

RosettaNet Methodology for Creating Choreographies

Released 11.00.00A

Specification Information	
Name	RosettaNet Methodology for Creating Choreographies
Publication Date	27 July 2011
Version Identifier	R11.00.00A

Table of Content

1.	Document Management	5
1.1	Legal Disclaimer	5
1.2	Copyright	5
1.3	Trademarks	5
1.4	MCC Phase 2 – Team Participants	5
1.5	Acknowledgments	5
1.6	Related Documents	6
1.7	Document Version History	6
1.8	Document Purpose	6
1.9	Notational Conventions	6
2.	Focus and Scope	7
3.	Current Industry Practice	10
3.1	Broad Variety of Solutions	10
3.1.1	Verbal Description	10
3.1.2	Verbal Description with Supporting Diagram	11
3.1.3	Proprietary Diagram with Description	12
3.1.4	Verbal Description with Standard BPMN Diagram	13
3.2	Improvement Potential	14
3.3	Expected Efficiency Gain	14
4.	Metamodel	15
4.1	From text to model	15
5.	The RosettaNet Choreography Approach	17
5.1	Choreography Purposes and Tool-Chains	18
5.1.1	Cartography Choreographies	18
5.1.2	Executable Choreographies	19
5.1.3	Sequential Multi-Party Choreographies	20
5.1.4	Tool/Model-Chains	21
5.2	Choreography Representation Elements	23
5.3	Choreography Grammar	34
5.3.1	Cartography Choreographies	34
5.3.2	Executable Choreographies	35
5.3.3	Sequential Multi-Party Choreographies	44
5.4	BPMN 2.0 Compliance	47

5.5	Choreography Profiles.....	49
6.	Extension Points	50
6.1.1	Glossary.....	51
7.	References.....	54
8.	Appendix	56
8.1	Classic Order to Cash Scenario - Cartography Style.....	56
8.2	Classic Order to Cash Scenario - Executable Style.....	61
8.3	Extended Order to Cash Scenario - Cartography Style	64
8.4	Extended Order to Cash Scenario - Executable Style	65
8.5	Order to Cash with LSP Scenario - Cartography Expanded Style.....	66
8.6	Order to Cash with LSP Scenario - Cartography Collapsed Style	68
8.6.1	Order to Cash with LSP –Global	68
8.6.2	OrderPlacement.....	69
8.6.3	ArrangeShipping.....	69
8.6.4	PerformShipping.....	69
8.6.5	Invoicing	70
8.7	Order to Cash with LSP Scenario - Expanded Sequential Multi Party Style...	71
8.8	Order to Cash with LSP Scenario - Collapsed Sequential Multi Party Style...	77
8.8.1	Order to Cash with Logistic Service Provider -Global	77
8.8.2	OrderPlacement.....	78
8.8.3	ArrangeShipping.....	78
8.8.4	PerformShipping.....	79
8.8.5	Invoicing	80
8.9	RosettaNet RIG 2A17 - Executable Style.....	81
8.10	RosettaNet RIG 3A1 - Collapsed Sequential MultiParty Style	83
8.10.1	Request-Quote-Customer.....	83
8.10.2	Quote-Exchange	84
8.10.3	QuoteAndAcknowledge	84
8.10.4	SFStockAndDebit	85
8.10.5	SeqMP-MultiPartyCollaboration.....	86

Table of Figures

Figure 1: A B2Bi Schema.....	7
Figure 2: Proprietary Process Model diagram for PIP 6A1	12
Figure 3: Diagram for Order Management in Japan scenario	12
Figure 4: BPMN Diagram highlighting Advance Ship Notice.....	13
Figure 5: Binary Cartography Choreography of PIPs	19
Figure 6: Multi-Party Cartography Choreography	19
Figure 7: Typical B2Bi Integration Architecture	20
Figure 8: Sample PIP Representations.....	23
Figure 9: Sample Start States.....	25
Figure 10: Sample End States.....	25
Figure 11: Types of Transitions	26
Figure 12: Sample Usage of Transitions.....	27
Figure 13: Representing Choreographies	27
Figure 14: Representing Decisions.....	28
Figure 15: Event-Based Choice Sample	29
Figure 16: Representing Parallel Structures	30
Figure 17: Sample Expanded Sub-Choreography with Implicit Role Mapping	31
Figure 18: Sample Expanded Sub-Choreography with Explicit Role Mapping	31
Figure 19: Sample Call Choreography with Explicit Role Mapping	31
Figure 20: Sample Scenario for an Interrupting Choreography Timer.....	33
Figure 21: Event-Based Choice Timer Scenario.....	33
Figure 22: Options for Starting Executable Choreographies	36
Figure 23: Options for Continuing PIPs	37
Figure 24: Options for Continuing Component Choreographies.....	38
Figure 25: Options for Continuing Event-Based Choices	40
Figure 26: Parallel Structure with Virtual Sub-Choreography	41
Figure 27: Parallel Structure without Virtual Sub-Choreography	42
Figure 28: Sample SeqMP Choreography	45

1. Document Management

1.1 Legal Disclaimer

RosettaNet, its members, officers, directors, employees, or agents shall not be liable for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document or the specifications herein, as well as associated guidelines and schemas. The use of said specifications shall constitute your express consent to the foregoing exculpation.

1.2 Copyright

©2011 RosettaNet. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the inclusion of this copyright notice. Any derivative works must cite the copyright notice. Any public redistribution or sale of this publication or derivative works requires prior written permission of the publisher.

1.3 Trademarks

RosettaNet, Partner Interface Process, PIP and the RosettaNet logo are trademarks or registered trademarks of "RosettaNet," a non-profit organization. All other product names and company logos mentioned herein are the trademarks of their respective owners. In the best effort, all terms mentioned in this document that are known to be trademarks or registered trademarks have been appropriately recognized in the first occurrence of the term.

1.4 MCC Phase 2 – Team Participants

We would like to recognize the following for their significant participation to the development of this Guideline.

Project Team Leader:

Dale Moberg
Jeff Hutchinson

Axway Inc.
Oracle

Lead Editor:

Hussam El-Leithy

RosettaNet / GS1US

Editors:

Andreas Schönberger
Pankaj Telang

Otto-Friedrich-University of Bamberg
Cisco

1.5 Acknowledgments

This document has been prepared by RosettaNet (<http://www.rosettanet.org/>) from requirements gathered during the Milestone Program and in conformance with the methodology. Listed below are the legal entities that contributed to the design and

development of this PIP.

Axway	Cisco
DHL	IBM
KJC Solutions	Oracle
OASIS	Software AG
Tibco	University of Bamberg
Vienna University of Technology	

1.6 Related Documents

- MCC ebMS V2 Profile R11.00.00A
- MCC ebMS V3 Profile R11.00.00A
- MCC Web Services Profile R11.00.00A
- MCC AS2 Profile R11.00.00A

1.7 Document Version History

<u>Version</u>	<u>Date</u>	<u>Description</u>
Released 11.00.00A	27 July 2011	Released Document

1.8 Document Purpose

The purpose of the document is to explain the different steps a user has to take to move from identifying their business processes writing them down visualizing them using choreography technology and ultimately identifying interface processes required for the business process.

1.9 Notational Conventions

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in (IETF, 1997).

2. Focus and Scope

This specification is about applying choreography technology to Business-to-Business integration (B2Bi). The purpose of choreography technology is the identification of the set of messages and the set of admissible message exchange sequences to be exchanged between communication partners. Applying choreography technology to B2Bi therefore means identifying the business documents and additional control messages to be exchanged between integration partners. Additionally, the special requirements of B2Bi have to be respected. Coordination upon the purpose of B2Bi projects not only is necessary between business process modelers and software engineers, but also between process modelers of different enterprises. Existing systems have to be reused, privacy issues of internal process management and implementation have to be respected, and heterogeneous computing resources must be assumed. Also, distributed IT infrastructure and unstable business relationships are commonly encountered in B2Bi projects. In consequence, a flexible and multi-layered approach is needed that fosters communication between business users and IT experts, users of different enterprises and that leverages the existing error-proof implementations of interaction scenarios.

To put this specification into scope, the abstraction layers of the B2Bi schema (taken from (Schönberger, et al., July 2008)) depicted in Figure 1 are briefly discussed.

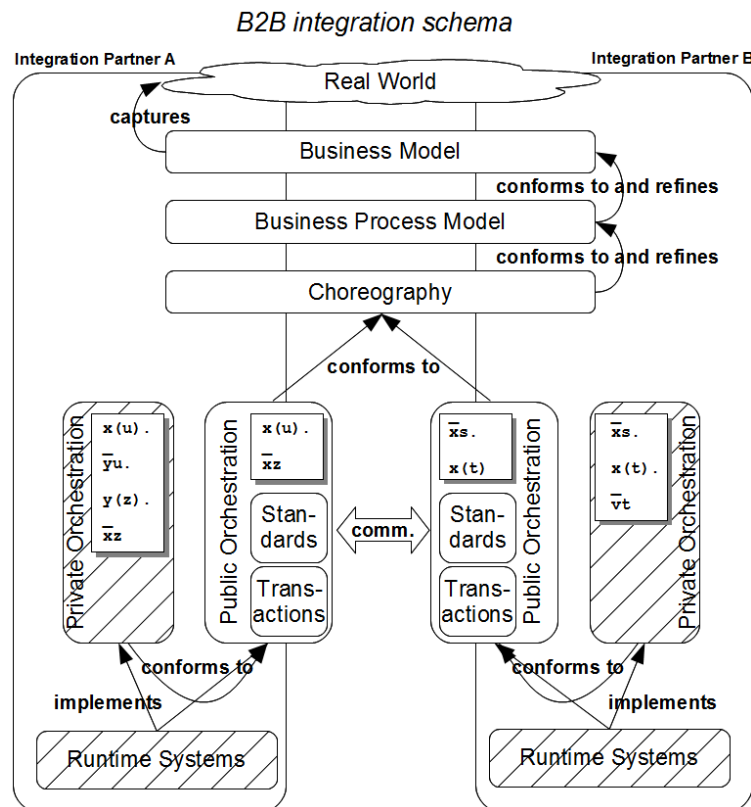


Figure 1: A B2Bi Schema

The schema shows six abstraction layers for capturing B2Bi processes from different perspectives and at different levels of detail. In an ideal world, a business model of a B2Bi scenario is first created for capturing its value proposition. Such a model then is to be refined by a business process model that identifies the type of information to

be exchanged, the tasks to be performed, and valid sequences of task executions from a global perspective. A so-called choreography model then should be used for refining a business process model by adding technical execution parameters, in particular message schema definitions and B2Bi relevant Quality-of-Service (QoS) parameters such as encryption and reliability. While a business process model includes the various activities for implementing the complete business process, the choreography layer focuses on the information exchange aspects. All three models should apply a global perspective in order to be a means for agreement and communication among personnel of integration partners. B2Bi connects information systems of distributed partners and therefore execution of local logic is an inherent B2Bi characteristic. So-called orchestration models can be used for describing the local behavior of participants using an executable specification format. While public orchestration models allow for focusing on the publicly visible message exchanges of an integration partner, private orchestration models allow for the specification of the full logic for integrating publicly visible message exchanges with internal business applications. Finally, the executable orchestration models have to be deployed to runtime systems.

This specification focuses on the choreography layer and deliberately disregards the details of orchestration models. This pays tribute to the large set of available realization options of choreography models. Also, orchestration models are not the only option for implementing B2Bi choreographies. However, the semantic gap between B2Bi business process models and implementation systems is very high and RosettaNet discovered that the B2Bi community is lacking a common terminology and methodology for discussing and representing the message exchange perspective of B2Bi projects.

This specification addresses this need by defining a visual model for capturing the message exchange perspective of B2Bi processes using choreography technology. This visual model can be used by process modelers of interacting enterprises to agree on the purpose, the contents and the procedures of B2Bi interactions. Additionally, this specification offers the possibility (but does not force upon you) to streamline the implementation of such choreographies by offering a modeling style with precise execution semantics. This means, that integration partners know exactly what the admissible behavior of their implementation systems is.

Further, this specification adopts common B2Bi terminology and concepts by reusing B2Bi standards that are known to the community. Most importantly, RosettaNet PIPs are eligible as atomic building blocks of B2Bi choreographies. This specification shows how PIPs can be represented in BPMN 2.0 choreographies ((OMG, January 2011), section 11), how complex multi-PIP compositions can be visualized and what the meaning of these visualizations is. Additionally, the relationship to adjacent abstraction layers is discussed, samples of common B2Bi scenarios are provided to exemplify the use of this standard and textual ebXML Business Process Specification Schema (ebBP, (OASIS, December 2006)) representations of these samples are given to show how choreographies can be refined with more technical detail.

Note that this specification is different from the RosettaNet Implementation Framework (RNIF, (RosettaNet, 2002)) and the RosettaNet PIP library. While the meta-model of PIPs is finite in the sense that there is a finite set of options for combining business documents and business signals, the set of B2Bi scenarios that can be built from multiple PIPs is by no means finite. Therefore, this specification provides a methodology for creating B2Bi choreography models and does not even try to enumerate the most common B2Bi scenarios. Yet, we encourage the B2Bi community to publish choreography models of their favorite scenarios.

This specification proceeds as follows:

Section 3 discusses the deficits of current industry practice in defining B2Bi collaborations. Section 4 discusses the abstract steps needed for turning a B2Bi business process model into a choreography model as well as subsequently into an implementation. Section 5 is the core of this specification and focuses on the choreography layer itself. It explains the rationale of the RosettaNet choreography methodology and gives the concepts and grammar for creating B2Bi choreographies. Section 6 discusses options for extending this specification and section 8 gives several samples that have been created according to this specification.

3. Current Industry Practice

3.1 Broad Variety of Solutions

In the past decade RosettaNet has noticed that companies combined PIPs to describe Business Collaborations. The RosettaNet PIPs are based on a standard which simplifies implementation; however the Business Collaborations are usually expressed by the companies in different ways. This causes a lot of confusion and requires additional resources in time and material for the companies who are asked to implement the Business Collaborations. Especially if some of the implementers are interacting with multiple companies all providing their own style of Business Collaborations description. There is also no standard for the level of detail provided in the Business Collaborations. The lack of standards creates an area where we see a lot of solutions and a lot of different styles and different level of granularity. All this is requiring a lot of money and time to be managed with and most companies have accepted the fact and usually use whatever their trading partner, in most cases the customer provides.

Below are a few examples how business collaborations are expressed by the industry.

3.1.1 Verbal Description

In this example we see a verbal description of the Business Collaboration from a RosettaNet Validation program for the Material Composition e-Business Processes (taken from (RosettaNet, 2006)).

The process includes the following steps:

- 1. The customer needs to know based on legal or market requirements the materials composition of a purchased product that might be included or are included in his design*
- 2. The customer defines the demands and defines the material composition information that is needed. The information needed might be based on guidelines decided by industry and/or based on a business-to-business agreement (B2B).*
- 3. The customer make an agreement with his supplier based on the defined demands that the supplier should provide a material composition information either based on guideline decided by a industry and/or based on a business-to-business agreement (B2B). The timing for this agreement differs between companies and thus out of scope for this milestone.*
- 4. The customer requests the agreed information from his supplier. The timing for this request differs between companies (e.g., to automatically be provided whenever a new product is released or to be provided only when the customer specifically requests it) and thus the timing of this request is out of scope for this milestone.*
- 5. The supplier might have requested information available in-house or the supplier might need to gather this information from his suppliers in his turn.*
 - 5.1. If the information is available in-house the supplier sends the requested information to his customer.*

- 5.2. *If the supplier needs to gather some or all of the requested information he uses the same business process as his customer. See items #2.--#4. When all required information is gathered the supplier compiles or aggregates the information and sends the requested information to his customer. Compilation or aggregation of in-house information will be done by the supplier's backend system and are out of the scope of this e-business process.*
6. *The customer receives the information. The customer might store the information in his in-house backend system to be able to provide his customer in his turn with compiled information. Storing and compiling in-house information will be done by customer backend systems and are out of the scope of this e-business process.*

3.1.2 Verbal Description with Supporting Diagram

In this example we see a verbal description with high level proprietary diagram of the Business Collaboration (taken from (RosettaNet, 2006)).

PIP 6A1 Notify of Service Contract Request enables the change request or query of service contract information from a contract requester to its contract issuer. Service Contract Request Notification is a suite of service contract update and query services.

By integrating using a PIP 6A1, Contract Requester can submit a change notification for the following functions:

- *Site to site move within a same contract*
 - *By site*
 - *By product*
- *Contract to contract move (site remains unchanged)*
 - *By site*
 - *By product*
- *Contract & Site move*
 - *By site*
 - *By product*
- *Contracts merge*

Contract Requester can also submit a query of the following:

- *Contract information query*
 - *By product serial number*
 - *By product number and product serial number*
 - *By product label and contract number*
 - *By maintenance purchase order number*
 - *By product purchase order number*

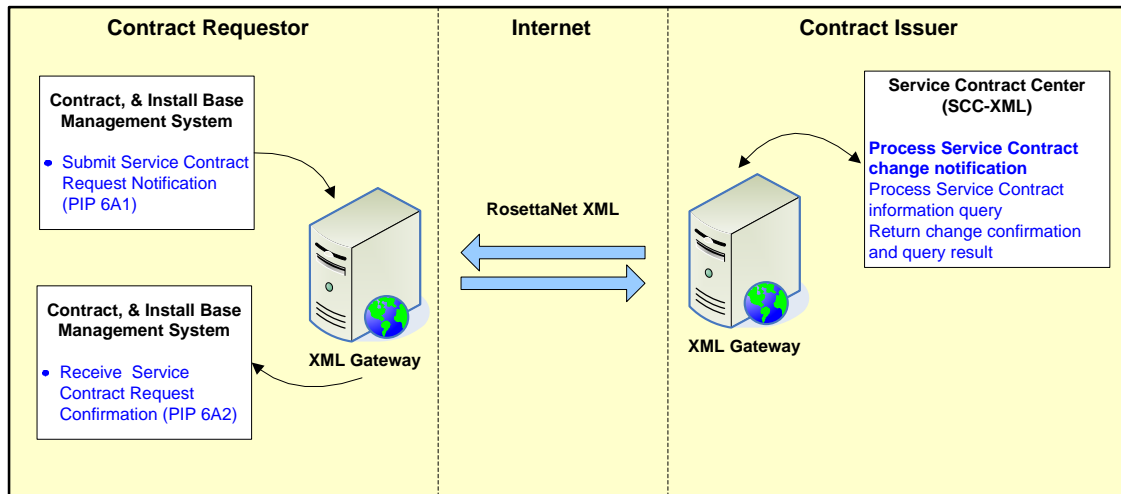


Figure 2: Proprietary Process Model diagram for PIP 6A1

3.1.3 Proprietary Diagram with Description

Business Collaboration with different pattern and proprietary diagram (taken from (RosettaNet, 2004)).

Here is an example of the business models with which order recipients supplies required stock to company or a third party owned warehouse based on demand information and purchase orders. The characteristics of this model are described as follows:

- Realize fine inventory and delivery control by order recipient based on
- Reduce stock space and physical inventory taking cost
- Apply special/custom-made products in addition to commercial products (since it is possible to clarify liability)

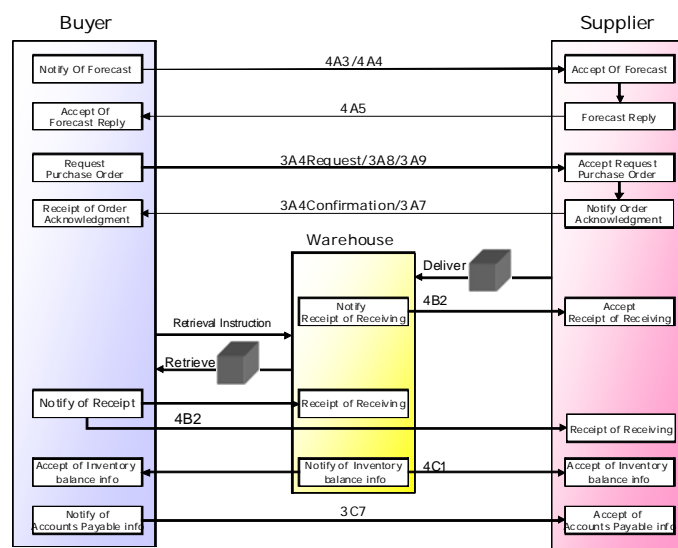


Figure 3: Diagram for Order Management in Japan scenario

3.1.4 Verbal Description with Standard BPMN Diagram

Verbal Description with high level BPMN diagram of the Business Collaboration (taken from (RosettaNet, 2003)).

The following is the primary Advance Ship Notice process implemented by this validation team; refer to Figure 4 (below) for the summary of the end-to-end process.

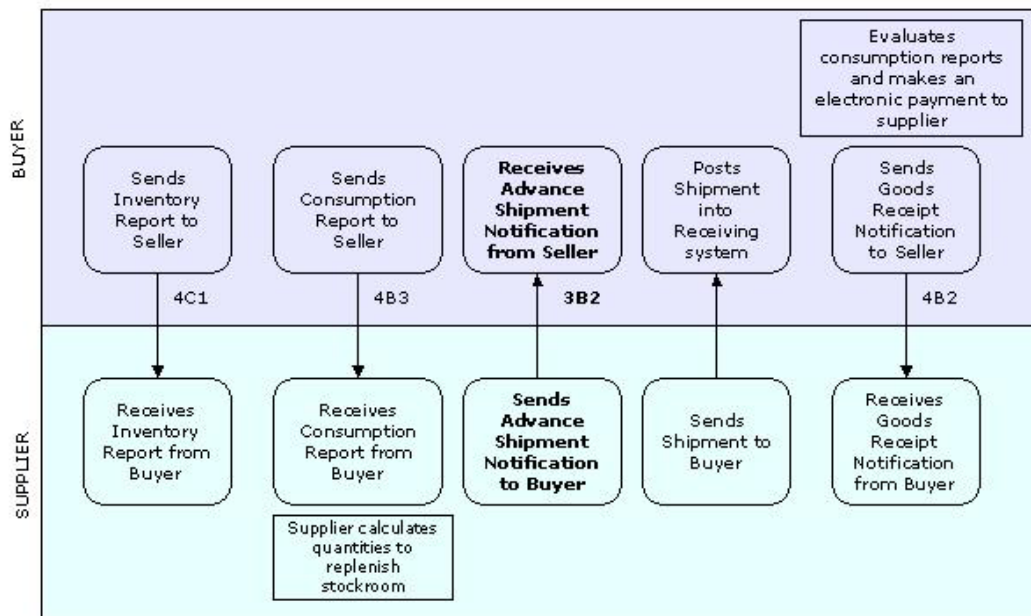


Figure 4: BPMN Diagram highlighting Advance Ship Notice

The process of the customers receiving an ASN from the sub-contractor with the customer looking to track orders versus shipments in transit. The PIP supports many different business scenarios where ship notices can be sent from a logistics provider, warehouse/hub, contract manufacturer or shipper to receiver or customer. The PIP can be used both for Raw materials, finished or semi-finished goods.

Validation team use case(s):

- *The Customer places an order with the Sub-contractor - sales person contact*
 - *Establish single or multiple line items for products, quantities and schedules*
- *The Sub-contractor receives the order and places into the system.*
 - *Acknowledges ability to support the order*
- *The Sub-contractor sends the ASN to the Customer:*
 - *When the Shipment is ready for shipment*
 - *Within a specified time frame after the shipment has left the dock.*
- *The Customer acknowledges the receipt of the ASN and uses the ASN to receive the shipment on the dock.*

In specific instances used to reconcile open PO line items with goods receipts and evaluate payments.

3.2 Improvement Potential

RosettaNet's approach to resolve this issue and intends to speed up the implementation process as well as to reduce the connection cost is to recommend the use of the standard described in this document. The document will explain how to create consistent Business Collaboration and how to describe them in a consistent way using existing standards and be able to describe critical parameters needed for implementation in a consistent way.

3.3 Expected Efficiency Gain

RosettaNet expects that the adoption of this methodology by the industry will result in long term savings from implementation caused by reduction in implementation time and simplification of the Business Collaboration description. RosettaNet also expects that the more complex Business Collaborations will generate higher savings and therefore very simple Business Collaborations may see very small or may not see any savings at all.

4. Metamodel

4.1 From text to model

In this section we will elaborate on that Midtown model and explain how to get from written process description towards business process model.

Step	Description	Input	Output
1	Determine the roles	Scenario description	Roles
2	Identify the business interactions and their temporal ordering	Scenario description	Interactions and their ordering
3	Identify the PIPs	Interactions, Roles	PIPs
4	Develop the cartography choreography by composing the PIPs	Interaction ordering, PIPs	Cartography choreography
5	Develop the executable choreography	Cartography choreography, scenario description	Executable choreography
6	Create ebBP model of the executable choreography	Executable choreography	ebBP model
7	Derive implementation from the ebBP model	ebBP model	Implementation

Step 1: Determine the roles

A cross-organizational business scenario description typically specifies the participants using terms like company, partner, and organization. If there is a single participant of its kind, then the scenario description usually specifies a unique name for it. In case we have multiple participants of the same kind, a scenario description may specify a role name. For each uniquely named participant, the business function it provides yields the associated role.

