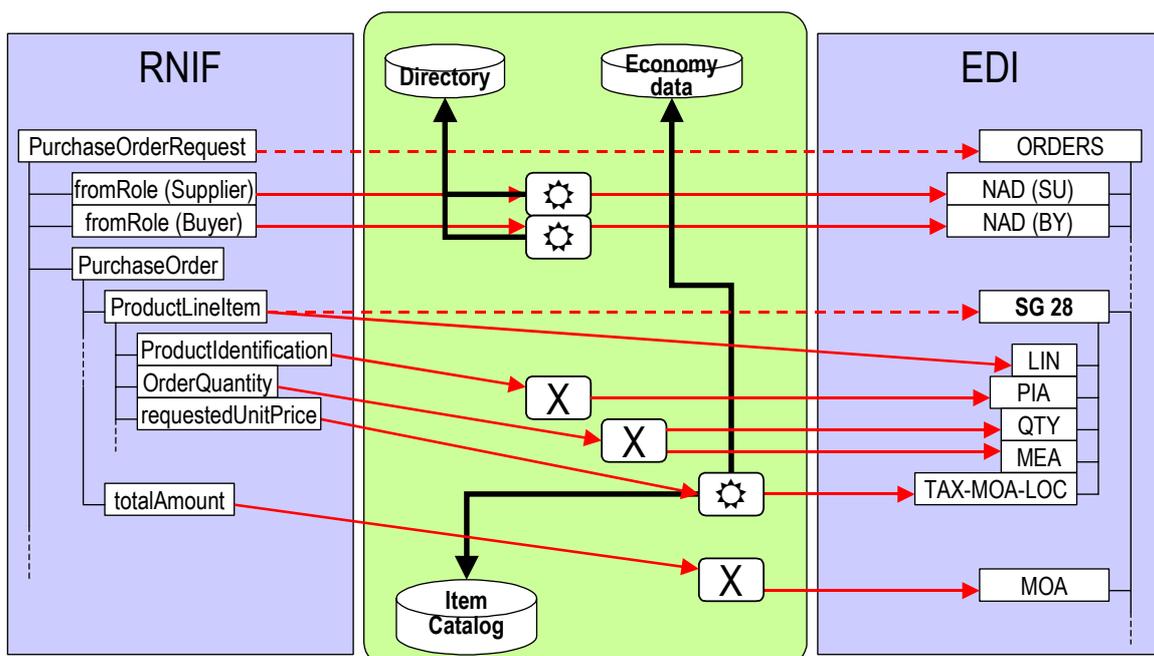


Figure 4 Message syntax mapping.

2 Again, a few observations can be made based on this example:

- 4 • This is a one-way mapping, as the arrows on the red links indicate. This means,
6 that this mapping is valid for translation of PurchaseOrderRequest messages into
8 ORDERS messages, and not necessarily vice versa (in fact, in our example
10 different external resources will be needed to perform the translation in the other
12 direction).
- 10 • The dashed lines represent the instance links, i.e. for each instance on one side a
12 corresponding instance on the other side is created. In this case, for one
14 PurchaseOrderRequest document one ORDERS message is created, and
16 similarly for one ProductLineItem one Segment Group 28 (SG28) is created.
18 Note, however, that additional limitations need to be considered here, which come
20 from the limitations on the allowed number of the given data elements in a
22 message. In this case, there can be no more than 200000 (according to EDIFACT
24 D99.A) occurrences of SG28 in a single ORDERS message. If there are more
26 ProductLineItems than that, they probably need to be divided into two
ORDERS messages – however, this changes significantly the flow of the low-level
transactions, as presented on the Figure 3.
- The boxes with a toothed wheel represent complex processing, with the use of
external resources. This is needed e.g. if the identification schemas for parties are
different, or in the above-mentioned example of different product classifications.
- The boxes with an “X” represent simple data transformation, like numeric or string
operations. E.g. as identified in the Semantic Translation step, the product ID
used in EDI (PIA element) needs to be a concatenation of the sub-elements of
the ProductIdentification element in RNIF.

28 In this step also the differences in the transport protocols and packaging are
30 considered. Some differences (like use of FTP vs. SOAP) will require providing
32 additional protocol parameters, e.g. FTP username and password, SOAP service
name, a WSDL file, details of the MIME packaging etc. Some of these parameters
can be expressed using ebXML CPP/CPA.

34

6. Generation of MANIFEST

36 As the final step, based on the models and transformation rules prepared in the steps
38 above, a MANIFEST needs to be generated - an abstract recipe for interoperability
between RNIF and EDI, within the given scope.

40 The example syntax of the MANIFEST document could look like the sample below:

```
42 <?xml version='1.0'?>
43 <Manifest>
44   <BusinessContextMatching name='Shipment'>
45     <BusinessContext id='WidgetsLtd'> ... </BusinessContext>
46     <BusinessContext id='JoeShipping'> ... </BusinessContext>
47   </BusinessContextMatching>
48   <ProcessMediation>
49     <Framework id='RNIF' name='WidgetsLtd'>
50       <BusinessProcessDefinition location='uddi:...PIP3A4...'/>
51       <BusinessProcessDefinition location='uddi:...'/>
52     </Framework>
53   </ProcessMediation>
54 </Manifest>
```

```

2       <BusinessProcessDefinition location='uddi:...'/>
</Framework>
4       <Framework id='EDI' name='JoeShipping'>
        <BusinessProcessDefinition>
            ... (here it follows, defined using ebXML BPSS)...
6         </BusinessProcessDefinition>
        </Framework>
8         <MediationRules>
            ...
10        </MediationRules>
</ProcessMediation>
12 <SemanticTranslation>
        <OntologyRef id='RNIF'>urn:ont1 ...</OntologyRef>
14        <OntologyRef id='EDI'>urn:ont1 ...</OntologyRef>
        <Rule id='rule1'>
16            <SourceCtxSet id='set1' />
            <TargetCtxSet id='set2' />
18            <formula id='formula1' />
            <formula id='formula2' />
20        </Rule>
        <ContextSet id='set1'><context id='ctx1' /></ContextSet>
22        <ContextSet id='set2'><context id='ctx2' /></ContextSet>
        <Context id='ctx1'>
24            <ConceptRef id='tv_set'>urn:...TV-set</ConceptRef>
        </Context>
        <Context id='ctx2'>
26            <ConceptRef id='box'>urn:...Box</ConceptRef>
        </Context>
        <Formula id='formula1'>
30            <body>
set2.ctx2.box.productID := set1.ctx1.tv_set.productID +
32            " " + set1.ctx1.tv_set.serialNo;
            </body>
34            </Formula>
        <Formula id='formula1'>
36            <body>
set2.ctx2.box.fragile := true;
38            </body>
        </Formula>
40    </SemanticTranslation>
    <SyntaxMapping>
42        <Mapping>
            <SourceMessage>PurchaseOrderRequest</SourceMessage>
44            <TargetMessage>ORDERS</TargetMessage>
            <Rules>
46            </Rules>
        </Mapping>
48        ...
    </SyntaxMapping>
50 </Manifest>

```

52 (This example uses the Semantic Translation ontology, developed for the purpose of
53 this project – see <http://www.ecimf.org/contrib/onto/ST/index.html> for more details).

54

55 Note that for the purpose of configuring the ECIMF-compliant runtime, only the
56 process mediation and syntax translation rules are needed. However, the models of
57 the two other layers are included as well in order to facilitate exchange of the ECIMF
58 models between the modeling tools, and to preserve the knowledge collected during
the process of mapping.

In the next step, as presented previously in the Figure 5, the ECIMF-compliant agent receives the MANIFEST and instantiates the necessary adapters. This may involve setting up processing pipelines for messages, creating state machines to keep track of complex interactions, creating translation maps for message elements, reading parameters provided by the communicating parties, etc. This reference environment for execution of the MANIFEST recipe can be provided as a commercial product.

Finally, at this stage it is possible for the parties to successfully establish business interaction, even though they use different e-commerce frameworks to express their activities.