Step 2: Identify the business interactions and their temporal ordering

A scenario will usually specify the interactions between the participants using the generally accepted business terms such as quote, order, and invoice. Each interaction will generally involve two participants. The natural order of the scenario description yields the temporal ordering of the interactions.

Step 3: Identify the PIPs

RosettaNet organizes PIPs in a two-level hierarchy consisting of clusters and segments. A cluster represents a high level core business process. A cluster contains multiple segments each representing a sub-process of the cluster's core business process. Within each segment are individual PIPs.

This step identifies a PIP for each of the business interaction that Step 2 derives. First, based on the high level intent of the interaction, this step identifies a *cluster* to which the interaction belongs. Second, within the identified cluster, this step refines the categorization of the interaction to a *segment*. Third, within the segment, this step determines a PIP whose purpose matches the intent of the interaction.

Step 4: Develop the cartography choreography by composing the PIPs

This step composes the PIPs in the same order as that of the interactions to develop the cartography choreography. This choreography ignores any exceptions and only captures the happy path.

Step 5: Develop the executable choreography

This step refines the cartography choreography to develop the executable choreography. The executable choreography specifies the failure paths. A PIP is wired to a preceding PIP by a connector that specifies a condition as an XPath expression. A PIP execution may end in some other terminal state such as failure.

Step 6: Create ebBP model of the executable choreography.

This step creates ebBP model of the executable choreography. ebBP model captures low level details on the PIPs such as security requirements, timeout values, versions, and message formats.

Step 7: Derive implementation from the ebBP model.

This is the final step that derives endpoint implementation for the participants from the ebBP model.

An organization may use variants of the above methodology. In one variant, the methodology ends at Step 4 after developing the cartography choreography. The engineers start from the cartography as a high level specification of B2Bi requirements to develop the implementation. In another variant, the methodology ends at Step 5, and engineers use the executable choreography for developing the implementation. We expect that in future a tool chain will be available that facilitates executing the above methodology.

5. The RosettaNet Choreography Approach

The purpose of using choreographies in B2Bi projects is the provision of a means to capture the business document and business message exchanges between integration partners. This concerns the type and format of messages as well as the characterization of valid message exchange sequences. This focus distinguishes choreography models from business process models and orchestration models (cf. discussion of section 2). Business process models put the emphasis on the activities that have to be performed for realizing value exchanges between integration partners and orchestration models put the focus on the local implementation of message exchange sequences as well as the integration with business applications. The boundaries of choreography models and business process/orchestration models may be fluent and highly depend on the choice of choreography language. Choreography languages such as BPEL4Chor (Decker, et al., July 2007) realize the focus on publicly visible message exchanges by abstracting away orchestration model details. Conversely, other choreography languages such as the ebXML Business Process Specification Schema (ebBP, (OASIS, December 2006)) achieve a similar effect by deriving business document and message exchanges from business process models. The semantic proximity to business process models or orchestration models is not the only criterion that discriminates between choreography languages. While BPEL4Chor or ebBP offer textual notations that easily can be used as interchange format or as basis for automated processing, other languages such as BPMN 2.0 (OMG, January 2011) choreographies, the Business Choreography Language (BCL, (Zapletal, et al., September 2009)) or the UN/CEFACT's Modeling Methodology (UMM, (UN/CEFACT, 2006)) offer visual notations that offer advantages in terms of usability. A major difference between BPMN 2.0 and BCL/UMM is that BCL/UMM offer B2Bi domain-specific concepts whereas BPMN 2.0 choreographies strive for offering a general-purpose choreography language. For a more detailed review on the nature of choreography languages, see (Schönberger, February 2011).

The diversity of choreography languages indicates that there is no single language that fits all purposes. The choice of choreography language depends on the purpose of modeling/specification, on the languages and methods used for capturing business process/orchestration models, and on available tooling. For this specification, a compact visual representation that is accessible to business analysts, standardization, straightforward representation of RosettaNet PIPs and PIP performance controls, amenability to automatic processing, and standardization are the most important criteria for choosing a choreography language. Therefore, the use of BPMN 2.0 choreographies is recommended for capturing B2Bi choreographies visually and the use of ebBP is recommended as serialization format.

Nonetheless, this specification acknowledges the need for a diversity of choreography languages and tool chains that depend on the settings of a particular B2Bi project. Therefore, specifying the usage of alternative choreography languages for capturing B2Bi choreographies is explicitly allowed for by means of a profile mechanism. The constraints to be met by MCC phase 2 choreography profiles are described in section 5.5.

The concept of this specification is fundamentally different from the specification of PIPs (cf. section 2). PIPs describe standardized processes for exchanging a single business document. This specification does not standardize processes. Instead, the methodology to model and agree upon complex PIP choreographies is specified.

The rest of this section is structured as follows. Section 5.1 describes the different purposes that choreographies are used for in this specification. "Cartography"

choreographies, “Executable” Choreographies and “Sequential Multi-Party” Choreographies are the different choreography styles that reflect these purposes. Section 5.2 introduces the various visual elements that are used to represent choreographies. Sections 5.3, 5.3.2 and 5.3.3 elaborate on the characteristics and grammar rules of cartography, executable and sequential multi-party choreographies respectively. Section 5.5 gives the basic requirements to create an MCC profile for additional choreography languages.

5.1 Choreography Purposes and Tool-Chains

In B2Bi projects, the system to be built is subsequently modeled at several abstraction layers such as those depicted in Figure 1 (page 7). Each new model helps in bridging the semantic gap between the real world partner interactions and the actual communication systems.

For example, a business model helps in bridging the semantic gap between the real world as is and a business process model by describing the value exchange relationships between integration partners. Value exchange relationships may be modeled as categories of product, service and payment exchanges or as contractual obligations that result from interactions. A business process model then can use such a business model to identify the activities that are necessary for implementing the value exchanges. Clearly, business models are not created for every B2Bi project and project teams may opt to directly create business process models. However, the semantic gap to be overcome when creating a business process model then is significantly higher as there is no business model that serves as an intermediate layer.

Bridging the semantic gap between business process models and orchestration models can be considered to be the purpose of choreography models. However, a large variety of business process model notations, orchestration technologies and development methodologies may be applied to B2Bi projects. In other words, the semantic gap between business process models and orchestration models cannot uniquely be characterized and the subsequent steps for closing the gap depend on the development methodology. In consequence, the way choreography technology is used may widely vary with the specific settings of B2Bi projects. The know-how and skill-set of personnel that is assigned to the task of creating choreographies may be very different.

The choreography styles described in sections 5.1.1, 5.1.2 and 5.1.3 capture different purposes that choreographies are used for and briefly reflect on the typical personnel to use the choreography style. Section 5.1.4 puts the choreography styles into context with tool/model-chains for deriving orchestration models from business process models.

5.1.1 Cartography Choreographies

Cartography choreographies put the emphasis on supporting discussions among personnel of the integration partners at the business level. For cartography choreographies, identification of roles and business document types is much more important than the exact definition of control flow, message formats and error handling. (Semi-) automatic derivation of implementation artifacts is of subordinate importance.

In consequence, there are only basic and easy-to-follow rules for creating cartography choreographies. Typically, just the happy paths through an interaction

are specified whereas error paths are only included if they are of major importance. For example, the cartography choreography in Figure 5 just expresses that PIPs 3A19, 3A20, 3B2 and 3C3 are to be used between a Customer and Supplier role and that the typical sequence is (3A19, 3A20, 3B2, 3C3). Whether or not PIP 3A20 always has to be performed or the reactions to erroneous 3B2 instances are not specified. Note that incorporating more precise choreography models within cartography choreographies is well acceptable. In Figure 6, the composition of a multi-party cartography choreography from several component choreographies is visualized. There is no reason why an existing full-featured definition of 'Arrange-Shipping' should not be reused.

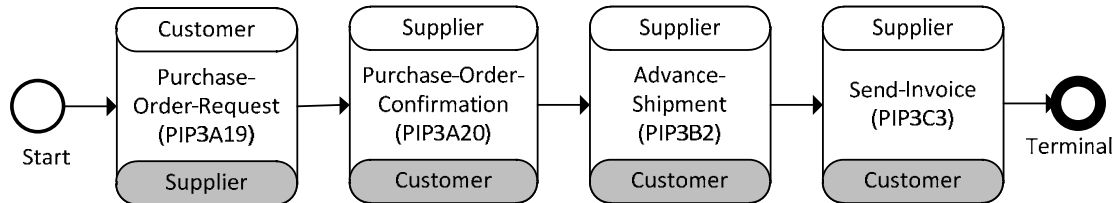


Figure 5: Binary Cartography Choreography of PIPs

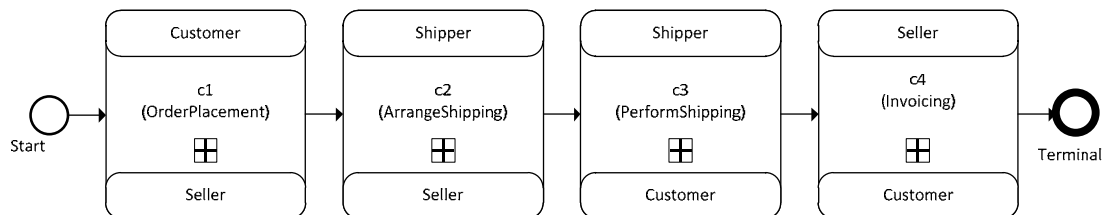


Figure 6: Multi-Party Cartography Choreography

The downside of cartography choreographies is that turning a particular cartography choreography into an implementation requires tight interactions between process modelers and software engineers as a lot of parameters are left unspecified. It may even turn out that the choreography has to be reworked at the business level because it is technically infeasible. Similarly, checking conformance of implementation models or runtime traces to the choreography is limited.

Cartography style choreographies are envisioned to be used as a communication means by business analysts and business modelers to identify the relevant PIPs and the ideal flow of PIP executions. Software engineers would use such a specification as (incomplete) requirements gathering for engineering the actual interactions.

5.1.2 Executable Choreographies

Executable Choreographies reflect integration settings that are very common to B2Bi and strive for providing advanced implementation support that fosters quick implementation and cross-partner interoperability. For this purpose, an integration architecture as depicted in Figure 7 is assumed where integration partners use so-called Business Services Interfaces (BSIs) for controlling the exchange of business documents whereas the actual application logic is encapsulated in backend systems or business applications. This basic integration architecture is frequently used by RosettaNet implementers. Executability of choreographies refers to the fact that complete implementations of BSIs can be derived from a particular choreography such that all message exchange sequences that conform to it are accepted by the

generated BSIs. To do so, choreography elements such as PIPs are translated into BSI implementation artifacts that interact according to standardized protocols. Standard interfaces are used for decoupling BSIs from each other and from the internal business applications/backends.

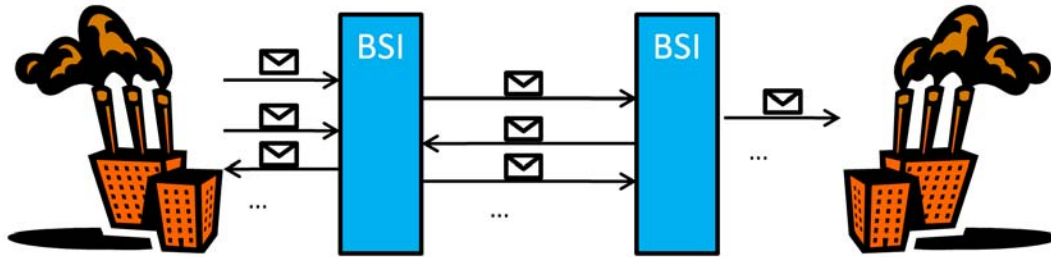


Figure 7: Typical B2Bi Integration Architecture

Executable choreographies are not necessarily used to really create BSI implementations. In practice, a full integration with existing backend systems or business applications will still require the aid of software engineers. But, the BSI implementations actually provided by integration partners can be compared to derived BSI implementations for checking conformance to the choreography, i.e., derived BSI implementations can be used as test or reference systems. This ability can be of great help for ensuring interoperability between partner systems. Samples of how executable ebBP choreographies may be designed and how WS-BPEL (OASIS, 2007) based BSI implementations can be derived are available in (Schönberger, et al., 2010) and (Schönberger, et al., November 2010).

The downside of executable choreographies is that more strict rules for modeling choreographies apply. The number of integration partners is limited to exactly two, control flow expressiveness is limited and all PIPs of a particular choreography have to be specified. Details are given in section 5.3.2.

The foundation for leveraging executable choreographies in this specification is the analysis of RosettaNet's Implementation Guideline (RIG) library. An important result is the observation that the vast majority of the RIG B2B interactions are limited to exactly two integration partners and that control flow definition is not very complex.

Executable choreographies are envisioned to be used as a means for technically skilled business modelers or software engineers who participate in the choreography definition process to constrain the set of valid implementations significantly.

5.1.3 Sequential Multi-Party Choreographies

Sequential Multi-Party Choreographies (SeqMP choreographies) provide a means for capturing and analyzing sequences of executable choreographies. This reflects the situation that real-world production and service delivery processes require the cooperation of several partners while the majority of B2B interactions is defined on a bilateral basis. SeqMP choreographies not only can be used to compose multi-party choreographies from binary executable choreographies, they also serve as basis for detecting synchronization deficits that emerge from multiple parties interacting with each other. An example of such an information deficit can be identified in Figure 6 (page 19). Assume the 'ArrangeShipping' component choreography carried out between the Supplier role and the Seller role fails. Then it is not clear how the Customer role is notified about the premature termination of the overall choreography. However, the Customer role may wait for the start of component choreography 'PerformShipping' because she ordered products in component

choreography 'OrderPlacement'. While such synchronization deficits barely can completely be avoided, SeqMP choreographies offer so-called 'escalation sets' to capture them. Escalation sets are sets of top-level choreography roles and can intuitively be characterized as follows:

"If a role has already participated in the overall choreography and may participate in the future and is not participating in the current component choreography and if a transition is taken that excludes that particular role from further participation, then the role is to be included in the escalation set of that particular transition."

Note that the calculation of such escalation sets relies on the availability of synchronized result values of the component choreographies, i.e., the participants of component choreographies must have agreed upon its outcome. That is the reason why only executable choreographies are admissible as component choreographies of SeqMP choreographies. A SeqMP choreography or a cartography choreography are not admissible as a SeqMP component choreography.

SeqMP choreographies are envisioned to be used as a means for technically skilled business modelers or software engineers who participate in the choreography definition process to discuss the effects of bilateral interactions on multi-party settings and to detect synchronization deficits.

5.1.4 Tool/Model-Chains

Tool-chains or model-chains support the subsequent phases of a development process. Each tool or model helps in closing the semantic gap between the real-world B2Bi scenario and its implementation. The different choreography languages and choreography styles described above can be applied in various ways. The choice of a particular choreography language and style also depends on the choice of business process modeling approaches, the actual implementation artifacts and the personnel on the project. So, this specification proposes the following two model chains as modeling options that neglect the business process layer and do not preclude alternative approaches:

1. BPMN-ebBP-Implementation Chain

- a. *Create cartography choreographies as BPMN choreographies*
This serves for discussing the relevant PIPs to be used and to capture the happy path of the interaction.
- b. *Create executable choreographies (as BPMN choreographies) to refine cartography choreographies*
This serves for specifying the failure paths of the choreography and for agreeing upon what effect the result of a PIP execution has. Beginning with this step and leaving out cartography choreographies may be acceptable in certain circumstances.
- c. *Derive ebBP representations of the executable choreographies*
This serves for pinning down PIP performance controls such as security requirements, timeout values and the exact message formats and versions to be used.
For simplistic scenarios with very few PIPs starting with this step may be acceptable.
- d. *Derive the implementation from the ebBP representation*
There are various ways of implementing PIP compositions, most notably the MCC phase 1 Web Services and AS2 profiles.

2. BCL-Implementation Chain

- a. *Create cartography choreographies as BCL choreographies*
This serves for discussing the relevant PIPs to be used and to capture the happy path of the interaction.
- b. *Create executable choreographies including PIP performance controls (as BCL choreographies) to refine cartography choreographies*
This serves for specifying the failure paths of the choreography, for agreeing upon what effect the result of a PIP execution has and for pinning down PIP performance controls. Beginning with this step and leaving out cartography choreographies may be acceptable in certain circumstances.
- c. *Derive the implementation from the BCL representation*

5.2 Choreography Representation Elements

This section describes the BPMN Choreography (OMG, January 2011), section 11) constructs that are used for representing RosettaNet MCC choreographies using a series of “Construct Advices”. Although some samples in this section will comprise more than one construct, the actual interplay of constructs as well as composition rules are contained in section 5.3. In particular, constraints on the use of elements or names of elements may be tightened for ensuring executability or analyzability. Therefore, such tightened constraints typically will be listed in sections 5.3.2 or 5.3.3.

Construct Advice 1: Representing PIPs

RosettaNet PIPs are the atomic building blocks of RosettaNet MCC phase 2 as MCC phase 1 already specifies all the details of performing a single PIP. The most important things to be represented for a PIP are its type, its activity id as well as the role assignment.

The PIP type corresponds to the PIP identifier that follows RosettaNet's cluster/segment/PIP taxonomy whereas the activity id is a name for referencing the execution of a particular configuration of a PIP within the choreography. The activity id is important for distinguishing between several uses of the same PIP type within the same choreography. For example, the exchange of a purchase order (PIP 3A19) may be used for different purposes (*placing a purchase order to be answered within 3 days* and *placing an additional order to be answered within 12 hours*) within the same choreography and different PIP performance controls may be assigned (i.e. the *TimeToPerform* parameter and different security parameters if need be). In this situation, the activity id helps in distinguishing the two different uses. Finally, the mapping of the choreography roles to the PIP roles must be defined, i.e., it must be clear which of the choreography roles is assigned the requester role or responding role of the PIP respectively.

For the following explanations, note that a PIP definition distinguishes the *communication role* (requester or responder) from the *functional role* (say, Buyer or Seller). Whereas the communication role determines the sender and receiver of the business document respectively, the functional role characterizes the business meaning of dealing with the document (the Buyer commits to buying the items listed in the business document whereas the Seller commits to delivering the items). In the RosettaNet methodology, there is a one-to-one relationship between the communication roles and the functional roles of a PIP, i.e., a functional role is assigned either the requester role or the responding role.

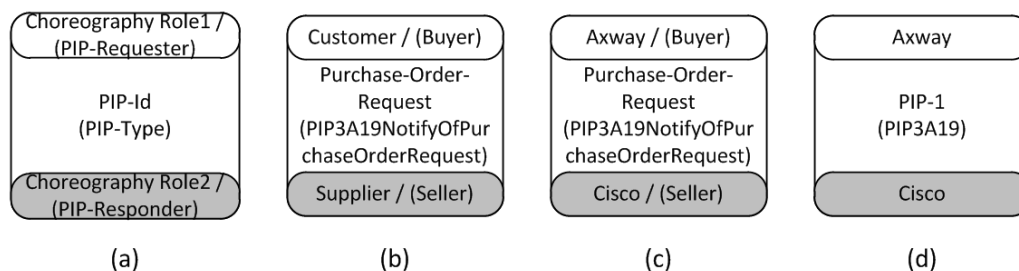


Figure 8: Sample PIP Representations

Figure 8 (a) shows an abstract characterization of how to represent a PIP as a BPMN 2.0 *choreography task*. The center of the rounded rectangle carries the activity id ('PIP-Id') followed by the actual type of the PIP in parentheses. The two

choreography roles that perform a PIP are listed within two *bands* at the top and at the bottom of the rectangle. The PIP functional roles that the choreography roles play can be listed after a slash character within parentheses. The PIP communication role that a choreography role takes is specified by coloring the bands. The white band identifies the requester role whereas the grey band identifies the responder role. The position of the bands is insignificant, i.e., you could have a grey band at the top and a white band at the bottom as well. As there is a one-to-one relationship between functional roles and communication roles of a PIP, the mapping to PIP functional roles within the bands can be omitted. For example, look at Figure 8 (b) that exemplifies the specification of performing PIP 3A19. The choreography role 'Customer' is mapped to the PIP functional role 'Buyer'. But, as the Buyer role is defined to take the requester role in the definition of PIP 3A19 and as the Customer role is assigned the requester role in Figure 8 (b) by being put into the white band, it is clear that the Customer role takes the Buyer role without explicitly writing that down.

Figure 8 (c) and (d) express the exact same information as Figure 8 (b). The only difference is the style of specification. Instead of "anonymous" choreography roles such as Customer and Supplier, "scenario specific" roles such as 'Axway' and 'Cisco' could be used. From a technical perspective, this does not make a difference. If 'Software AG' was to interact with 'Cisco' then Software AG could either be declared to take the Customer role or the Axway role. From a modeling perspective, however, assigning Software AG the Axway role may be confusing for the reader. Similarly, there is no technical reason why choreography role names should be required to be distinct from PIP role names. It is perfectly acceptable to map a choreography Seller role to a PIP Seller role.

For the definition of activity ids, any string that does not contain special characters is admissible. Additionally, activity ids **MUST** be unique relative to the enclosing choreography.

For the definition of PIP type names, this specification **REQUIRES** that the type begins with the string 'PIP' followed by the cluster/segment/PIP identifier that complies to the RosettaNet PIP taxonomy. Subsequently, any string that does not start with a digit and does not contain spaces or special characters may be appended.

Construct Advice 2: Representing Start States

Start states represent the entry points into choreographies and are visualized using BPMN *Start Event* nodes as depicted in Figure 9. By following the outgoing transitions of start states the initially admissible PIPs of a particular choreography can be identified. This specification allows for but does not explicitly discuss more sophisticated types of starting processes or choreographies as specified in BPMN. That means that adopters of this specification may leverage alternative BPMN start events ((OMG, January 2011), section 11.5.1) if needed although these are not discussed here.

Note that start states should carry a name that starts with 'Start' for better readability although this specification neither requires the use of a name nor a particular format. Therefore (a), (b) and (c) of Figure 9 all are acceptable. The only limitation is that special characters **MUST NOT** be used.

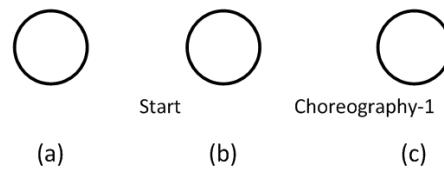


Figure 9: Sample Start States

Construct Advice 3: Representing End States

End states represent the termination of choreographies and may indicate different outcomes. End states as used in this specification do not express any actions, but just define that no more PIPs will be performed. Therefore, basic BPMN *End Events* are used for representing end states. End states can be distinguished from start states by the following two properties:

- In conformance to the BPMN specification, end states have “thick” lines (OMG, January 2011), section 11.5.3) whereas start states have “thin” lines.
- End states only have incoming transitions whereas start states only have outgoing transitions.

This specification allows for but does not explicitly discuss more sophisticated types of end states as specified in BPMN. That means that adopters of this specification may leverage alternative BPMN end events (OMG, January 2011), section 11.5.3) if needed although these are not discussed here.

There are no naming rules for end states except that special characters **MUST NOT** be used. Also, end state ids **MUST** be unique relative to the enclosing choreography. Hence, the labels in Figure 10 (a) – (c) all are acceptable end state representations.

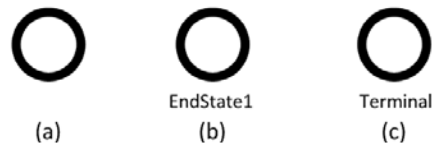


Figure 10: Sample End States

Construct Advice 4: Representing Transitions

Transitions connect the various states or nodes of choreographies and are crucial for specifying control flow. The interpretation of transitions is essential for defining the semantics of a visual language. This specification is dedicated to B2Bi semantics that follow the RosettaNet methodology and adopts this semantics for specifying control flow. Therefore, this specification uses *Sequence Flows* as defined in the BPMN standard for connecting the various choreography elements, but does not fully comply with the BPMN general considerations (OMG, January 2011), section 8.3.13) and choreography considerations (OMG, January 2011), section 11.3.1) for *Sequence Flows*. The details of interpreting transitions are given in sections 5.3.1 to 5.3.3 and the visual representations of different transition types are described here.

Figure 11 shows the four basic types of transitions that are used in this specification. All types of transitions are directed which implicitly means that the source state is left and that the target state is entered upon “*firing*” a transition. A transition of type

(a) may be fired when the subsequent PIP or choreography is started, when a related timer runs out or when leaving any state is to be specified without elaborating on the conditions for moving from one state to another. Transitions of type (b) and (c) carry condition expressions that are used to evaluate the result of a PIP execution. That means that a transition only can fire if the expression evaluates to true. The definition of a condition expression consists of the name of the expression language separated from the actual expression by means of a colon. The expression language CGV (ConditionGuardValue) is taken from the ebBP standard that defines generic protocol outcomes applicable for RosettaNet PIPs, such as 'AnyProtocolFailure' or 'ProtocolSuccess'. The computation of those ConditionGuardValues is specified in the MCC phase 1 deliverables. As an alternative to using generic protocol outcomes, expression languages for evaluating the contents of the exchanged business documents can be used. Note that evaluating the contents of a business document only is valid if the PIP succeeded from a protocol perspective. Following the concept outlined in ebBP, XPath2 (W3C, 14 December 2010) is suggested as important evaluation language of XML based content. However, as Figure 11 (c) suggests, XPath2 expressions may be way too complex to be included in a visual model. This specification leaves this problem of visualizing valid and complete XPath2 expressions to tool implementations. Apart from CGV and XPath2, the use of alternative expression languages can be agreed upon by integration partners.