7. Implementation: ECIML-compliant agent

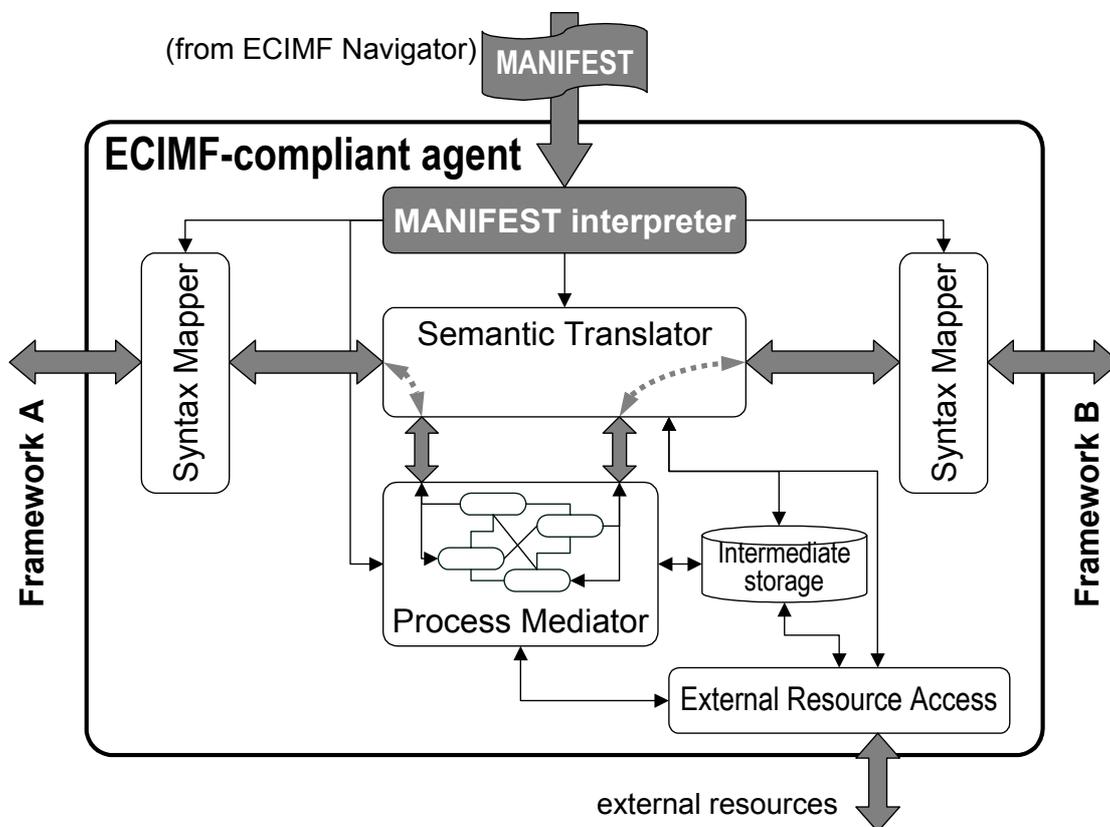


Figure 5 ECIMF-compliant agent implementation.

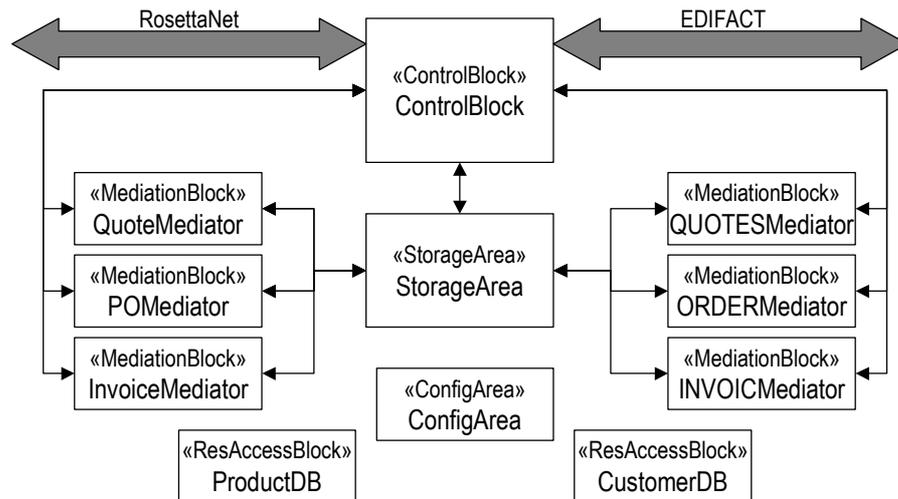


Figure 6 Process Mediator model.

2

4

7.1. Description and limitations

6

7.2. Results

8. Summary