The CRes (short for CollaborationResult) language is used for evaluating the result of choreographies (or collaborations in terms of ebBP and UMM) instead of PIPs. The admissible results of a choreography are defined to be the names of the end states of the choreography. Transitions that evaluate the result of choreography additionally may carry an *escalation set* definition in curly braces. An escalation set is a set of choreography roles that may suffer from synchronization deficits upon firing a transition. Details are given in section 5.3.2 and 5.3.3.

Finally, this specification leaves it open to integration partners to define project specific labels. For example, the Trigger-Guard-Effect notation as defined in the UML standard (OMG, May 2010), section 15.3.14; sometimes referred to as Event-Condition-Action notation) could be agreed upon by the partners.

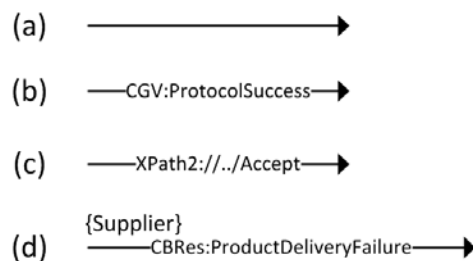


Figure 11: Types of Transitions

Figure 12 shows a basic sample of using transitions. The incoming transition on the left-hand side of the PIP2A1 choreography task is fired when the 'InfoDistributor' triggers the PIP (as the InfoDistributor is assigned the "initiator band" of the task). On the right-hand side, two transitions are used to evaluate the PIP result using the CGV expressions 'AnyProtocolFailure' or 'ProtocolSuccess'. The determination of such CGV expressions has been defined in MCC phase 1. Depending on the result PIP 2A1, either the end state called 'ProtocolFailure' or the end state 'ProtocolSuccess' is entered upon completion of the PIP execution protocol. Note that both choreography roles follow the same transition as the result of a PIP always is synchronized by

means of MCC phase 1 technology.

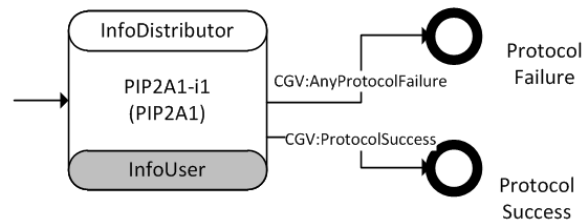


Figure 12: Sample Usage of Transitions

Construct Advice 5: Representing Choreographies

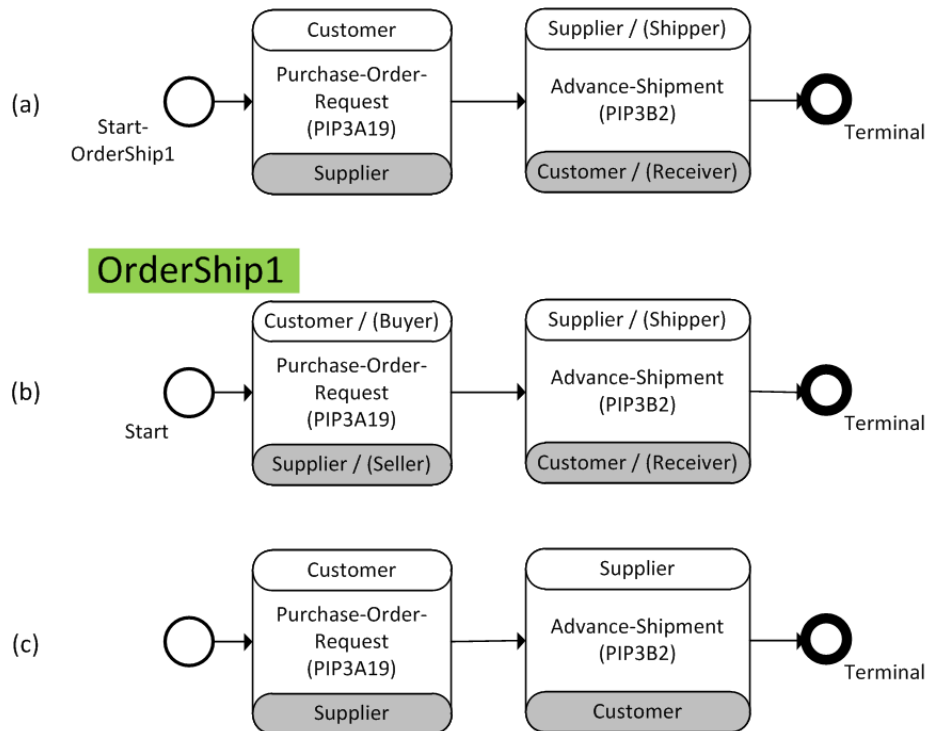


Figure 13: Representing Choreographies

The BPMN standard does not offer a dedicated construct for visualizing choreographies. In particular, there is no “framing” mechanism to delineate the constructs of choreographies from other choreographies’ constructs and no naming mechanism. This task is to be implemented by tools. In consequence, choreographies essentially are identified by identifying start and end states that are interconnected via series of control flow states, PIPs, component choreographies and transitions. Figure 13 (a), (b) and (c) show different ways of representing a basic choreography that consists of two subsequent PIP choreography tasks and does not contain any branching logic. Note that all three options basically express the same business information. In particular, it does not make a difference whether or not the choreography roles ‘Customer’ and ‘Supplier’ are explicitly mapped to PIP functional roles or not (confer Construct Advice 1). However, the benefit of separating choreography roles from PIP functional roles becomes evident. By means of choreography roles it is clear that the very same ‘Supplier’ role takes the ‘Seller’ role in PIP 3A19 and the ‘Shipper’ role of PIP 3B2.

As the BPMN standard does not constrain the labeling of choreographies this specification proposes to either assign the name of the choreography to its start state (if the start state is unique) or to a text box placed in the “proximity” of the start state. If no visualization of the choreography’s name is needed not visualizing the name is acceptable as well.

Construct Advice 6: Representing Decisions

Decisions are needed in choreographies to realize alternative control flow paths based on the result of PIPs or choreographies. This specification visualizes such decisions either implicitly by means of the set of outgoing transitions of a PIP or component choreography or by means of BPMN *Exclusive Gateways* (OMG, January 2011), section 11.6.1) that are represented as a diamond. Figure 14 (a) gives a sample for the former and Figure 14 (b) gives a sample for the latter way of visualization. The condition expressions defined in “Construct Advice 4: Representing Transitions” are admissible for transitions. Modelers should use only one single incoming transition into BPMN exclusive gateways because condition expressions are defined relative to the preceding PIP or component choreography. For example, XPath expressions are to be evaluated against the business document exchanged during the latest PIP instance.

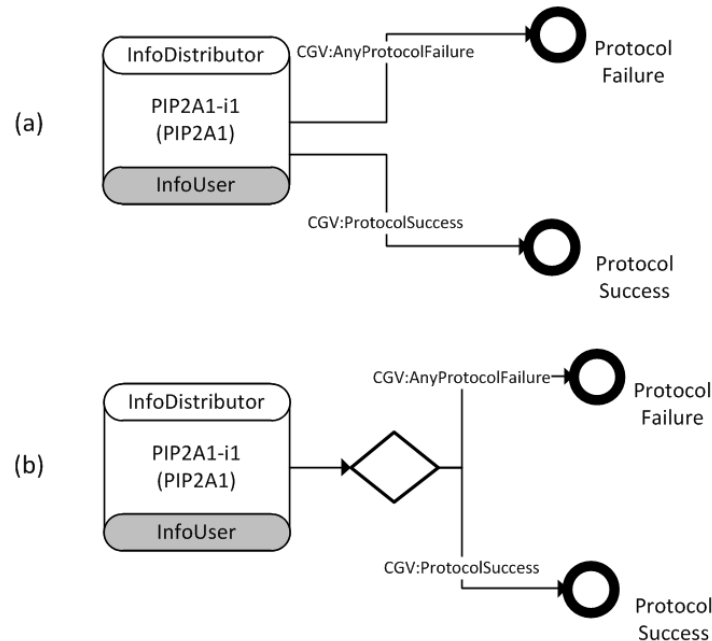


Figure 14: Representing Decisions

It is good practice to define the condition expressions of the branches of a particular decision such that they are complete and disjoint. Completeness means that there is at least one condition expression that evaluates to true for any outcome of the preceding PIP or component choreography. Disjointness means that there is no outcome of the preceding PIP or component choreography for which more than one condition expression of the decision evaluates to true.

There is no evaluation order for the separate condition expressions of the branches of a decision. The only exception is that CGV-based condition expressions are evaluated before any other condition expressions.

Exclusive gateways can but do not have to carry names. If names are assigned these

should not contain spaces or special characters.

Construct Advice 7: Representing Event-Based Choices

An event-based choice is used within choreographies to realize alternative control flow paths based on events rather than based on the result of preceding activities. The initiation of PIPs, component choreographies or timeouts can be such events. BPMN *Event-Based Gateways* ((OMG, January 2011), section 11.6.2) that take the shape of a diamond with a double circle and a pentagon inside are used to represent event-based choices. Admissible events are timeouts, represented by BPMN timer symbols, and PIP or component choreography initiations, represented by the corresponding visualizations. The semantics is such that the first event fired out of the events connected to the event-based choice triggers leaving the event-based choice. The implementation of this semantics is left to the integration partners which may lead to restrictions about the types of events that can be combined after an event-based choice, e.g., the follow-on PIPs could be constrained to share a common requesting role. However, this specification does not impose such restrictions upon adopters. Figure 15 demonstrates that it is perfectly acceptable to associate two PIP choreography tasks with distinct initiators with the same event-based choice. Note that an event-based choice can be the target of one or more transitions as well as the source of one or more transitions.

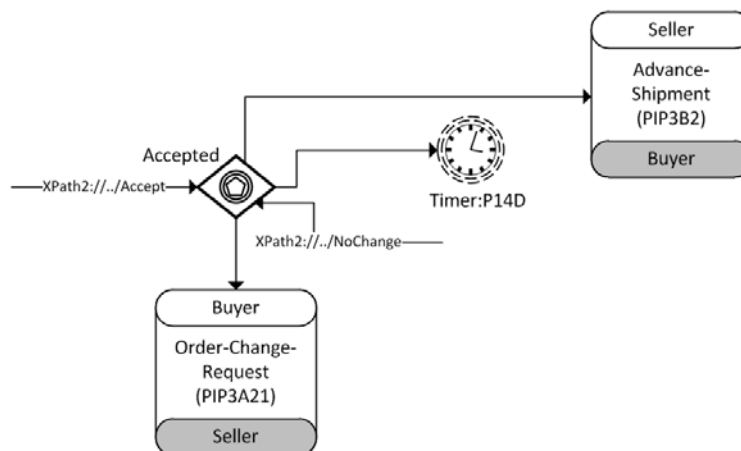


Figure 15: Event-Based Choice Sample

Event-based choices can carry names. If names are assigned these MUST NOT contain special characters and MUST be unique relative to the enclosing choreography. Modelers may want to use this feature to document the progress of choreographies by assigning meaningful names. For example, the event-based choice of Figure 15 carries the name 'Accepted' to highlight that a prior purchase order has been accepted. In this 'Accepted' state, either PIP 3B2 or 3A21 can be triggered or a timeout will be fired after 14 days.

Construct Advice 8: Representing Parallel Structures

Parallel structures may be used to define the possibility that two or more PIPs or component choreographies can be performed at the same time. BPMN *Parallel Gateways* ((OMG, January 2011), section 11.6.4) visualized as diamonds with black, solid crosses inside are used to define parallelism. The semantics of this type of pseudo states depends on the number of transitions. It joins incoming flows if there is more than one incoming transition and it forks control flow if there is more than one outgoing transitions or both. If a parallel pseudo state joins multiple incoming

flows then no outgoing transition is fired until all preceding parallel activities have completed. Similarly, all outgoing transitions are fired and not only a subset thereof. In that sense, parallel pseudo states have “AND” semantics. It is good modeling practice to combine forking and joining gateways in pairs and not to define control dependencies between the individual branches. However, this specification does not define such restrictions in general, in particular not for cartography choreographies. Moreover, it is acceptable to define a parallel pseudo state with only one single incoming and one single outgoing transition.

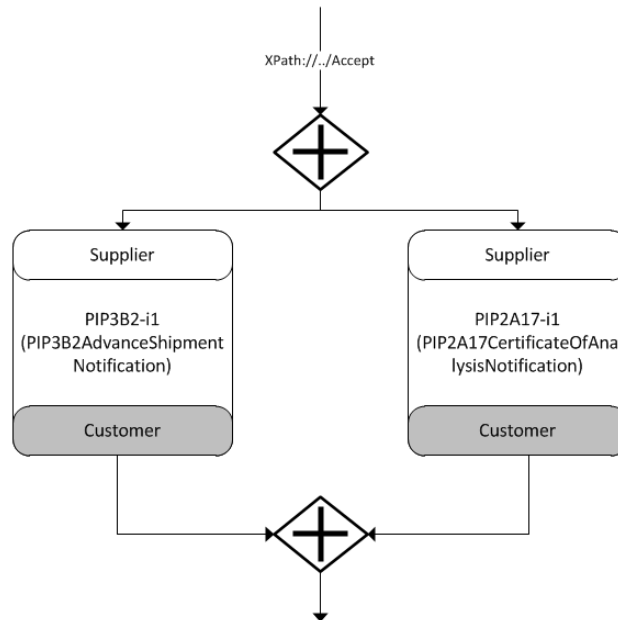


Figure 16: Representing Parallel Structures

Parallel gateways can but do not have to carry names. If names are assigned these these MUST NOT contain special characters and MUST be unique relative to the enclosing choreography.

Construct Advice 9: Representing Component Choreographies

Component choreographies are used for encapsulating choreography logic and reusing it within larger parent choreographies. Using component choreographies, hierarchical decomposition of complex choreography models can be facilitated. BPMN offers two different options for visualizing component choreographies, so-called *sub-choreographies* ((OMG, January 2011), section 11.4.2) and so-called *call choreographies* ((OMG, January 2011), section 11.4.3). While BPMN defines *expanded* and *collapsed* versions of both types only *expanded sub-choreographies* and *collapsed call choreographies* are used in RosettaNet MCC for avoiding unnecessary complexity. Figure 17 and Figure 18 both show the visualization of the component choreography named ‘subchor’ as an expanded sub-choreography. In these visualizations, the full definition of ‘subchor’ is given. Figure 19 shows the visualization of a component choreography as a call choreography task. The ‘Invoicing’ choreography of Figure 18 is incorporated into the choreography of Figure 19 by referring to its name ‘Invoicing’ within parentheses. While the full definition of the component choreography is available it is not displayed within the parent choreography. The call choreography task itself carries a separate identifier ‘c4’ for being able to distinguish separate ‘Invoicing’ instances within the parent choreography (which are not displayed).

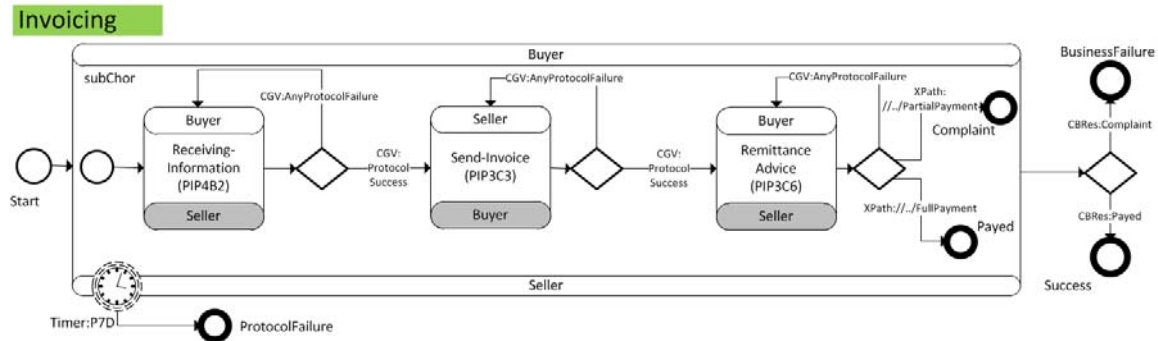


Figure 17: Sample Expanded Sub-Choreography with Implicit Role Mapping

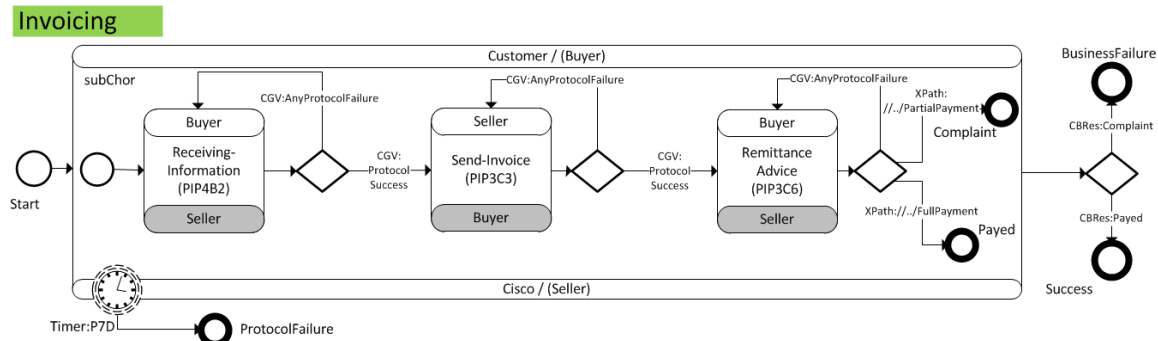


Figure 18: Sample Expanded Sub-Choreography with Explicit Role Mapping

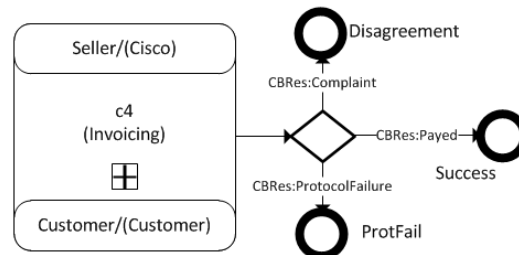


Figure 19: Sample Call Choreography with Explicit Role Mapping

For component choreographies, the full set of constructs described in this section is available. However, some restrictions may apply depending on the choreography style (see section 5.3). The semantics of component choreographies is such that when a transition to a sub-choreography or call choreography state is fired then the start state of the component choreography becomes activated and the parent choreography's state can only be left upon termination of the component choreography (except for timers, see Construct Advice 10).

Additionally, the role definitions of the parent choreography must be mapped to the roles of the component choreography. Note that there always are distinct roles for the parent choreography and its component choreographies even if the same names are used. Therefore, in Figure 17, there are 'Seller' and 'Buyer' roles at the parent 'Invoicing' choreography level and at the component 'subchor' level. As no explicit role mapping is provided, the parent level roles are matched with the component level roles by means of string equality. The choreography of Figure 18 semantically is identical to the one of Figure 17 except for different parent level role names. Due to this difference the parent level roles are mapped explicitly to the component level roles using the notation introduced in Construct Advice 1. Explicit and implicit role

mapping are available for call choreographies as well, although only the explicit version is exemplified in Figure 19. Note that, from a technical perspective, that the 'Cisco' role indeed is just a role definition that can be mapped to.

The results of component choreographies can be used to specify the control flow of parent choreographies by leveraging the CBRes expression language introduced in Construct Advice 4. Basic CBRes expressions refer to the names of the end states of component choreographies and become true when the corresponding end state is reached. More complex expressions can be created by using the standard Boolean operators to combine basic CBRes expressions. The use of CBRes expressions is exemplified in each of the Figure 17, Figure 18 and Figure 19. At this point, note that Figure 19 indeed has four choreography levels, namely the outermost choreography level depicted in Figure 19, the 'Invoicing' and 'subchor' choreography levels of Figure 18, and the choreography level for performing PIPs according to RosettaNet MCC phase 1.

For component choreographies with more than two roles, additional participant bands are added to the top or the bottom of the sub-choreography or call choreography shapes. In this case, it may be necessary to map one parent role to more than one component role. This is explicitly notated as follows:

ParentRole / (ComponentRole1, ComponentRole2, ...)

There are no constraints with respect to the sub-choreography or call choreography id names except for that special characters MUST NOT be used. Additionally, those id names MUST be unique relative to the enclosing choreography. In case of global call choreographies, this means that its id name MUST be distinct from all other top-level choreography id names.

Construct Advice 10: Representing Timeouts

Timers come in two flavors, component choreography timers and event-based choice timers. These two types of timers reflect ebXML BPSS functionality and BPMN 2.0 interrupting as well as non-interrupting timer shapes ((OMG, January 2011), table 10.90) are used to represent the according events.

Component choreography timers are added to the boundary of sub-choreography or call choreography shapes and specify a time interval or a date and time that complies to ISO 8601. A component choreography timer has a follow-on node that is reached when a timer runs out. If the timer is interrupting (solid outer line of the shape) then the currently active task of the component choreography (if any) is interrupted and the timer's follow-on node is immediately reached. If the timer is non-interrupting (dotted outer line of the shape) then the completion of the currently active task of the component choreography is waited for before the timer's follow-on state is reached. Figure 17 above shows a non-interrupting component choreography timer that is attached to the 'subchor' sub-choreography shape and specifies a timeout after 7 days after the start of the sub-choreography ('P7D' is the ISO 8601 definition for Period-7-Days). Similarly, Figure 20 shows an interrupting component choreography timer that is attached to call choreography task 'c4' and specifies a timeout after 12 hours ('PT12H' stands for Period-Time-12-Hours). Note that interrupting timers can be associated with sub-choreographies and non-interrupting timers with call choreographies as well.

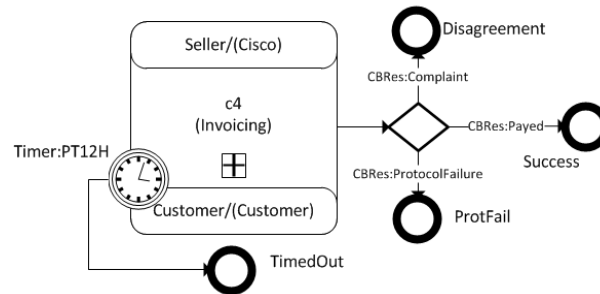


Figure 20: Sample Scenario for an Interrupting Choreography Timer

Event-based choice timers (ebc timer) may be used as an alternative to choreography tasks after event-based choice states. ebc timers specify a time interval or a date and time and if the timer runs out before any of the alternative outgoing transitions of the respective event-based choice is fired then the follow-on state of the timer is reached. Figure 21 shows an ebc timer that is triggered if the 'Buyer' role does not initiate PIP 3A21 within 3 days.

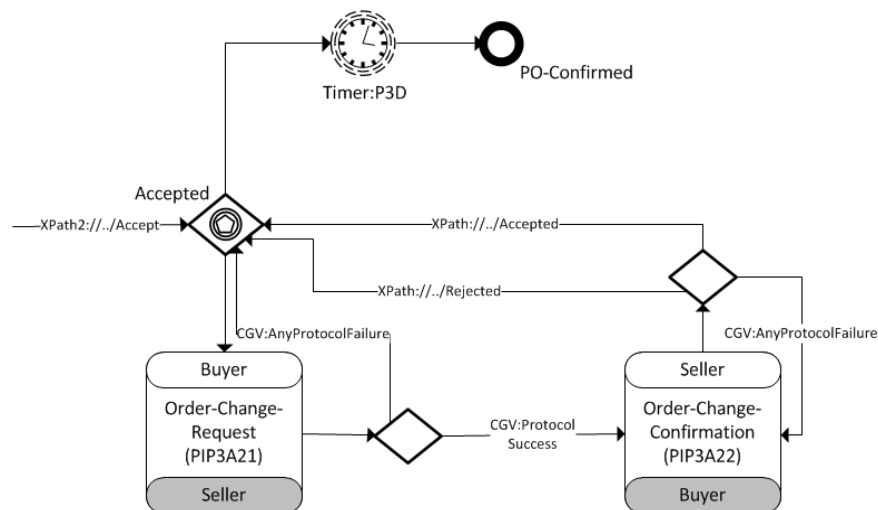


Figure 21: Event-Based Choice Timer Scenario

In this scenario, the event-based choice 'Accepted' can be entered multiple times in case the Buyer initiates one or more PIP 3A21 executions. If such iterative behavior is part of a choreography definition it has to be decided whether or not an ebc timer is to be reset in case it is not reached for the first time. The default semantics is that an ebc timer is not reset. If reset is needed then a reset flag has to be added to the transition that enters the corresponding event-based choice. If an ebc timer runs out while some successor state of an alternative outgoing transition of the respective event-based choice is active then the event is not processed until the event-based choice is reached again. That means that ebc timers always are non-interrupting.

Note that ebc timers MUST not be confused with RosettaNet PIP TimeToPerform parameters. While ebc timers are controlled at the level of the including choreography, PIP TimeToPerform parameters are controlled at the PIP level.

This specification requires that labels of timers start with the string "Timer:" and then are followed by an ISO 8601 compliant definition of an absolute or relative date and time.

5.3 Choreography Grammar

This section presents the rules for composing the choreography constructs described in the last section. Note that these rules are characteristic for the modeling style that you choose. Consequently, the section on 'cartography choreographies' (sect. 5.3.1) only contains very coarse rules as modelers are supposed to use choreographies as a loose gathering of implementation requirements and to leverage any BPMN construct (also those not described in the last section) as needed. Conversely, sections 5.3.2 and 5.3.3 present much more tight rules for ensuring executability and analysis features of the 'executable choreography' and 'sequential multi-party choreography' style. For more detailed discussion of the rationale for offering several choreography modeling styles, see section 5.1.

5.3.1 Cartography Choreographies

The 'cartography choreography' modeling style motivated in section 5.1.1 only contains very few choreography construct composition rules and modelers are free to use any modeling construct of the BPMN choreography standard (OMG, January 2011), sect. 11). This is to make this modeling style accessible and to foster discussions between process modelers of interacting B2Bi partners as well as process modelers and software engineers.

For creating valid cartography choreographies, you must make sure that the constituent parts of a choreography model are easily identifiable. As the BPMN choreography standard does not comprise 'framing rules' (this is left to tool implementations of the standard), you have to respect the following rules:

Cartography Rule 1: Start States

Cartography Choreographies **MUST** have one or more start states as defined in (OMG, January 2011), section 11.5.1. However, there **SHOULD** be exactly one start state as defined in *Construct Advice 2: Representing Start States*.

Cartography Rule 2: End States

Cartography Choreographies **MUST** have one or more end states as defined in (OMG, January 2011), section 11.5.3.

Cartography Rule 3: Connectedness

Any choreography element, i.e., any choreography task, choreography or control-flow construct **MUST** be connected (without considering direction of transitions) to any start state and to any end state of the choreography.

Cartography Rule 4: Construct Semantics

By default, the informal semantics presented in the last section and in sections 5.3.2 as well as 5.3.3 is assumed for any construct of cartography choreographies that are covered by the construct advices of the last section. If additional constructs are used or if deviating semantics are to be applied, the interacting B2Bi partners are assumed to agree on common semantics.

5.3.2 Executable Choreographies

This section defines the composition rules for executable choreographies as motivated in section 5.1.2. Executability of choreographies refers to the fact that complete implementations of BSIs can be derived from a particular executable choreography definition such that all message exchange sequences that conform to it are accepted by the generated BSIs. Therefore, at any point in time the admissible transitions and events must be clear. To ensure this executability property, you **MUST** comply with the following rules:

ExecChor Rule 1: Bilaterality

'Executable choreography' models **MUST** define exactly two top-level choreography roles.

ExecChor Rule 2: Eligible Constructs

'Executable choreography' models are restricted to the constructs described in the construct advices of section 5.2.

ExecChor Rule 3: Transition Coordination

The processing of transitions is crucial for the execution semantics of process models. For 'executable choreography' models, an outgoing transition of any node represents one option to continue the choreography. Two preconditions must be met for a transition to really fire.

First, the transition must be enabled, i.e., the source of the transition **MUST** be member of the set of the choreography's currently active states and if the transition carries a guard then this guard **MUST** evaluate to true.

Second, depending on the target of the transition, firing **MUST** be coordinated between the integration partners by means of requesting and confirming firing. Coordination ensures that only one out of multiple enabled transitions is fired. Further, by means of coordination on starting PIPs as well as component choreographies, both partners are aware of the fact that activities have started. Consequently, even protocol failures are valid activity outcomes to take routing decisions upon. Details on how such coordination **MAY** be implemented is described in (Schönberger, et al., November 2010). The following list describes which transitions require coordination depending on the target state of the transitions:

- PIP:
Firing the transition **MUST** be coordinated.
- Component Choreography:
Firing the transition **MUST** be coordinated.
- Fork state node:
If the transition under consideration is the only enabled transition then firing **MAY** be coordinated. Otherwise, firing **MUST** be coordinated.
- Join state node:
Firing the transition is performed immediately and **MUST NOT** be coordinated.
- Event-Based Choice:
If the transition under consideration is the only enabled transition then firing **MAY** be coordinated. Otherwise, firing **MUST** be coordinated.
- Decision state:
Firing the transition is performed immediately and **MUST NOT** be coordinated.
- End state:
Firing the transition is performed immediately and **MUST NOT** be coordinated.

ExecChor Rule 4: Start States of Choreographies

'Executable choreography' models MUST define exactly one start state as defined in *Construct Advice 2: Representing Start States*. Figure 22 enumerates the different options, (a) to (e), for using start states to define the entry point into executable choreographies. Although not explicitly specified, the same options are available when defining a sub-choreography, i.e., in Figure 22 (e) the three dots can be replaced with any of the listed options. Note that the start state of an executable choreography has exactly one outgoing transition that does not carry any guard and therefore immediately is enabled. Actual firing depends on the type of target state as described in *ExecChor Rule 3: Transition Coordination*.

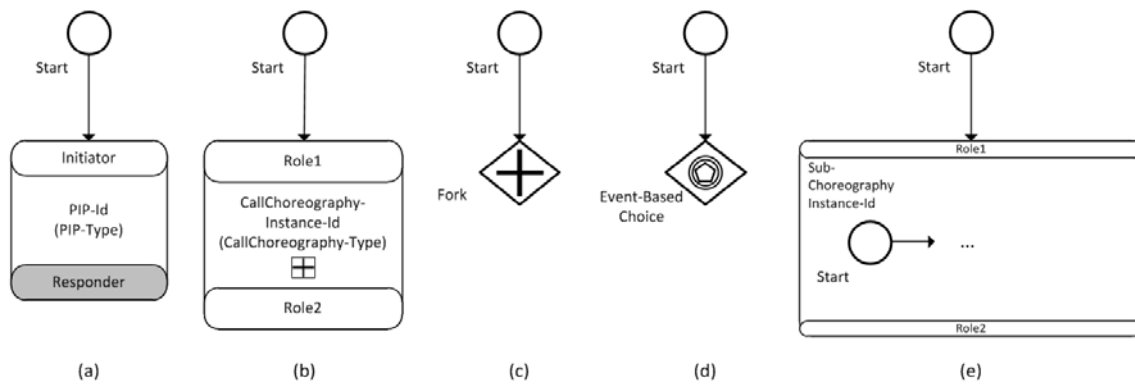


Figure 22: Options for Starting Executable Choreographies

ExecChor Rule 5: End States of Choreographies

'Executable choreography' models MUST define one or more end states as defined in *Construct Advice 3: Representing End States*. Upon reaching an end state, executable choreographies terminate. In case the choreography under consideration is a sub-choreography of some superordinate choreography then the name of the end state immediately is propagated to the superordinate choreography and represents the result of the choreography execution. The next state to be entered then is computed according to the semantics presented in *ExecChor Rule 7: Component Choreography Evaluation*.

If the choreography under consideration is a top-level choreography then no further activities are admissible. Note that due to the set of rules presented here, executable choreographies do not allow for multiple threads of the same choreography instance that terminate in separate top-level end states.

ExecChor Rule 6: PIP Evaluation

PIPs are performed according to the protocol definition put down in the MCC phase 1 specification and take time. A PIP can only be left upon termination of the MCC phase 1 protocol which ensures that both integration partners have agreed upon the result of the PIP execution, may it be a protocol failure or the contents of the exchanged business document. This result MAY be used for control flow routing purposes, i.e., to determine the follow-on state of the PIP.

However, specifying unconditional progress without evaluating the outcome of a PIP as depicted in Figure 23 (c) is acceptable as well. Note that although that particular transition is immediately enabled upon completion of the PIP, actual firing depends on the target state as described in *ExecChor Rule 3: Transition Coordination*.

If the PIP result is used for routing purposes then a set of 'decision transitions' is used to represent the branches of a decision that determines the follow-on state. Each of these decision transitions **MUST** carry a guard that complies to Construct Advice 4. Decision transitions are either added directly to the PIP choreography task itself or to a dedicated *decision* state node. It is acceptable to combine both possibilities for adding decision transitions (see Figure 23 (a) and (b)). However, this specification **RECOMMENDS** to either exclusively add decision transitions directly to the PIP choreography task or exclusively to a decision node. Note that a transition from a PIP to a decision node does not carry any guard and does not count to the set of decision transitions. Whether or not and when a decision transition is fired depends on the value of its guard and transition coordination as described in ExecChor Rule 3.

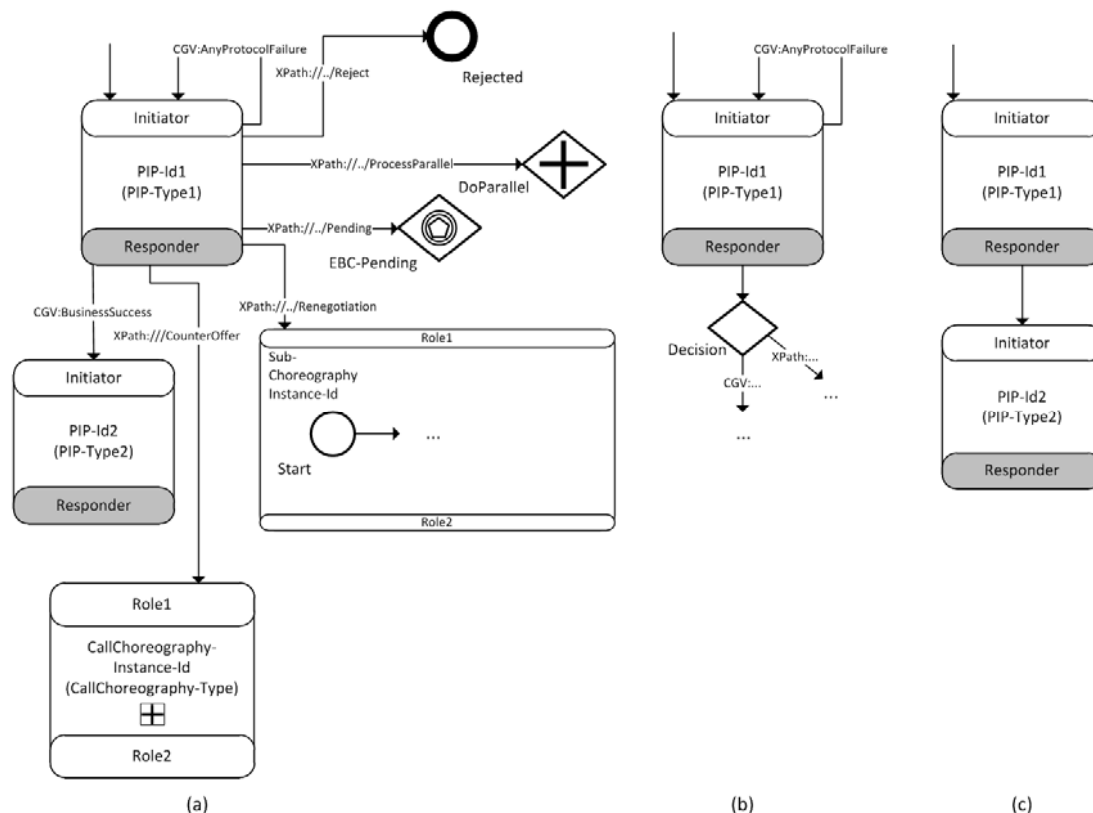


Figure 23: Options for Continuing PIPs

Note that the RosettaNet PIP performance control *TimeToPerform* is controlled at the PIP level and therefore not explicitly visualized using some timer construct at the choreography level. Therefore, timer constructs **MUST NOT** be added to PIP choreography tasks.

ExecChor Rule 7: Component Choreography Evaluation

Component choreographies either are call choreographies or sub-choreographies according to Construct Advice 9. The necessary steps for determining the follow-on states of component choreographies (evaluation) are identical for both types. Also, these steps are very similar to the rules for PIPs (cf. ExecChor Rule 6). Therefore, the considerations for unconditional progress and the optional use of dedicated decision nodes hold true correspondingly.

However, the guards of decision transitions are restricted to Boolean expressions

built from the names of the end states of the component choreography (cf. Construct Advice 4: Representing Transitions). Figure 24 demonstrates how such guards can be used. Evaluation of the guards takes place upon component choreography termination. Again, the guard of a transition MUST evaluate to true for a particular transition to be enabled. Actual firing depends on the target of the transition and follows *ExecChor Rule 3: Transition Coordination*.

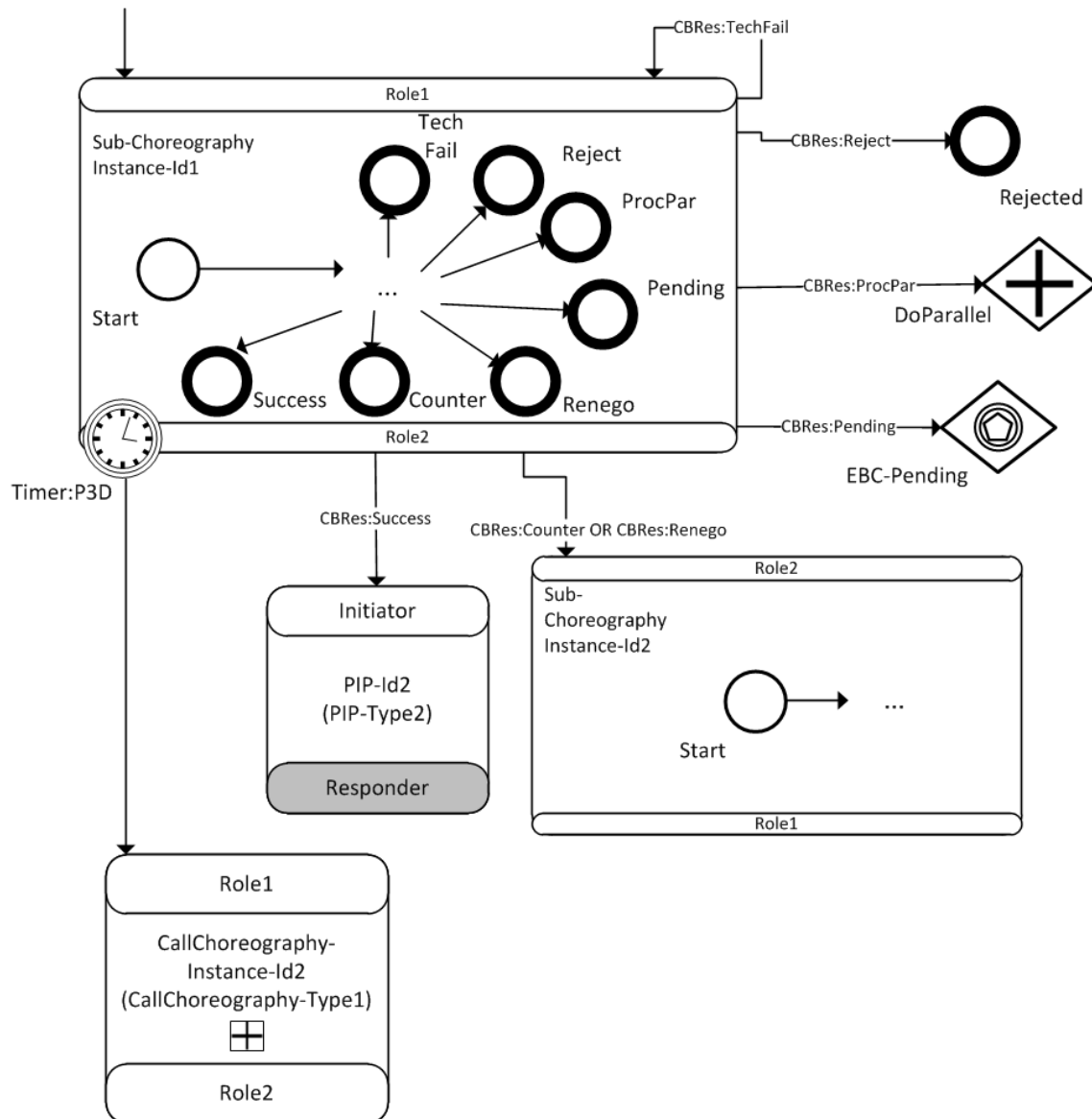


Figure 24: Options for Continuing Component Choreographies

Component choreographies allow for the specification of interrupting and non-interrupting timers that are added to the boundary of the respective construct and that MUST comply with Construct Advice 10. Additionally, timers have exactly one outgoing transition without a guard. A timer becomes enabled when a component choreography gets started and disabled when it terminates. If the timer runs out in between, the processing depends on whether the timer is interrupting or non-interrupting. In the former case, completion of the component choreography's

currently active state is awaited for whereas the state is interrupted in the latter. It is RECOMMENDED that integration partners track the progress of the timed out component choreography in order to negotiate the business effect of timeouts that may go beyond determining the next state of choreographies.

The following list discusses processing of timeout events for both cases for the respective types of states that may be the currently active state of a component choreography:

- **Start state:**
The outgoing transition of the start state is immediately deactivated independent of whether the timer is interrupting or non-interrupting. This also holds true in case the integration partners already have begun coordinating *upon the start* of the start state's successor.
- **PIP:**
If the timer is non-interrupting, completion of the PIP protocol is awaited for and processing of the component choreography is stopped thereafter. Then, the outgoing transition of the timer is activated.
If the timer is interrupting, a cancel signal is sent to the PIP execution protocol defined in the MCC phase 1 specification. If the cancel signal is accepted, the result of the PIP execution is an *AnyProtocolFailure*. If the cancel signal is not accepted, which may be the case as the PIP execution protocol may be in its finalization phase, then completion of the PIP execution protocol's finalization phase is awaited for. Afterwards the outgoing transition of the timer is activated.
- **Component choreography:**
This rule is applied recursively.
- **Parallel structure:**
This rule is applied to each of the branches of the parallel structure.
- **Event-Based Choice:**
All outgoing transitions of the event-based choice state are immediately deactivated independent of whether the timer is interrupting or non-interrupting.
- **Decision state:**
Not applicable because the processing of decisions is assumed to take zero time.
- **End state:**
Not applicable because the component choreography timers are deactivated upon reaching one of its end states.

If a timer runs out, the follow-on state of the choreography is determined by its outgoing transition. The point in time when the transition is fired complies to *ExecChor Rule 3: Transition Coordination*.

ExecChor Rule 8: Event-Based Choice States

Event-based choice states are used to select one out of multiple possible events. For 'executable choreography' models, all outgoing transitions do not carry guards and therefore immediately are enabled. Except for timers, the integration partners are the source of events that trigger firing one of the transitions. As both integration partners may detect the need to take different transitions at the same time, they coordinate on which transition actually is taken. Therefore, specifying multiple PIPs with different PIP initiator roles as successors of an event-based choice is perfectly acceptable. One of the two integration partners takes the task of controlling a

possible timer successor. If the timer runs out, this partner is responsible for coordinating with its partner that the event-based choice has been left by means of a timeout.

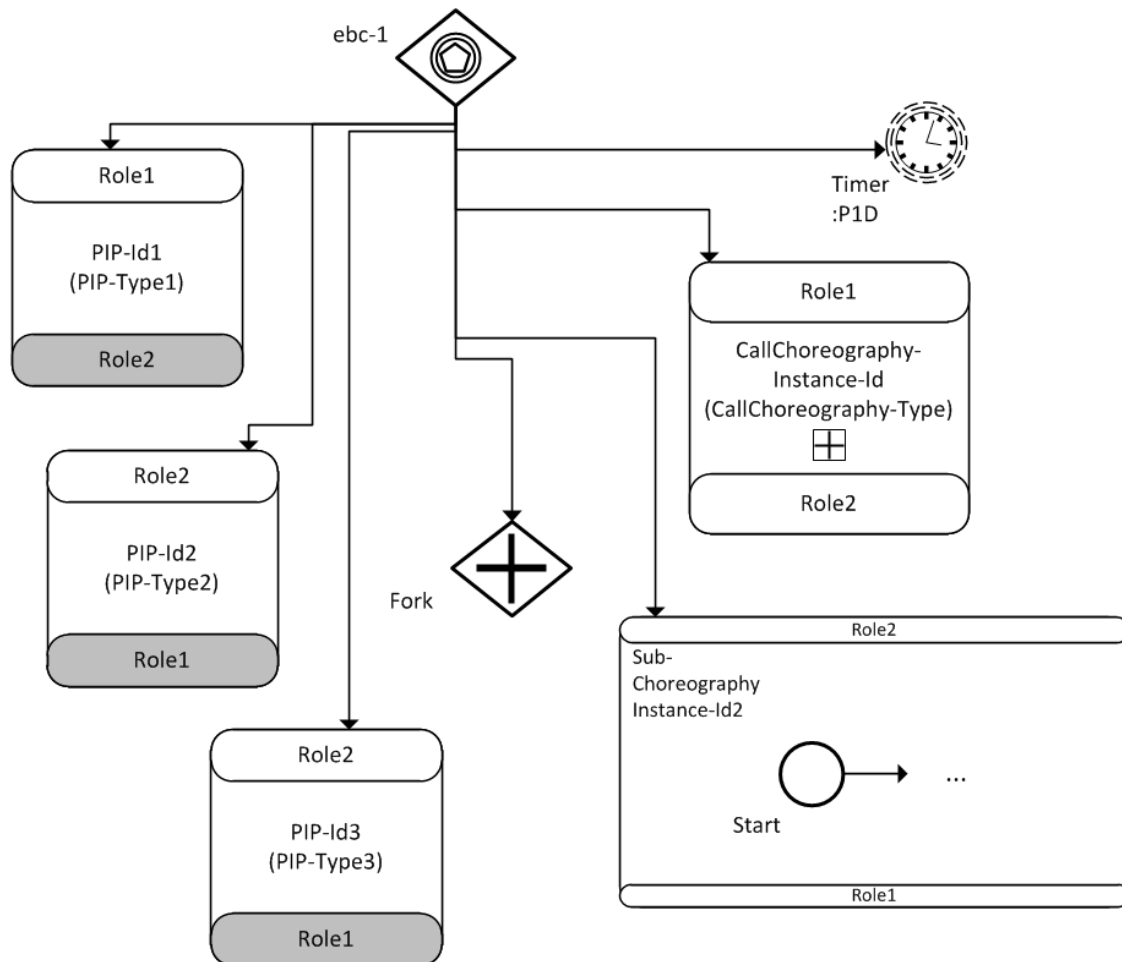


Figure 25: Options for Continuing Event-Based Choices

ExecChor Rule 9: Parallel Composition

Parallel Structures in 'executable choreography' models always are defined by means of a pair of fork and join states. One or more branches are defined between a fork state and a join state that are processed in parallel. Each branch is interpreted as a component choreography. However, this does not necessarily require the use of call choreography tasks or sub-choreographies as shown in the middle branch and the right-hand branch of the parallel structure of Figure 26. If alternative constructs are used to start a branch of a parallel structure then the respective construct is interpreted as the first state after the start state of a virtual sub-choreography. So, for Figure 26, the choreography task with instance id 'PIP-Id1' is considered to be the initial state after a virtual component choreography's start state. The grammar rules for creating choreographies (this section) then are applicable for creating the virtual sub-choreography. The only restriction of the rules is that exactly one transition to an end state of the virtual sub-choreography is replaced by a transition to the join state of the parallel structure. The virtual sub-choreography is assumed to

be performed as if a dedicated sub-choreography construct would be used. This means that multiple end states may be specified and that reaching any end state of the virtual sub-choreography results in reaching the join state of the parallel structure afterwards. For clarification, consider that Figure 26 and Figure 27 semantically are equivalent.

Although the use of virtual sub-choreographies for defining branches of a parallel structure may be convenient for modeling, this specification explicitly RECOMMENDS the use of dedicated call choreography tasks or explicit sub-choreographies to avoid confusion about the interpretation of virtual sub-choreographies.

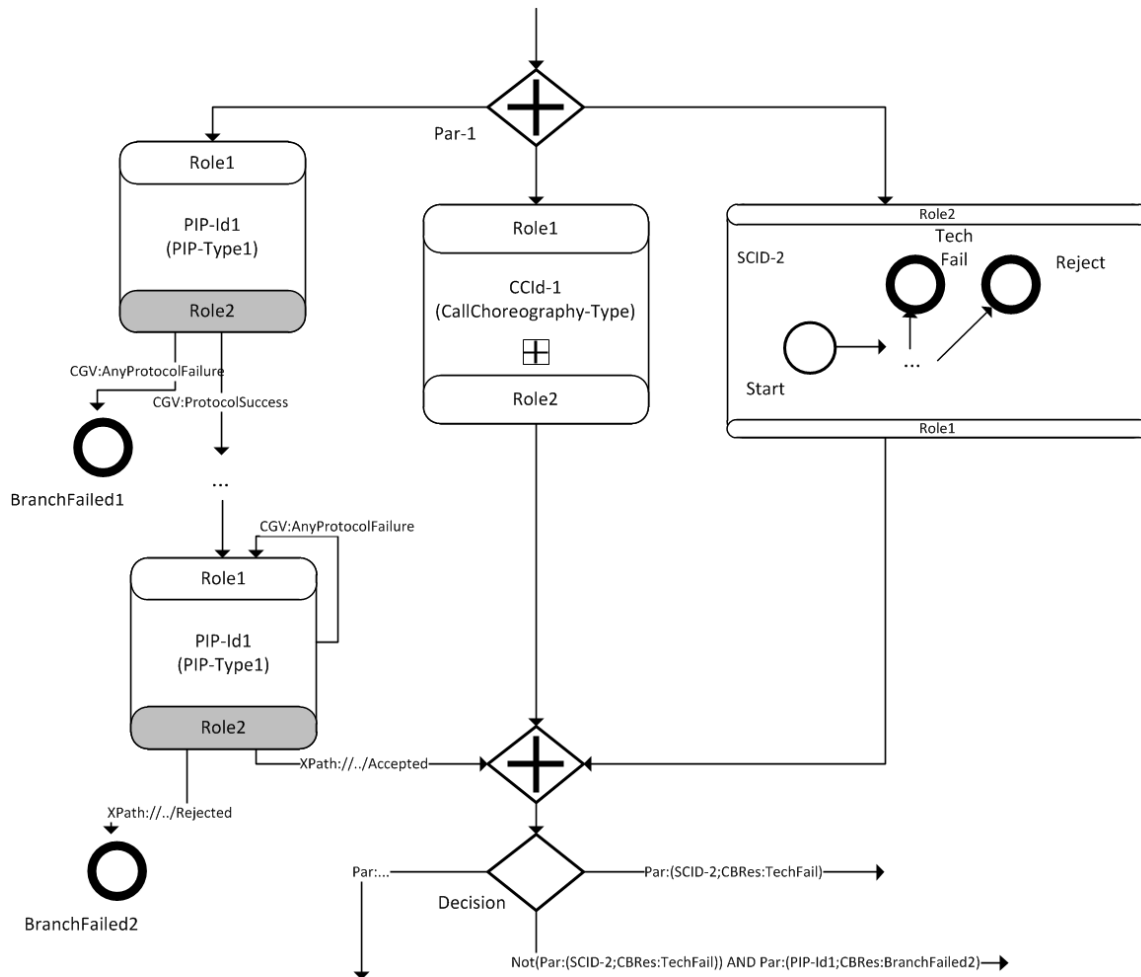


Figure 26: Parallel Structure with Virtual Sub-Choreography

Note that the incoming and outgoing transitions to the component choreographies that make up the branches of a parallel structure do not carry any guards and immediately are fired. Also, transitions between the branches of a parallel structure MUST NOT be defined.

Finally, there may be the need for evaluating the results of a parallel structure's activities to determine the control flow after its join state. Therefore, the "Par" expression language is defined to be available for the outgoing transitions of a join state. Basically, "Par" expressions are built from the result expressions for the component choreographies that make up the branches of a parallel structure. So, basic Par expressions are 2-tuples, denoted as "Par:(Branch-Id;CBRes-Exp)", where

Branch-Id is the id of such a component choreography and CRes-Exp is a valid CRes expression for the component choreography as described in Construct Advice 4. In case a virtual sub-choreography is to be evaluated, the Branch-Id refers to the id of the first construct of the sub-choreography. For example, the Par expression “Par: (PIP-Id1; CRes: BranchFailed2)” is valid for the parallel structure depicted in Figure 26. Also, the standard Boolean operators can be used to create complex Par expressions from basic Par expressions.

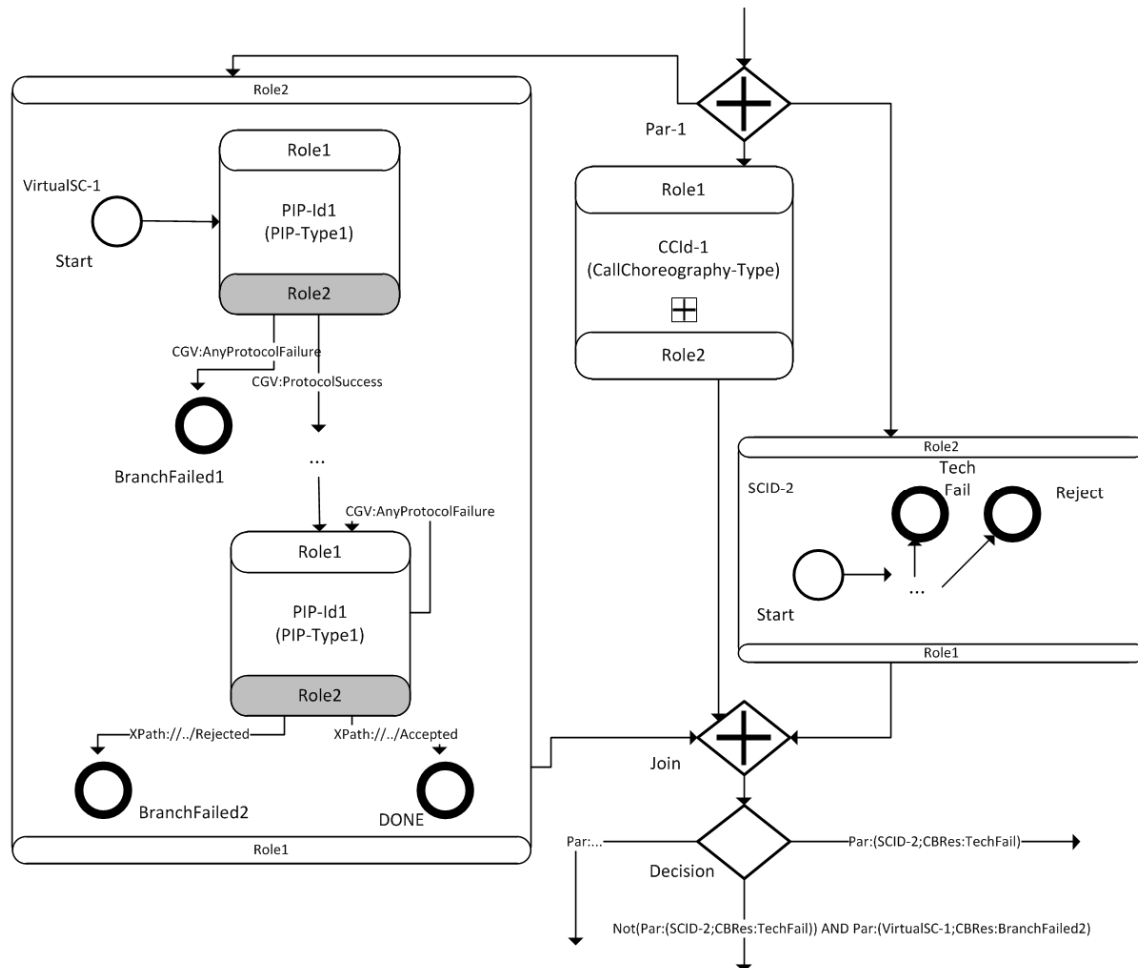


Figure 27: Parallel Structure without Virtual Sub-Choreography

ExecChor Rule 10: Connectedness

Without considering the direction of transitions, any state of an ‘executable choreography’ model MUST be connected to the start state and all the end states of the same model.

ExecChor Rule 11: Guard Constraints

For the set of decision transitions as used in ExecChor Rule 6 and ExecChor Rule 7 the following holds true:

- **Completeness:**
The disjunction of the guards of all decision transitions must evaluate to true for any result of the evaluated activity.

- Disjointness:
No two guards of decision transitions may both evaluate to true for the same result of the evaluated activity.
- PIP ProtocolSuccess:
If the evaluated activity is a PIP then any non-CGV guard implicitly is AND connected with the expression "CGV:ProtocolSuccess".

ExecChor Rule 12: Producibility

'Executable choreography' models must be producible according to the rules of this section.

5.3.3 Sequential Multi-Party Choreographies

This section defines the composition rules for sequential multi-party (SeqMP) choreographies as motivated in section 5.1.3. Remind that the purpose of SeqMP choreographies not only is defining multi-party choreographies (for an arbitrary large, but fixed set of roles) but in particular the analysis of synchronization deficits. The following rules that **MUST** be followed for defining valid SeqMP choreographies reflect this focus on analysis features:

SeqMP Rule 1: Eligible Constructs

SeqMP choreography models are restricted to start states, end states, transitions, decisions and component choreographies as described in section 5.2. The following rules abstract from whether component choreographies are represented as call choreographies or sub-choreographies. However, the component choreographies of SeqMP models **MUST** be valid 'executable choreography' models (cf. section 5.3.2). Integration partners only **MAY** agree to use different types of component choreographies if they make sure that the end states of the respective component choreography is commonly reached by all component choreography roles. In consequence, standard SeqMP models **MUST NOT** be used as component choreographies of other SeqMP choreographies.

SeqMP Rule 2: Subsequent Role Participation

In SeqMP choreography models, any two subsequent top-level component choreographies **MUST** share at least one top-level choreography role.

SeqMP Rule 3: Transition Coordination

Two preconditions must be met for a transition to really fire.

First, the transition must be enabled, i.e., the source of the transition must be the SeqMP choreography's currently active state and if a guard is defined on the transition then this guard must evaluate to true.

Second, depending on the target of the transition, firing **MUST** be coordinated between the integration partners by means of requesting and confirming firing. Concrete coordination **MAY** be designed according to the definitions in (Schönberger, et al., November 2010). The following list describes which transitions require coordination depending on the target state of the transitions:

- Component Choreography:
Firing the transition **MUST** be coordinated.
- Decision State:
Firing the transition is performed immediately and **MUST NOT** be coordinated.
- End State:
Firing the transition is performed immediately and **MUST NOT** be coordinated.

SeqMP Rule 4: Start States

SeqMP choreography models **MUST** define exactly one start state as defined in Construct Advice 2: Representing Start States. This start state has exactly one outgoing transition the target of which **MUST** be a component choreography.

SeqMP Rule 5: End States

SeqMP choreography models **MUST** define one or more end states as defined in

Construct Advice 3: Representing End States. Upon reaching an end state, executable choreographies terminate. As SeqMP choreographies are not composable (see SeqMP Rule 1), result propagation to higher-order choreographies is not applicable.

SeqMP Rule 6: Component Choreography Evaluation

The rules for determining the follow-on state of component choreographies in SeqMP models are very similar to the rules for determining the follow-on state of component choreographies in 'executable choreography' models (cf. ExecChor Rule 6 and ExecChor Rule 7). Therefore, the considerations for unconditional progress, for the use of the CBRes expression language and the optional use of dedicated decision nodes hold true correspondingly. However, decision transitions may only point to either component choreographies or end states.

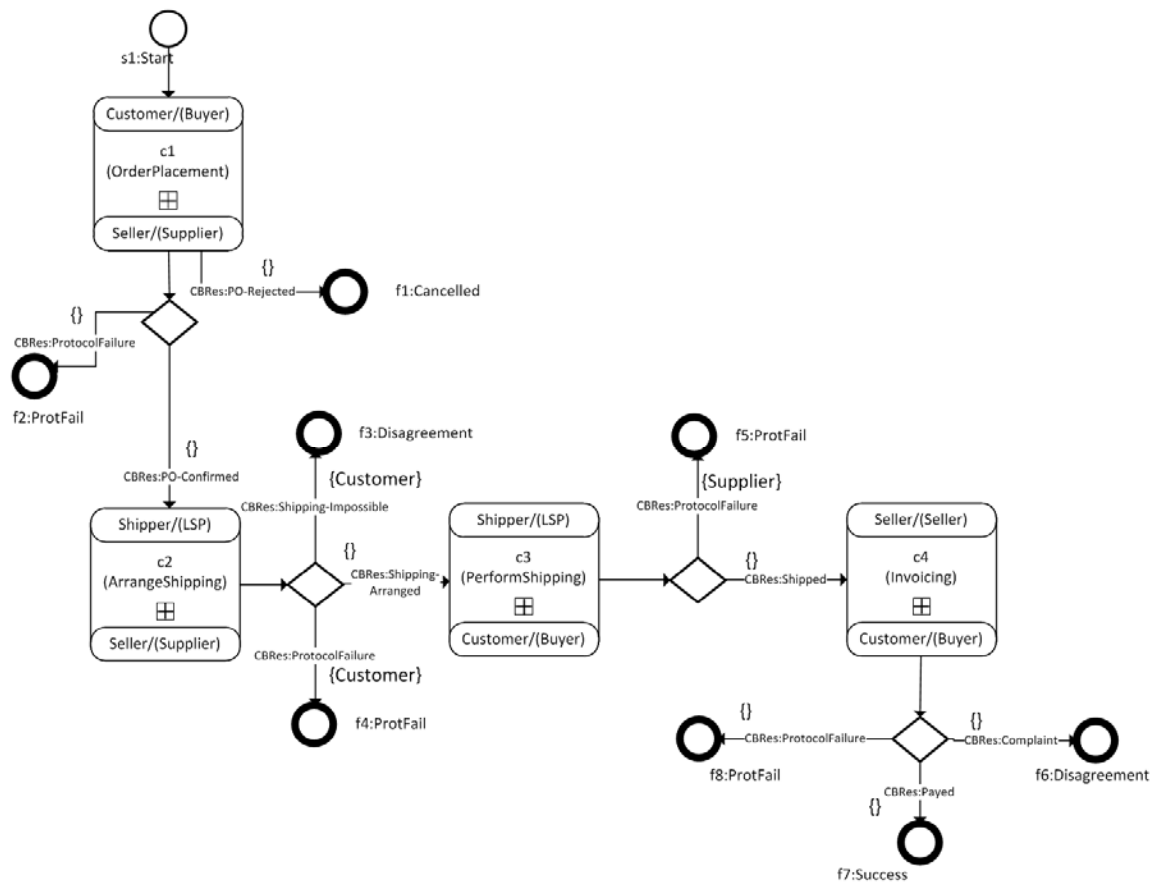


Figure 28: Sample SeqMP Choreography

SeqMP Rule 7: Escalation Assignment

SeqMP choreographies are designed to analyze synchronization deficits that result from activities without participation of a particular top-level SeqMP choreography role. Consider the SeqMP choreography depicted in Figure 28. The choreography starts out with an 'OrderPlacement' call choreography performed between the 'Customer' and the 'Seller' role. If the follow-on call choreography between the Seller and the Shipper fails, then the Customer is not automatically informed about that. However, the Customer may expect to participate once more in the overall SeqMP choreography, i.e., receive the product and invoice. Without explicit notification, the

Customer may wait unnecessarily long which constitutes a synchronization deficit. SeqMP choreographies are not designed to avoid or automatically resolve such deficits. Instead, they provide a sound framework for identifying these. Therefore, so-called escalation sets may be added to transitions (using curly braces) to identify roles that may suffer from synchronization deficits upon firing the respective transition. Escalation sets are sets of top-level choreography roles and can intuitively be characterized as follows:

"If a role has already participated in the overall choreography and may participate in the future and is not participating in the current component choreography and if a transition is taken that excludes that particular role from further participation, then the role is to be included in the escalation set of that particular transition."

For example, Figure 28 shows the escalation set '{Customer}' for the decision transition after component choreography 'c2' with guard 'CBRes:ShippingImpossible'. This escalation set basically says that the Customer is the only one that may suffer (in the sense of having an information deficit) from the premature termination of the overall choreography upon firing this transition. The Seller and Supplier role do not have a synchronization deficit because they participate in the source component choreography and therefore both have knowledge about the component choreography's result.

More details of the standard framework for analyzing information deficits are described in (Schönberger, et al., December 2010) and configurable analysis features are available in (Schönberger, et al.).

SeqMP Rule 8: Connectedness

Without considering the direction of transitions, any state of a SeqMP choreography model **MUST** be connected to the start state and all the end states of the same model.

SeqMP Rule 9: Guard Constraints

For the set of decision transitions as used in SeqMP Rule 6 the following holds true:

- **Completeness:**
The disjunction of the guards of all decision transitions must evaluate to true for any result of the evaluated component choreography.
- **Disjointness:**
No two guards of decision transitions may both evaluate to true for the same result of the evaluated component choreography.

SeqMP Rule 10: Producibility

SeqMP choreography models must be producible according to the rules of this section.

5.4 BPMN 2.0 Compliance

This specification DELIBERATELY is not strictly compliant to the BPMN choreography standard (OMG, January 2011), section 11). While the BPMN standard offers a notation for specifying general purpose choreography models, this specification strives for providing an easy-to-use choreography model that is tailored to the specific needs of B2Bi. During an analysis of the use cases defined in the RosettaNet RIG library a wealth of choreography concepts was discovered in BPMN that lends itself well to B2Bi choreography modeling. However, some BPMN constructs and rules were identified that hinder straightforward modeling of B2Bi choreographies and yet some other BPMN constructs proved to be superfluous. Consider further that concepts of dedicated B2Bi choreography standards such as ebBP, UMM or former RosettaNet deliverables are not reflected in the BPMN choreography standard. For example, the concept of PIPs (or BusinessTransaction in ebBP/UMM terminology) that require a protocol for ensuring information alignment and that are configurable according to so-called PIP performance controls is not reflected in BPMN choreographies. Although PIP logic may be encodable in BPMN sub-choreographies, the MCC phase 2 team decided to simply interpret choreography tasks as PIPs and to use the common PIP performance controls to configure execution. Similarly, standard expression languages to capture the result of PIPs (as available in ebBP or UMM) are not available in BPMN choreographies. On the other hand, some constraints defined for BPMN choreographies hinder B2Bi choreography specification. For example, (OMG, January 2011), section 8.3.13 says that “A source Gateway [of a conditional sequence flow] MUST NOT be of type Parallel or Event.” However, conditional expressions after join nodes may be convenient to determine the next choreography states depending on the results achieved in a parallel structure (cf. ExecChor Rule 9: Parallel Composition). Finally, a large set of BPMN constructs not described in section 5.2 (because those were not needed for modeling the use cases taken from the RosettaNet RIG library) shows that BPMN choreographies offer much more functionality than needed for common B2Bi scenarios. In our opinion, it does not make sense to confront B2Bi process modelers with the complexity of those concepts without clear value attribution (although these may be leveraged for cartography choreographies if need be).

Despite the listed issues with using the BPMN choreography standard for B2Bi choreography modeling ‘AS IS’, we believe that it can be a valuable part in a B2Bi tool-chain (cf. section 5.1.4) due to its common notation and expected tool support. In that sense, the above sections define a kind of a ‘BPMN choreography profile’ that is dedicated to B2Bi. The very purpose of that ‘profile’ is providing a visual notation for discussing B2Bi choreography contents and for determining the admissible sequences of PIPs and component choreographies. Obviously, extending, restricting and violating the BPMN choreography standard was necessary to tailor the “profile” to B2Bi needs. Also note that we did not even try to visually represent all the details that are necessary for B2Bi choreographies such as PIP performance controls or business document versions. Instead, we believe that adding those technical details is better done at a lower-level abstraction layer such as textual choreographies in the form of ebBP specification or even configurations of business service interfaces.

Below, we list the most important extensions and violations of the BPMN standard without claiming completeness. Restrictions on the use of BPMN choreography constructs are implicitly given in section 5.2 and adaptations of the execution semantics to B2Bi settings are predominately laid out in sections 5.3.2 and 5.3.3.

The following BPMN extensions have been defined (some of which also are violations):

- Choreography tasks are interpreted as PIPs that require the execution protocol defined in MCC phase 1 (cf. *Construct Advice 1: Representing PIPs*).
- Expression languages for evaluating the result of PIP executions have been imported from ebBP (cf. *Construct Advice 4: Representing Transitions*).
- Expression languages for capturing the result of component choreographies and parallel structures have been defined (cf. *Construct Advice 4: Representing Transitions* and *ExecChor Rule 9: Parallel Composition*).
- Basic rules for labeling top-level choreographies have been defined (*Construct Advice 5: Representing Choreographies*).
- A notation for role mapping from choreography roles to subordinate choreography roles has been defined (cf. *Construct Advice 1: Representing PIPs*, *Construct Advice 5: Representing Choreographies*, *Construct Advice 9: Representing Component Choreographies*).
- Reset semantics for timers that may be entered from an event-based choice several times have been enabled (cf. *Construct Advice 10: Representing Timeouts*).

The following BPMN rules are violated:

- This specification allows for adding guards to transitions that are not visualized with a so-called “mini-diamond marker” as defined in (OMG, January 2011), section 8.3.13:
“A conditional outgoing Sequence Flow from an Activity MUST be drawn with a mini-diamond marker at the beginning of the connector.”
- This specification does not define an evaluation order of conditions of an exclusive gateway. This contradicts (OMG, January 2011), section 13.3.2:
“In order to determine the outgoing Sequence Flows that receives the token, the conditions are evaluated in order.”
- This specification does not distinguish between collapsed sub-choreographies and call choreographies by means of “LINE THICKNESS” as BPMN does. Instead, this specification only defines collapsed Call Choreographies and expanded sub-choreographies. That does not mean, however, that tool vendors aren’t allowed to implement such folding functionality. This specification unifies these two concepts and provides the same functionality by simply distinguishing between implicit and explicit role mapping. Therefore, the following visualization constraints defined in (OMG, January 2011), section 11.4.3 are disregarded:
“If the Call Choreography calls a Choreography, then there are two options:
- *The details of the called Choreography can be hidden and the shape will be the same as a collapsed Sub-Choreography, but the boundary of the shape MUST have a thick line (see Figure 11.25).*
- *The details of the called Choreography can be shown and the shape will be the same as an expanded Sub-Choreography, but the boundary of the shape MUST have a thick line (see Figure 11.26).”*

5.5 Choreography Profiles

This specification acknowledges that there is more than one single B2Bi choreography methodology and more than one suitable tool-chain (see introduction to section 5). Taken together with the deficiencies of BPMN choreographies for visualizing choreographies identified in the last, this calls for the possibility to incorporate additional choreography languages into the RosettaNet Choreography Methodology. For example, process modelers may want to use UMM (UN/CEFACT, 2006) or BCL (Zapletal, et al., September 2009) for visually specifying choreographies as these languages offer dedicated B2Bi concepts.

Hence, this specification allows for defining choreography profiles in order to accommodate YOUR B2Bi process and not to require you to adopt your process to this specification.

In order to define a choreography profile, the following information **MUST** be provided:

- At least one tool-chain for bridging the gap between business process models and implementation systems must be described. As a minimum, this includes an identification of the relevant technologies to be used and the purpose of using the particular technologies, i.e., whether models specified in a particular language are to be used as cartography, executable, SeqMP choreographies or some other special-purpose choreography modeling style.
- The concepts for choreography representation **MUST** be described in detail. Although those concepts deviate from language to language, the following **MUST** be provided:
 - How PIPs are represented.
 - How choreographies and component choreographies are represented.
 - How event-based and document-based control flow routing is represented. This includes the identification of admissible expression languages for defining guards.
 - What kind of control flow concepts are available.
- The grammar for composing choreographies from choreography representation elements **MUST** be defined. This can either be done by providing a formal specification, by providing semi-formal examples and explanatory text (as done in this specification) or by providing informal considerations only.

6. Extension Points

Apart from allowing for choreography profiles (see section 5.5) the following options for extending this specification are identified:

1. **Deriving ebBP choreographies**

ebBP is a textual XML-based choreography language that is tailored to the needs of B2Bi. While textual languages may not be ideal for modeling, ebBP is a promising candidate for becoming an interchange format for choreography models. So, if a visual BPMN choreography model created in some BPMN tool needs to be imported into a UMM tool, then ebBP lends itself well for serving as an interchange format. Also, ebBP may be used for enriching a visual choreography model defined by business process models with technical details that are necessary for creating implementations. Examples for such technical details are PIP performance controls (such as `isAuthenticationRequired`, `isReliableMessagingRequired` etc.) and concrete business document versions.

For reference, we provide ebBP representations of the samples provided in section 8. However, an extension point is providing algorithms and implementations for converting various choreography formats (and modeling styles) into ebBP representations.

2. **Deriving implementations**

The choreography modeling styles defined in section 5.3.2 and 5.3.3 have clear semantics. In particular, 'executable choreography' models precisely define the set of admissible PIP executions. Hence, part of BSI implementations are derivable. Examples for how such partial implementations can be generated as WS-BPEL orchestrations are given in (Schönberger, et al., November 2010) and (Schönberger, et al., 2010). This specification invites extensions that provide (semi-)automatic support for turning choreography models into implementations.

Note that a major rework of the BPMN representation for B2Bi choreographies as provided in this specification is not a foreseen extension point. The subset of BPMN constructs used in this specification have carefully been selected according to a B2Bi requirements analysis and the set of use cases contained in the RosettaNet RIG library. In particular, we deliberately did not try to visualize PIP performance controls in order not to overstrain process modelers. A flexible concept for parameterizing PIPs has been put forward in MCC phase 1 and this specification is built upon the MCC phase 1 deliverables.

6.1.1 Glossary

Term	Definition
Activity id	An id for identifying the control flow position of a choreography activity. Enables the use of the same PIP type within the same choreography model with different semantics.
ASN	Advance Ship Notice
B2Bi	Business-to-Business integration
BCL	Business Choreography Language
BPEL4Chor	An alternative proposal for textual choreography specification based on WS-BPEL.
BPMN 2.0	Business Process Modeling and Notation Specification, version 2.0, provided by the OMG – In this document the choreography section of BPMN is of major concern.
BPMN 2.0 choreography task	A visual BPMN construct that is used for modeling the execution of a PIP within choreographies.
BPMN 2.0 exclusive gateway	A visual BPMN construct that is used as one option for modeling decisions within choreographies.
BPMN 2.0 event-based gateway	A visual BPMN construct that is used for representing event based choices (ebc).
BSI	Business Services Interfaces
business process models	High level specifications of business processes that may include physical, financial and information flows and according processing activities. Business process models serve as the basis for identifying choreography models which focus on the information exchange relationships between integration partners.
Cartography choreography	A choreography modeling style targeted at communication between integration partners. Cartography choreographies serve as a loose gathering of implementation requirements that need refinement for defining the set of admissible message exchange sequences.
CBRes	Collaboration Result
CGV	ConditionGuardValue, an expression language defined within the ebBP specification that can be used to

	capture generic outcomes of PIP executions such as <i>AnyProtocolFailure</i> or <i>ProtocolSuccess</i> .
choreography model	A concrete message exchange scenario modeled in a choreography language such as WS-CDL or ebBP. An ebBP model is a choreography model.
Component Choreography	A choreography that may be executed within other choreographies. BPMN 2.0 expanded sub-choreographies or BPMN 2.0 collapsed call choreographies are used as visualizations of component choreographies where call choreographies have a type and may be used in several different choreographies. The visualization of collapsing sub-choreographies or expanding call choreographies is not defined in this specification is deliberately left to tool implementations. Tool providers are expected to care for distinguishable visualizations of expanded call and sub-choreographies as well as collapsed call and sub-choreographies (line thickness alone is not sufficient).
ebBP	ebXML Business Process Specification Schema
ebBP model	A concrete specification of a B2Bi choreography in the ebBP format. ebBP models are choreography models.
ebc	Event-based choice, describes a state of choreographies in which multiple events may occur. Triggering PIPs, component choreographies or timeouts are eligible events.
ebc timer	Event-based choice timer
ExecChor	Executable choreography
Executable choreography	A choreography modeling style for binary/bilateral choreographies that exactly defines the set of admissible message exchange sequences (hence executable).
LSP	Logistics Service Provider
MCC	Message Control and Choreography, the RosettaNet effort for defining the application of choreography technology to RosettaNet PIP-based processes.
MCC phase 1	The MCC phase that specifies the execution of single PIPs.
MCC phase 2	The MCC phase that describes the modeling of complex multi-PIP interactions as choreographies.

orchestration model	A model of an executable process of one interaction partner. Several orchestration models commonly implement a choreography model.
PIP	Partner Interface Process
PIP communication role	The generic <i>requester</i> and <i>responder</i> roles that capture the communication function of the integration partners, i.e., whether they send the PIP business document or receive the PIP business document. (Note that new PIP definitions contain exactly one single business document)
PIP functional role	The PIP specific functional roles of the integration partners that captures the purpose of interaction, e.g., <i>Buyer</i> and <i>Seller</i> .
RIG	RosettaNet Implementation Guidelines
SeqMP choreographies	Sequential Multi-Party Choreographies
Sequential Multi-Party Choreographies	A choreography modeling style that concatenates binary/bilateral choreography models such that these lend themselves well to analysis.
Timers	Timer symbols specify timeout events in choreographies and are available as component choreography timers or event-based choice timers (see section 5.2). PIP timeouts are not represented as timer symbols as these are part of the common PIP parameterization defined in MCC phase 1.
UMM	UN/CEFACT's Modeling Methodology
visual model	A model that leverages visual constructs to express meaning.
WS-BPEL	Web Services Business Process Execution Language - A specification of a Web Services orchestrations language provided by OASIS.
XPath1	Refers to the original W3C XPath language (http://www.w3.org/TR/xpath/) which is an language for making expressions and statements about XML documents.
XPath2	Refers to the W3C XPath language 2.0 (http://www.w3.org/TR/xpath20/)

7. References

Decker Gero [et al.] BPEL4Chor: Extending BPEL for Modeling Choreographies July [Conference] // Proceedings of the 2007 IEEE International Conference on Web Services (ICWS). - Salt Lake City, Utah, USA : IEEE, July 2007.

IETF Internet Engineering Task Force Request for Comments 2119, Key words for use in RFCs to Indicate Requirement Levels [Buch]. - 1997. - <http://www.ietf.org/rfc/rfc2119.txt>.

OASIS ebXML Business Process Specification Schema Technical Specification [Buch]. - December 2006.

OASIS Web Services Business Process Execution Language [Buch]. - 2007.

OMG Business Process Model and Notation, v2.0 [Buch]. - January 2011.

OMG OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.3 [Buch]. - May 2010.

RosettaNet Description of Partner Interface Process® for 2A13 [Online] // www.rosettanet.org. - August 2006.

RosettaNet Description of Partner Interface Process® for 6A1 [Online] // www.rosettanet.org. - July 2006.

RosettaNet RIG for PIP 3B2: Notify of Advance Shipment [Online]. - 2003.

RosettaNet RIG for PIPs 3A4, 3A7, 3A8, 3A9 - Order Management in Japan Business Model Alignment Process [Online] // www.rosettanet.org. - April 2004.

RosettaNet RosettaNet Implementation Framework 2.0: Core Specification, ed. V02.00.01 [Buch]. - 2002. - http://www.rosettanet.org/dnn_rose/DMX/tabid/2979/DMXModule/624/Command/Core_ViewDetails/Default.aspx?EntryId=4730.

Schönberger Andreas and Wirtz Guido Configurable Analysis of Sequential Multi-Party Choreographies [Article] // forthcoming in International Journal of Computer Systems Science and Engineering / ed. Publishing CRL.

Schönberger Andreas and Wirtz Guido Sequential Composition of Multi-Party Choreographies [Conference] // Proceedings of the International Conference on Service-Oriented Computing and Applications (SOCA'10). - Perth, Australia : IEEE, December 2010.

Schönberger Andreas and Wirtz Guido Taxonomy on Consistency Requirements in the Business Process Integration Context [Conference] // Proceedings of 2008 Conf. on Software Engineering and Knowledge Engineering (SEKE'2008). - San Francisco : Knowledge Systems Institute, July 2008.

Schönberger Andreas and Wirtz Guido Towards Executing ebBP-Reg B2Bi Choreographies [Conference] // Proceedings of the 12th IEEE Conference on Commerce and Enterprise Computing (CEC'10). - Shanghai, China : IEEE, November 2010.

Schönberger Andreas Do We Need a Refined Choreography Notion? [Conference] // Proceedings of the 3rd Central-European Workshop on Services and their Composition, ZEUS 2011. - Karlsruhe, Germany : CEUR-WS, February 2011.

Schönberger Andreas, Pflügler Christoph and Wirtz Guido Translating Shared State Based ebXML BPSS models to WS-BPEL [Article] // International Journal of

Business Intelligence and Data Mining - Special Issue: 11th International Conference on Information Integration and Web-Based Applications and Services in December 2009. - 2010. - 4 : Vol. 5. - pp. 398 – 442.

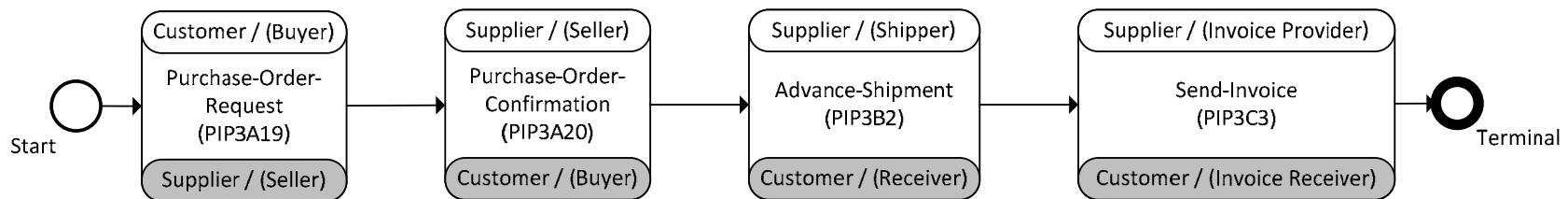
UN/CEFACT UN/CEFACT's Modeling Methodology (UMM): UMM Meta Model - Foundation Module Version 1.0 [Book]. - 2006.

W3C XML Path Language (XPath) 2.0 (Second Edition) [Book]. - 14 December 2010. - <http://www.w3.org/TR/xpath20/>.

Zapletal Marco, Motal Thomas and Werthner Hannes The Business Choreography Language (BCL) - a Domain-Specific Language for Global Choreographies [Conference] // Proceedings of the 5th 2009 World Congress on Services (SERVICES 2009 PART II), International Workshop on Services Computing for B2B (SC4B2B). - Bangalore, India : IEEE, September 2009.

8. Appendix

8.1 Classic Order to Cash Scenario - Cartography Style



BPMN 2.0 "Choreography Tasks" are used for specifying the use of a PIP.

For specifying the PIP-Type, the full type name and the short-hand identifier are admissible, i.e.,

either the local name of the corresponding XML root tag is used or the string "PIP" concatenated with the Cluster-Segment-PIP id.

PIP roles are identified via the so-called participant bands of choreography tasks. The band without fill is the RequestingRole of the PIP whereas the band with grey fill is the RespondingRole.

The identifier within the band denotes a role name of the choreography (Customer and Supplier in the example) and may be complemented with a role name of the PIP.

Hence the following two notations are admissible:

- 1) choreography role name / (PIP role name)
- 2) choreography role name

The first notation makes clear which functional PIP role a choreography role takes.

The second notation may be used as a shortcut when the functional PIP role is clear from the communication role (Requesting/RespondingRole).

For visualization purposes, the names of concrete participating companies may either replace choreography role names or be bound to choreography role names by means of a component choreography as illustrated below. There, an existing choreography is simply interpreted as a component choreography of a new top-level choreography. For that top-level choreography, new choreography roles (carrying the names of the participating companies) are defined that then are mapped to the role names of the component choreography by reusing the relabeling notation defined above:

- a) top-level role name / (component choreography role name1, component choreography role name2,...)
- b) top-level role name

While the full notation a) makes the mapping explicit the short-hand notation b) simply expresses that the top-level choreography role name shall be matched for equality with the component choreography role names.

These samples are part of the RosettaNet Message Control and Choreography (MCC) standard. Therefore, all samples are based on RosettaNet PIPs. However, RosettaNet acknowledges that some integration scenarios may require the use of different business document definitions or PIP-like concepts such as UMM/ebBP BusinessTransactions.

These alternative business document (exchange) definitions may be used as long as the concepts laid out for RosettaNet PIPs within this standard are applied accordingly.

A "Cartography" choreography only specifies the happy path of an interaction. A cartography does specify how many times a particular PIP may be performed or which kind of actions have to be undertaken between two PIP executions. Error handling at the business level such as dealing with order rejections deliberately is not specified.

Obviously, the implementation of such a cartography choreography requires additional agreements between integration partners for exactly defining the formats and sequences of all message exchanges.

In so far, cartography style choreographies can be regarded as a communication means for business analysts and business modelers to identify the relevant PIPs and the ideal flow of PIP executions. Software engineers would use such a specification as (incomplete) requirements gathering for engineering the actual interactions.

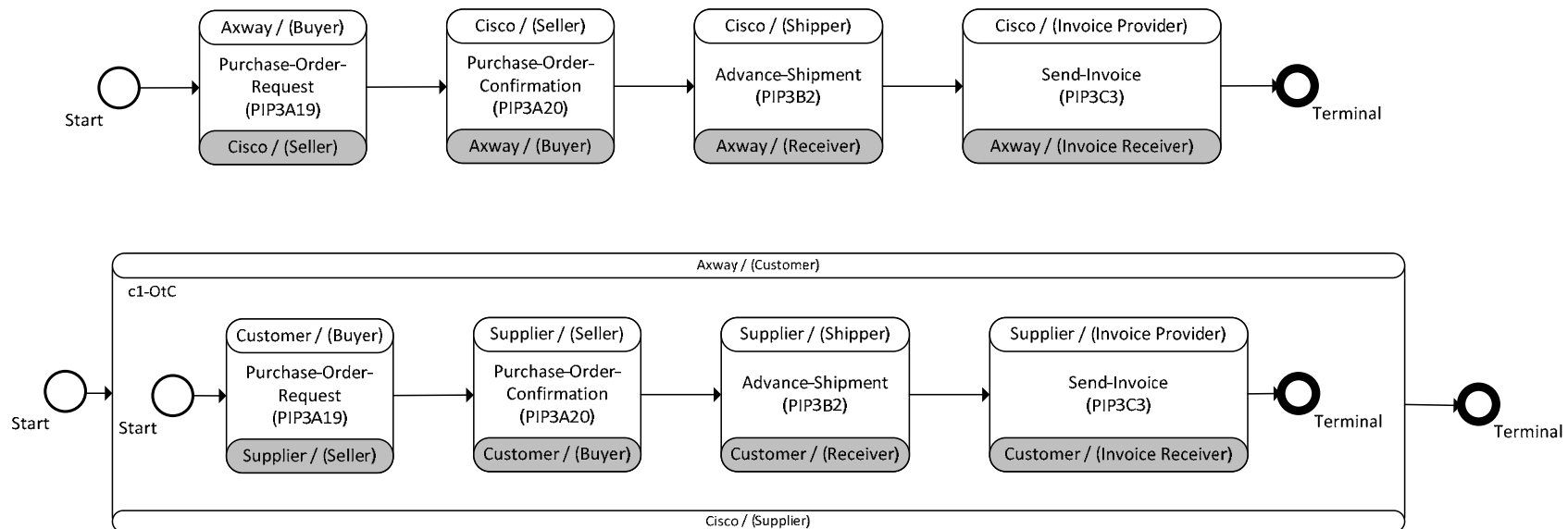
Note:

BPMN offers two different options for visualizing the relationship between parent choreographies and component choreographies.

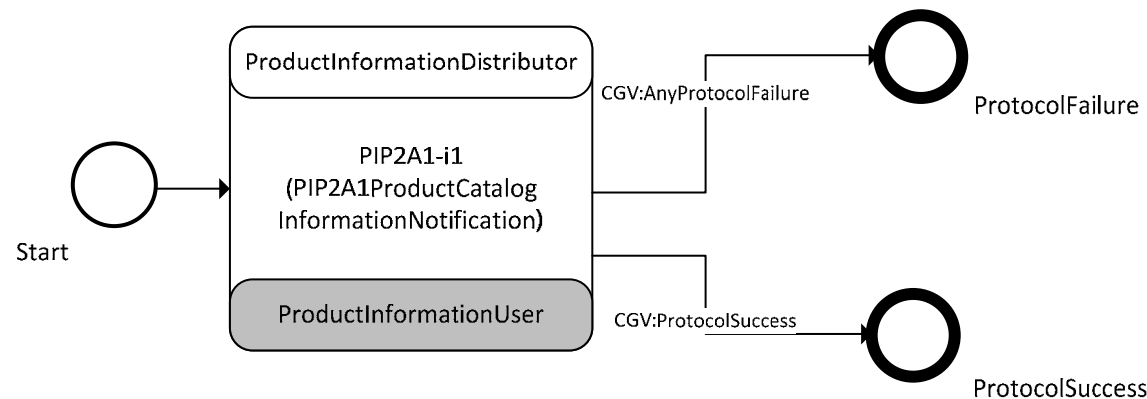
If the component choreography is implicitly embedded in the parent choreography then the term "sub-choreography" is used.

If the component choreography is called by name then the term "call choreography" is used.

For RosettaNet MCC, we only use expanded sub-choreographies and collapsed call choreographies. The term component choreography may refer to both, sub-choreographies and call choreographies. More details are given in the subsequent samples of this document.



See for reference: RosettaNet RIG 2A1 V02.00.00, 2A1 RIG V02_00_00.doc, pages 1-5



This sample represents a very simple choreography that consists of a single PIP the result of which are evaluated using guards. The definition of guards is borrowed from the ebXML BPSS (ebBP) specification.

The guards on the transitions marked with "CGV" (ConditionGuardValue) denote generic ebBP BusinessTransaction outcomes that can be used for RosettaNet PIPs as well.

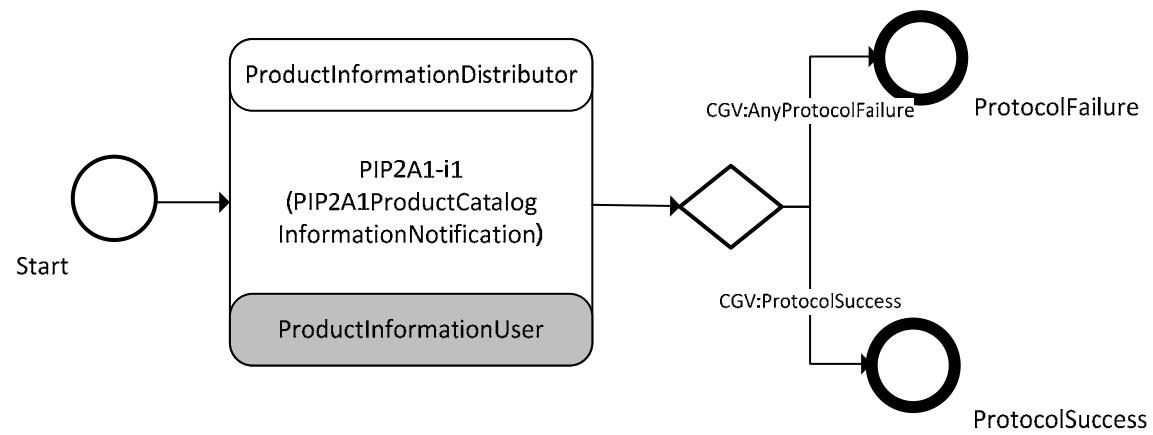
That means that there are rules for computing those values based on the BusinessTransaction execution protocol that has been adapted to executing PIPs.

Typical results are "AnyProtocolFailure" or "ProtocolSuccess".

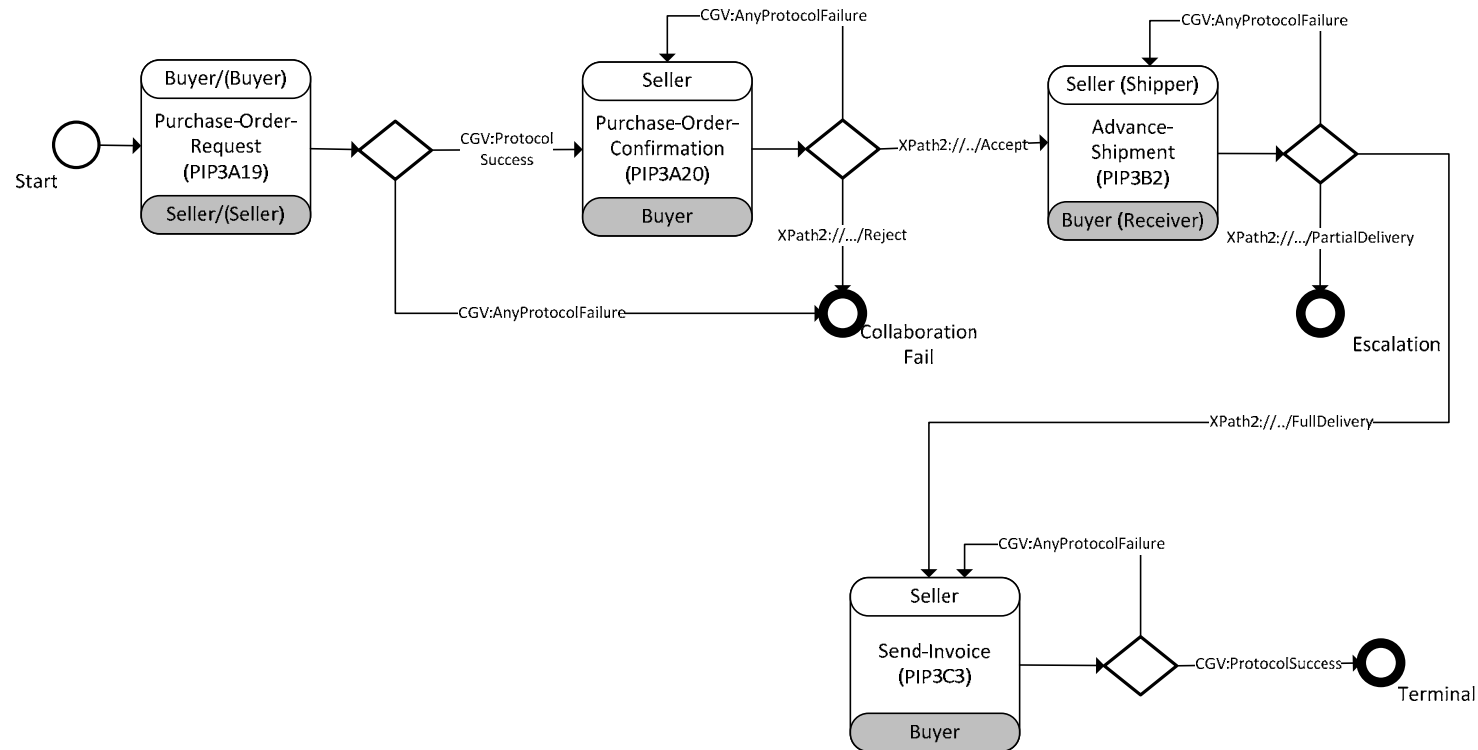
Guards marked with "XPath1" or "XPath2" denote XPath1 or XPath2 expressions defined on the exchanged business documents. Note that determining the result of a PIP based on business document contents requires the PIP execution protocol to have succeeded beforehand. Only then agreement of the integration partners with respect to the actual content of the business document can be assumed.

For a description of the full set of available expression languages, see the ebBP specification.

For visualization purposes, a "decision" node may be used to "gather" the transitions for evaluating a particular PIP (see below).



8.2 Classic Order to Cash Scenario - Executable Style



Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation.

An "Executable" choreography not only defines the PIPs to be used and the happy path of an interaction, it also defines the guards based on which the path through the interaction is taken. For representing the "decision" of which path to take next through the choreography, diamond shapes are used. For example, the overall interaction (choreography) depicted above terminates if a protocol failure is detected during executing PIP 3A19. To be "executable", the guards of a choreography must be defined such that they are complete and disjoint per PIP. That means that the guards used for evaluating the result of a

particular PIP must capture every possible result of the PIP execution and that no two guards may capture the same result. Additionally, executable choreographies must be defined between exactly two top-level choreography roles. This is due to synchronization problems that may result from performing binary PIPs in a multi-party setting. Finally, the grammar rules defined in the MCC phase 2 standard must be followed to create an "executable" choreography. By following these rules, it is possible to derive the admissible message exchange sequences BETWEEN integration partners automatically. However, a full integration with existing backend systems or business applications will still require the aid of software engineers. In so far, executable choreographies are envisioned to be used as a means for technically skilled business modelers or software engineers who participate in the choreography definition process to constrain the set valid implementations significantly.

The definition of guards is borrowed from the ebXML BPSS (ebBP) specification.

The guards on the transitions marked with "CGV" (ConditionGuardValue) denote generic ebBP BusinessTransaction outcomes that can be used for RosettaNet PIPs as well.

That means that there are rules for computing those values based on the BusinessTransaction execution protocol that has been adapted to executing PIPs.

Typical results are "AnyProtocolFailure" or "ProtocolSuccess".

Guards marked with "XPath1" or "XPath2" denote XPath1 or XPath2 expressions defined on the exchanged business documents.

For a description of the full set of available expression languages, see the ebBP specification.

This sample defines the RosettaNet Order-To-Cash scenario as follows:

The overall choreography is performed between a "Seller" role and a "Buyer" role. The entry point to the choreography is the start state at the left-hand side labeled "Start".

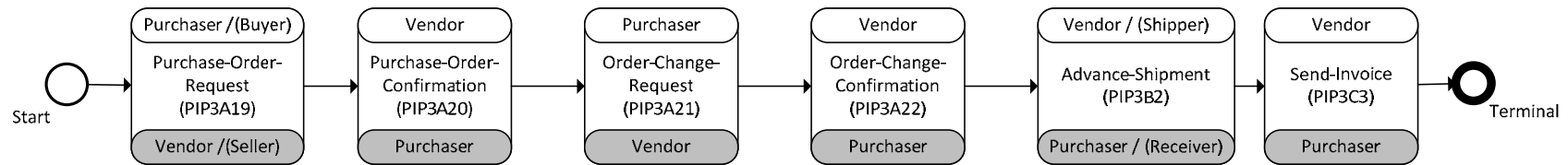
The initial PIP to be performed is PIP 3A19. From the participant bands we know that the "Buyer" is the initiator of the PIP because the Buyer's band has a white fill whereas the Seller's band has a grey fill. Moreover the choreography roles are bound to the PIP roles. In case of PIP 3A19 the choreography roles and the PIP roles are the same. So, Buyer is bound to Buyer. This is different for PIP 3B2 where the choreography role Buyer is bound to the PIP role Receiver. Basically we also could have left out the PIP roles just as in PIP 3A20 and 3C3 because we use the standard role assignment of the PIPs. Just look at PIP 3C3. The standard initiator of this PIP is the PIP role "Invoice Provider". So by giving the Seller's participant band a white fill we assign him the initiator role which implies that the Seller takes the Invoice Provider role. Accordingly, it is clear that the Buyer has PIP 3C3's "Invoice Receiver" role.

Only if PIP 3C3 was reversed in some special scenario where the Invoice Provider is not the PIP's initiator, i.e., a Self-Billing Invoice scenario, then the association of the choreography roles Buyer and Seller to the PIP roles would have to be made explicit.

After PIP 3A19 has been performed, its result is evaluated using a decision node. So-called ConditionGuardValues taken from the ebBP specification are used to capture the basic protocol outcomes. We know that ConditionGuardValues are used from the acronym CGV that is prepended before the actual value and separated using a colon. The CGV value "AnyProtocolFailure" captures any protocol failure such as timeouts, signature validation errors etc. (detailed communicating automata definitions for determining the results of a PIP are given in the MCC phase 1 specification). "ProtocolSuccess", in turn, denotes the fact that the PIP execution protocol completed successfully. Note that ProtocolSuccess does not say a thing about the actual business document. It just states that the integration partners have agreed upon the successful exchange as well as the contents of the document and that the relevant Quality-of-Service constraints have been respected. In case AnyProtocolFailure is detected the choreography is terminated and the end state CollaborationFail is reached. Note that both choreography roles (Buyer and Seller) know that this state is reached. Otherwise, the next PIP 3A20 is entered. To be more precise a state is reached in which the Seller (as the initiator of PIP 3A20) is allowed to trigger PIP 3A20. The point in time the Seller decides to do so is unknown. Note also, that the two guards used for evaluating PIP 3A19 are disjoint and complete. There is no PIP 3A19 result for which both guards evaluate to true and there is no result for which both guards evaluate to false.

After PIP 3A20 another decision is used to evaluate the PIP's results. If a protocol failure is detected the PIP shall be performed again. Note that the Seller does not have to trigger the next PIP 3A20 instance immediately. If no protocol failure happened, XPath2 expressions are used for evaluating the contents of the exchanged business documents. A necessary precondition for using business document based expressions is that the PIP succeeded from a protocol perspective, i.e., that CGV:AnyProtocolFailure evaluates to true. Further, in this sample, the XPath2 definitions are not complete. Software engineers would have to participate in the deployment process to refine the XPath2 expressions such that they are valid relative to the business document definitions. In case the exchanged business document of PIP 3A20 carries data for which the "XPath2:///./Accept" expression evaluates to true the next PIP 3B2 is enabled. In case "XPath2:///./Reject" evaluates to true the choreography is terminated. Again all guards used for evaluating PIP 3A20 taken together shall be complete and any pair of these guards shall be disjoint.

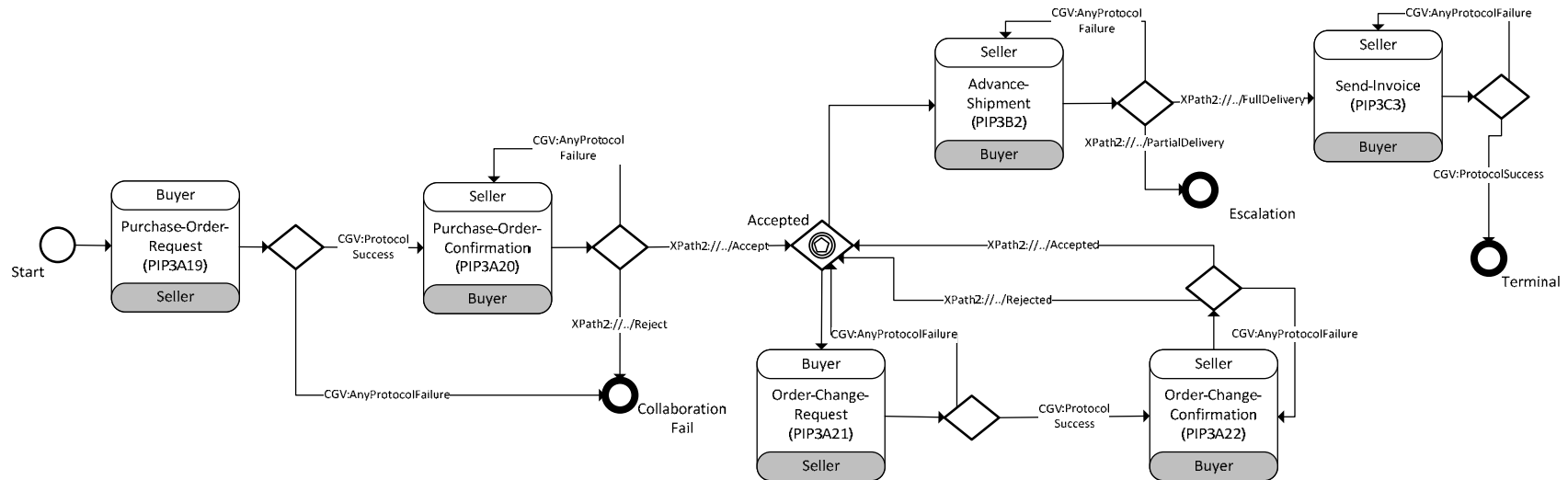
8.3 Extended Order to Cash Scenario - Cartography Style



Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and basic documentation of the characteristics of "Cartography" choreographies.

The example defined above gives the "Cartography" choreography representation of RosettaNet's "Extended Order To Cash" sample scenario.

8.4 Extended Order to Cash Scenario - Executable Style

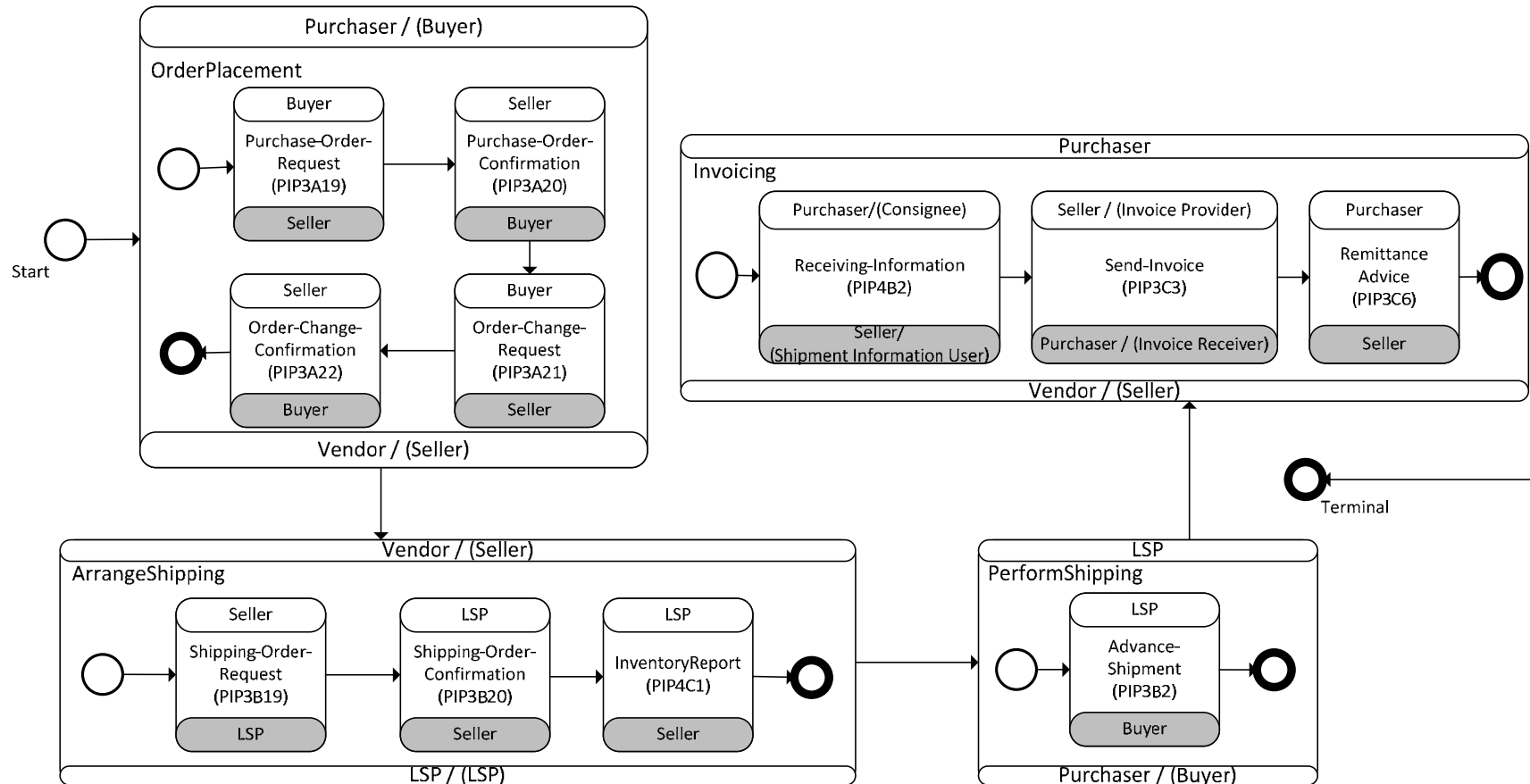


Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and sheet "2-ClassicOrderToCash-Executable" for basic documentation of the characteristics of "executable" choreographies.

The example defined above gives the "executable" choreography representation of RosettaNet's "Extended Order To Cash" sample scenario.

A new kind of node in this sample is the event-based choice named "Accepted" that is entered after having performed PIP 3A20 successfully with an "Accept" flag in the business document. Obviously, the XPath2 definition of the corresponding guard is not complete, but a placeholder for an expression that can be evaluated against a concrete business document. The event-based choice "Accepted" represents that multiple follow-on paths through the interaction may be taken, depending on which "event" occurs first. In this sample, either the Buyer may trigger a PIP 3A21 or the Seller may trigger a PIP 3B2. It is assumed that the underlying execution platform provides the necessary functionality for avoiding that both PIPs are performed at the same time within the same choreography instance.

8.5 Order to Cash with LSP Scenario - Cartography Expanded Style



Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and basic documentation of the characteristics of "Cartography" choreographies.

This sample gives the "Cartography" choreography representation of RosettaNet's "Extended Order To Cash with Logistic Service Provider" scenario.

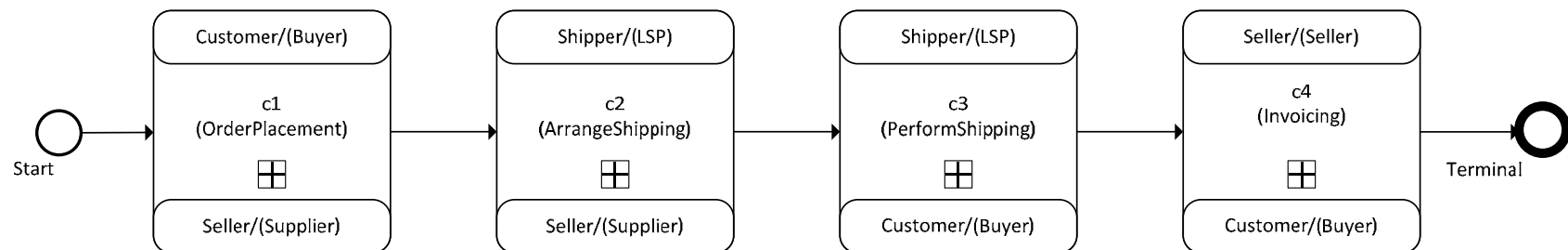
This sample introduces choreographies with more than two top-level choreography roles as new concept, i.e., "Purchaser", "Vendor" and "LSP" (Logistic Service Provider). Note that this does not prevent a company to take more than one of the top-level roles in a concrete deployment. The sub-choreographies are arranged such that exactly two top-level roles participate in each sub-choreography. However, this is not a necessary requirement and more than two top-level roles may participate in a particular sub-choreography. Note that, as a "Cartography" choreography is depicted in this sample, the problem of notifying top-level roles in case the interaction terminates prematurely is not handled at all. For example, in case the "Arrange-Shipping" sub-choreography does not succeed, the obligation of informing the "Purchaser" role about that is not clearly assigned to neither the "Vendor" nor the "LSP".

8.6 Order to Cash with LSP Scenario - Cartography Collapsed Style

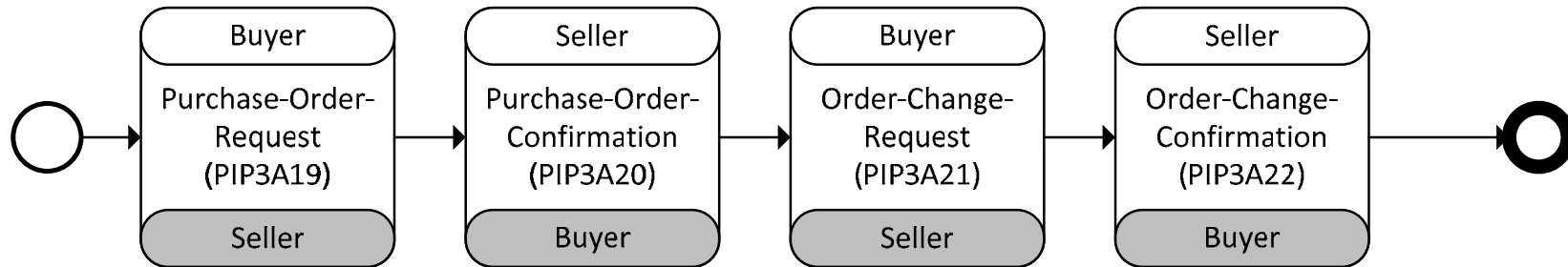
Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and basic documentation of the characteristics of "Cartography" choreographies.

The sample below describes the same choreography specified on sheet "5-OrderToCashLSP-Expanded-Cartography". The difference is in the presentation of the top-level choreography ("OtCWithLSP-Global") that uses so-called BPMN call choreographies to hide the activities of the component choreographies. Component choreographies are included by name that is given in parentheses after the instance identifier (c1 to c4) of the respective call choreography. The "expanded" version of this choreography (cf. sheet 5) can be derived by putting the called choreographies ("OrderPlacement", "ArrangeShipping"...) into expanded sub-choreography shapes and replacing the corresponding call choreography shapes with these expanded sub-choreography shapes accordingly.

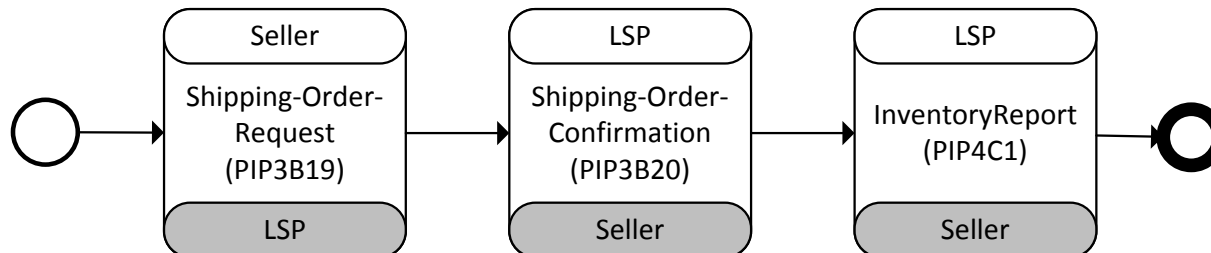
8.6.1 Order to Cash with LSP –Global



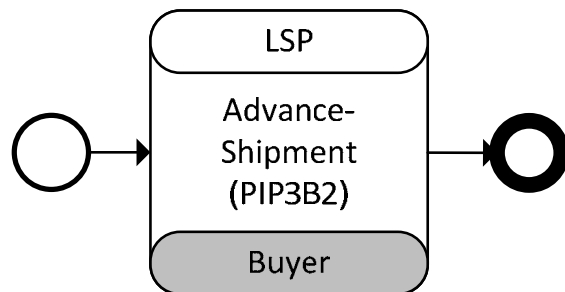
8.6.2 OrderPlacement



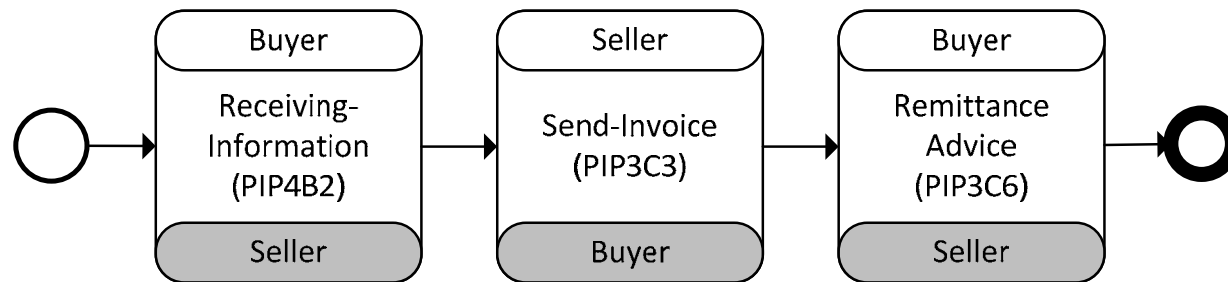
8.6.3 ArrangeShipping



8.6.4 PerformShipping



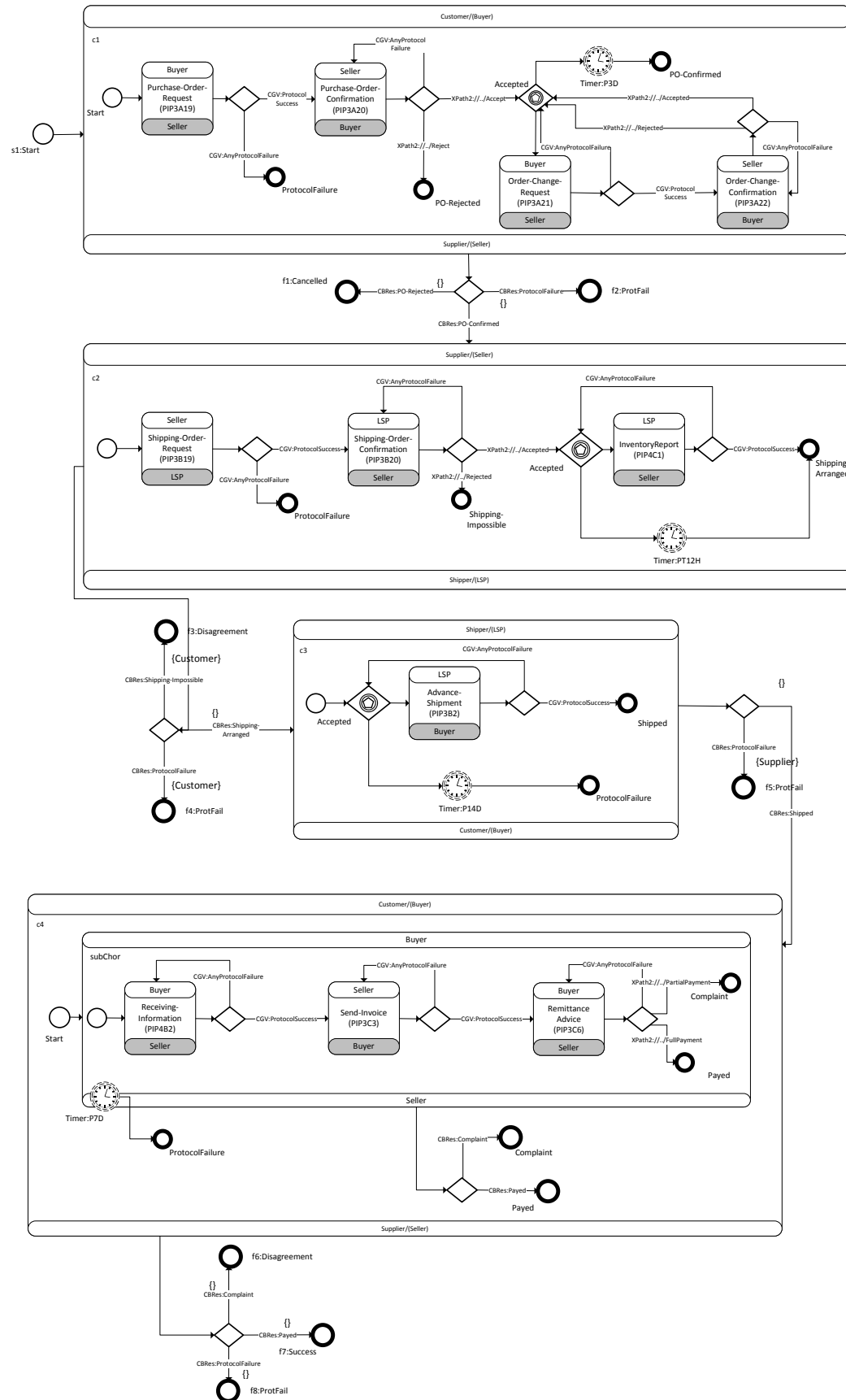
8.6.5 Invoicing



8.7 Order to Cash with LSP Scenario - Expanded Sequential Multi Party Style

Note:

Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and sheet "2-ClassicOrderToCash-Executable" for documentation on determining the path through a choreography using decision nodes and guards.



Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and sheet "2-ClassicOrderToCash-Executable" for documentation on determining the path through a choreography using decision nodes and guards.

Note:

This sample gives the "Expanded Sequential Multi-Party" (expanded SeqMP) choreography representation of RosettaNet's "Extended Order To Cash with Logistic Service Provider" scenario.

The following features are characteristic for SeqMP choreographies:

1) Executable choreographies as components

SeqMP choreographies are composed from several executable choreographies that are performed by at least three top-level choreography roles (otherwise it is just an executable choreography composition). Due to the nature of executable choreographies, the result of each component choreography can be considered to be synchronized between the two roles that perform the component choreography. Therefore the result of each component choreography (captured by different end nodes) is used to determine the follow-on component choreography to be performed. The top-level roles performing two subsequent component choreographies may not be the same. In the sample, component choreography "c1" is performed between the top-level roles "Customer" and "Supplier" whereas component choreography "c2" is performed between "Supplier" and "Shipper".

2) No parallel composition

The component choreographies of a SeqMP choreography may not be composed in a parallel structure. Parallel composition is ONLY available for executable choreographies as the top-level roles performing the concurrent component choreographies are the same. This in turn simplifies synchronization.

3) No SeqMP component choreographies

As the top-level roles performing two subsequent component choreographies may not be the same, synchronization issues may arise. In the sample, the Supplier and the Shipper may end up in end state "Shipping-Impossible" of component choreography c3 which results in the top-level end state "f3:Disagreement". In that case, the top-level "Customer" role may not gain awareness that the choreography already has been terminated, i.e., there is a synchronization problem with respect to the end state.

In consequence, using SeqMP choreographies as component choreographies of other SeqMP choreographies is disallowed because decision about the follow-on component choreography (based on the component choreography result) may not be synchronized among all participants.

4) Escalation sets

There are different ways of resolving synchronization issues as described above. For example, the issues may simply be ignored or one of the top-level roles may be assigned the task of synchronizing each of the remaining roles after each component choreography. Instead of enforcing a standard way of synchronization issue resolution, SeqMP choreographies just provide so-called escalation sets to capture these issues. An escalation set is a set of top-level choreography roles that is assigned to a transition (denoted as a pair of curly braces) that emerges from a decision node that is used for evaluating a component choreography result.

An escalation set can intuitively be characterized as follows:

"If a role has already participated in the overall choreography and may participate in the future and is not participating in the current component choreography and if a transition is taken that excludes that particular role from further participation, then the role is to be included in the escalation set of that particular transition."

A SeqMP choreography is different from an "Executable" choreography in the following sense:

1) SeqMP choreographies are not composable

Whereas SeqMP choreographies cannot be composed due to potential synchronization issues, an executable choreography may be used as a component choreography in other executable choreographies or in SeqMP choreographies.

This is due to the fact that the component choreographies (if any) of executable choreographies are performed by the exact two same top-level roles of that particular executable choreography.

2) SeqMP choreographies define so-called "escalation sets"

As there are only two top-level roles in an executable choreography there are not inter-role synchronization problems and hence there is no need for escalation sets.

A SeqMP choreography is different from a "Cartography" choreography in the following sense:

1) SeqMP choreographies use executable choreographies as component choreographies

Whereas cartography choreographies just capture a subset of the possible paths through an interaction, a SeqMP choreography captures the complete set of paths by mapping the different end states of an executable component choreography to follow-on

component choreographies or end states by using guards. Note that it is conceivable to embed an executable choreography or a SeqMP choreography into a cartography choreography.

2) Escalation sets for capturing synchronization issues are provided.

As cartography choreographies are only used to capture the admissible execution paths partly, an analysis for escalation sets as outlined above is incomplete at best. Hence no escalation sets are defined for cartography choreographies.

All in all, SeqMP choreographies are envisioned to be used as a means for technically skilled business modelers or software engineers who participate in the choreography definition process to constrain the set valid implementations significantly and to detect synchronization deficits.

Note:

This sample introduces timers and choreography result based routing as new concepts. Both concepts are not tied to SeqMP choreographies and may be used in executable and cartography choreographies as well. This is just the first sample that the two concepts appear in.

1) Timers:

Timers come in two flavors, component choreography timers and event-based choice timers. These two types of timers reflect functionality available in ebXML BPSS (ebBP) and BPMN 2.0 timer shapes are used to represent the according events.

a) Component Choreography timers:

Component choreography timers are added to the boundary of sub-choreography or call choreography shapes and specify a time interval or a date and time that complies to ISO 8601. A component choreography timer has a follow-on node that is reached when a timer runs out. If the timer is interrupting (solid outer line of the shape) then the currently active task of the component choreography (if any) is interrupted and the timer's follow-on node is immediately reached. If the timer is non-interrupting (dotted outer line of the shape) then the completion of the currently active task of the component choreography is waited for before the timer's follow-on state is reached.

b) Event-based choice timers

An Event-based choice timer (ebc timer) may be used as an alternative to choreography tasks after event-based choice gateways. ebc timers specify a time interval or a date and time and if the timer runs out before any of the alternative outgoing transitions of the respective event-based choice is fired then the follow-on state of the timer is reached. In the sample, in component choreography c2, a timer is used after having performed PIP 3B2 successfully to avoid the situation that the Seller

waits for the Shipper's InventoryReport indefinitely. If PIP 4C1 is not performed within 12 hours then the end state ShippingArranged is entered.

It may happen that an event-based choice of an interaction is entered multiple times, e.g., if there is iterative behavior. In that case, it has to be decided whether or not a follow-on ebc timer is reset or not in case an event-based choice is not reached for the first time. The default semantics is that an ebc timer is not reset. If reset is needed then a reset flag has to be added to the transition that enters the corresponding event-based choice.

Note that an ebc timer always is non-interrupting.

2) Routing based on choreography results

As the results of executable choreographies are synchronized among its participating roles, the end states of an executable component choreography may be used for routing purposes. Similarly to decision nodes after choreography tasks (cf. sheet "2-ClassicOrderToCash-Executable"), decision nodes after component choreographies determine the follow-on states by using guards. These guards capture the results of the component choreographies by building boolean expressions from the names of the end states of the component choreographies. Such a boolean expression evaluates to true if the component choreography terminates in the corresponding end state. The typical boolean operators may be used to build more complex guards. For making clear that the result of a component choreography is used for routing, the string "CBRes" must be prepended before the boolean expressions.

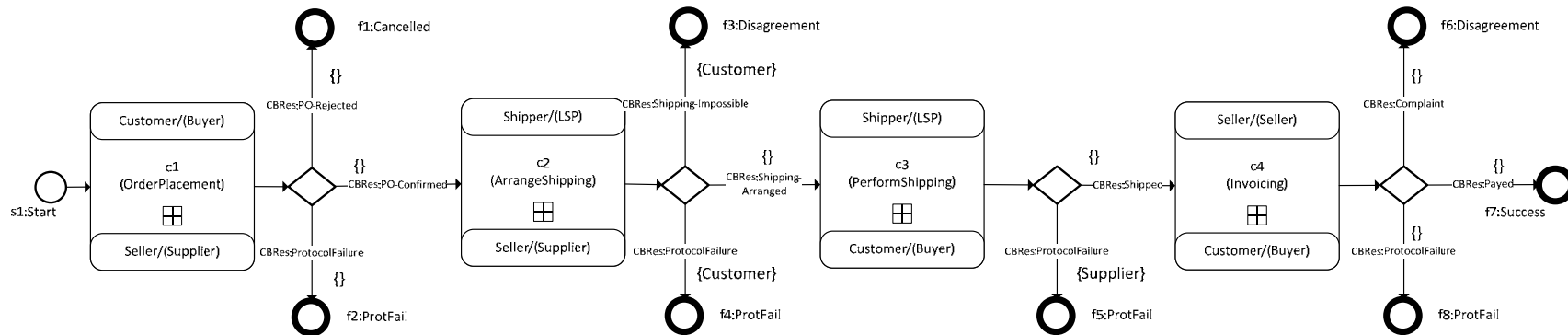
8.8 Order to Cash with LSP Scenario - Collapsed Sequential Multi Party Style

Note:

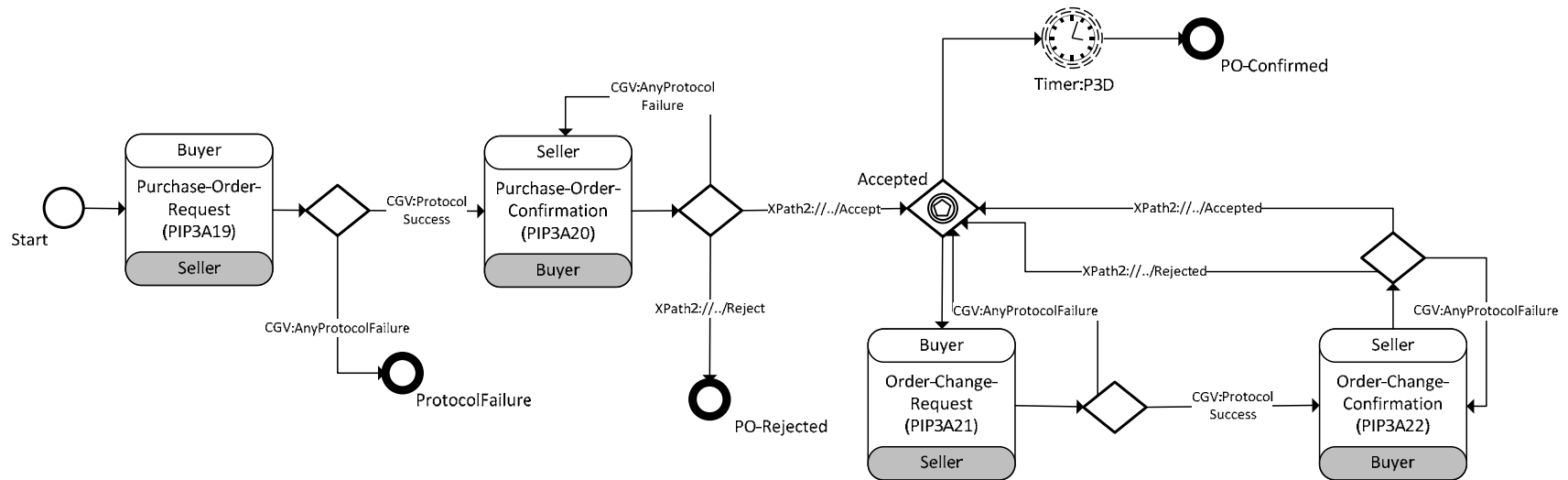
Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and sheet "2-ClassicOrderToCash-Executable" for documentation on determining the path through a choreography using decision nodes and guards. See sheet "6-OrderToCashLSP-Collapsed-Cartography" for documentation of collapsed call choreographies and sheet "7-OrderToCashLSP-Expanded-SeqMP" for documentation on the characteristics of SeqMP choreographies.

This sample defines exactly the same information as the sample of sheet "7-OrderToCashLSP-Expanded-SeqMP". The only difference is that the component choreographies are hidden within the top-level choreography by using call choreographies that reference the respective component choreographies by name.

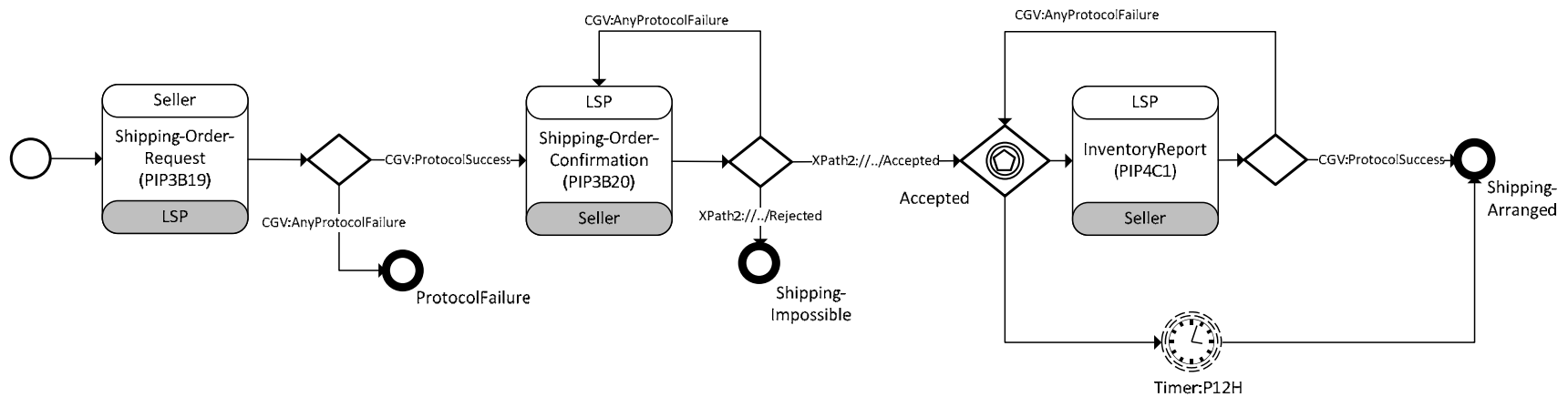
8.8.1 Order to Cash with Logistic Service Provider -Global



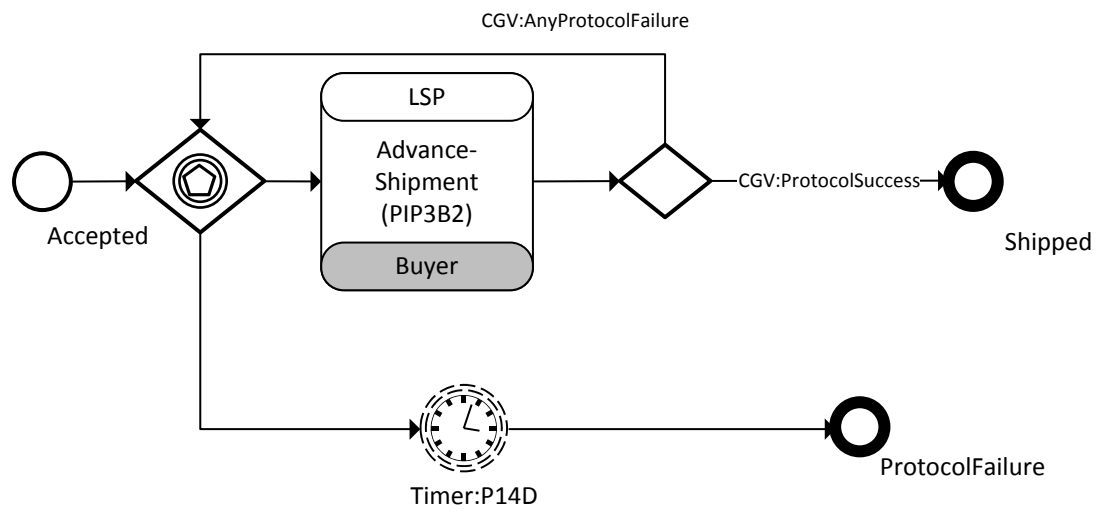
8.8.2 OrderPlacement



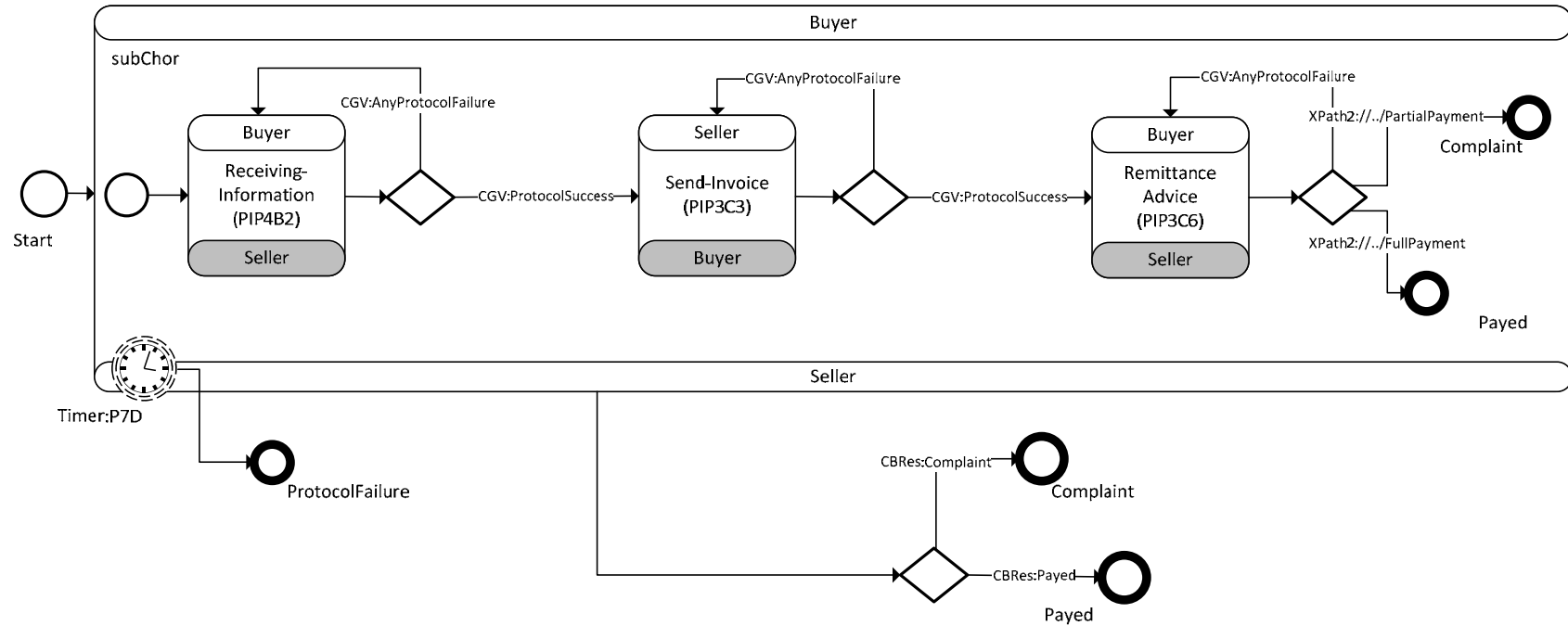
8.8.3 ArrangeShipping



8.8.4 PerformShipping



8.8.5 Invoicing



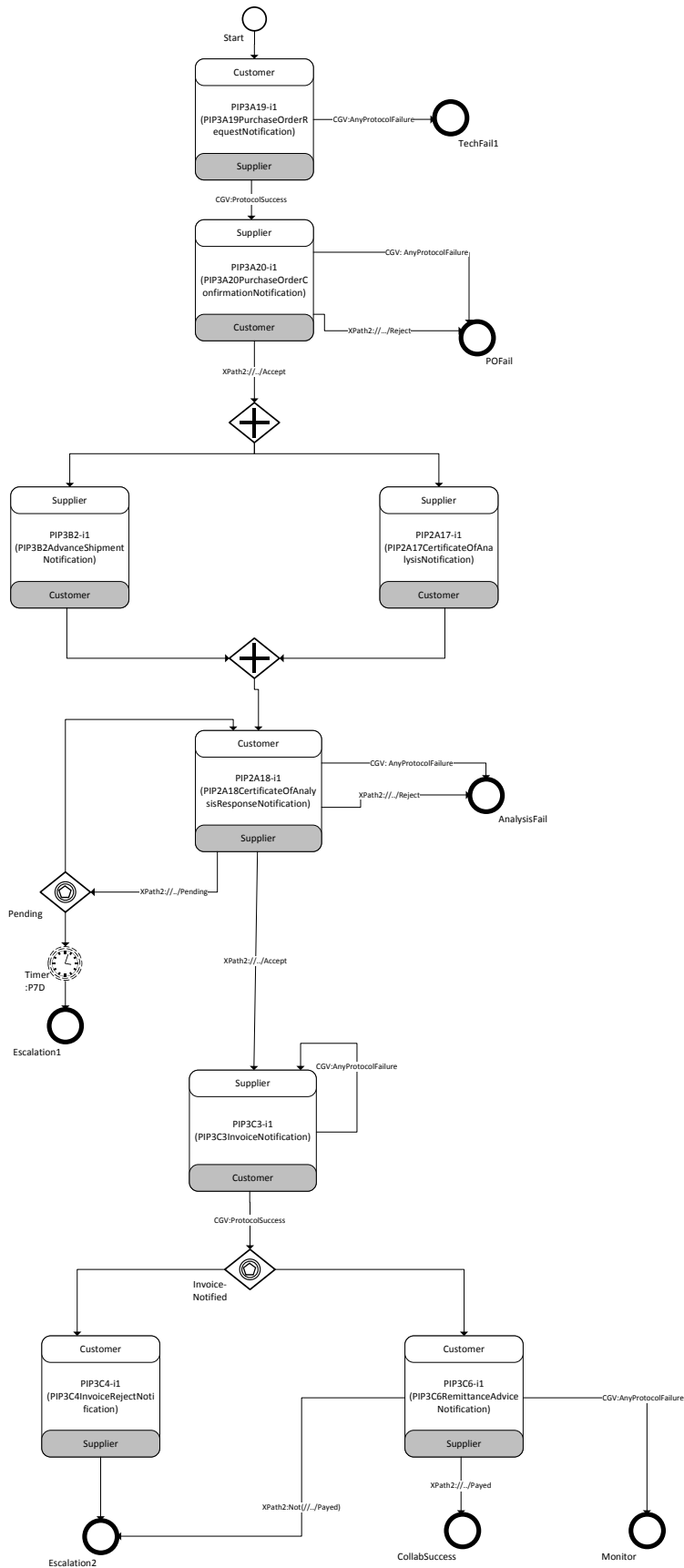
8.9 RosettaNet RIG 2A17 - Executable Style

(for reference see: RosettaNet RIG 2A17 V11.00.00, RIG_2A17_Final.doc, section 2.3.1)

Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and sheet "2-ClassicOrderToCash-Executable" for basic documentation of the characteristics of "executable" choreographies. Further, see sheet "4-ExtendedOrderToCash-Executable" for documentation on event-based choice gateways and sheet "7-OrderToCashLSP-Expanded-SeqMP" for documentation on timers.

The new concept introduced in this sample is parallel composition where BPMN 2.0 parallel gateways (diamonds with a cross) are used to delineate the concurrent activities.

Note that (if an executable choreography is to be designed) the concurrent branches of a parallel composition may not overlap, i.e., no control dependencies between the branches are allowed for. Further, each branch of a parallel structure that splits up alternative paths of control flow (using event-based choices or decisions) must join these paths before reaching the join gateway of the parallel structure. Finally, termination of a branch within a parallel structure is disallowed.



8.10 RosettaNet RIG 3A1 - Collapsed Sequential MultiParty Style

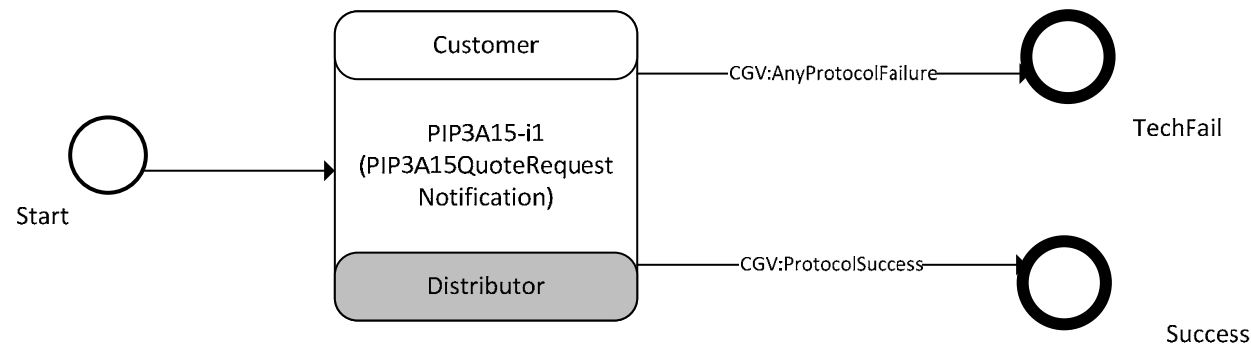
See for reference: RosettaNet RIG 3A1 V02.00.00, RIG 3A1 V02.00.00.doc (page 4)

also specified in: RosettaNet RIG 5D1 V01.00.00, RIG 5D1_V01.00.00.doc (page 4)

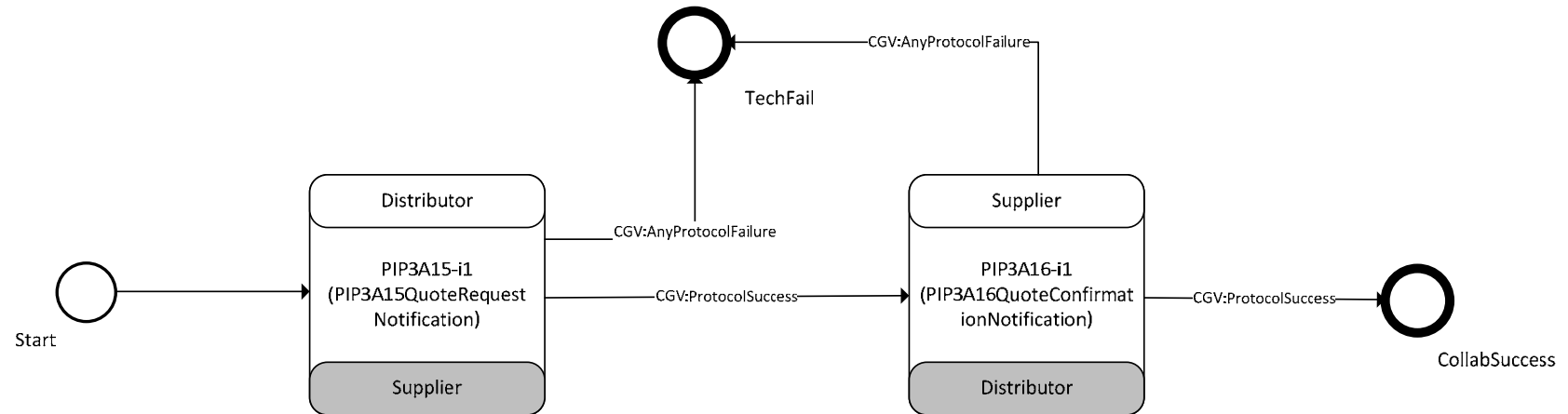
This is just an additional sample that demonstrates the use of SeqMP choreographies.

Please see sheet "1-ClassicOrderToCash-Cartography" for documentation of PIP and role representation and sheet "2-ClassicOrderToCash-Executable" for basic documentation of the characteristics of "executable" choreographies. Further, see sheet "4-ExtendedOrderToCash-Executable" for documentation on event-based choice gateways and sheet "7-OrderToCashLSP-Expanded-SeqMP" for documentation on timers.

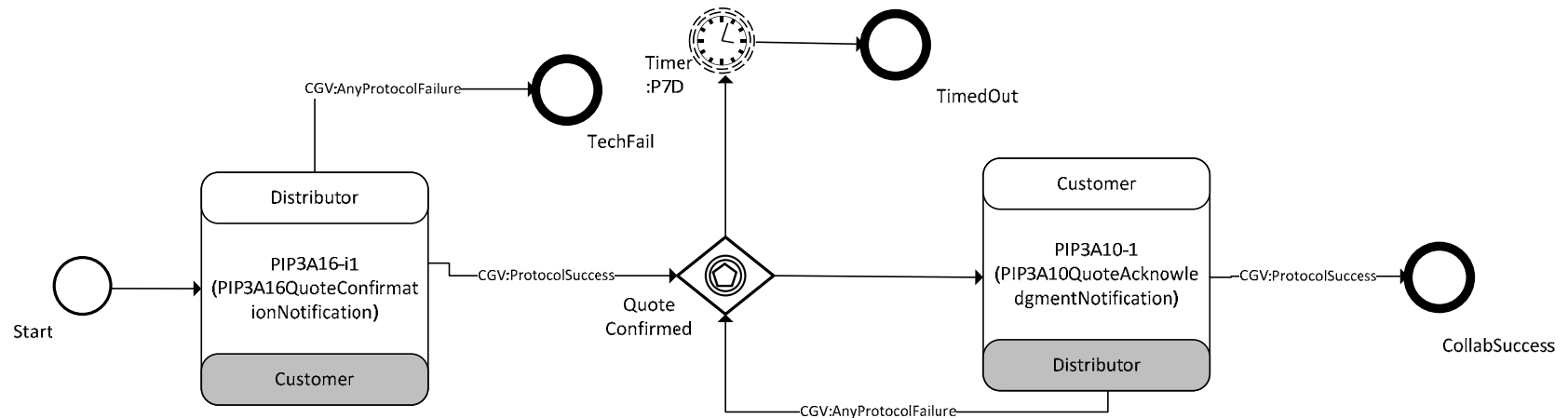
8.10.1 Request-Quote-Customer



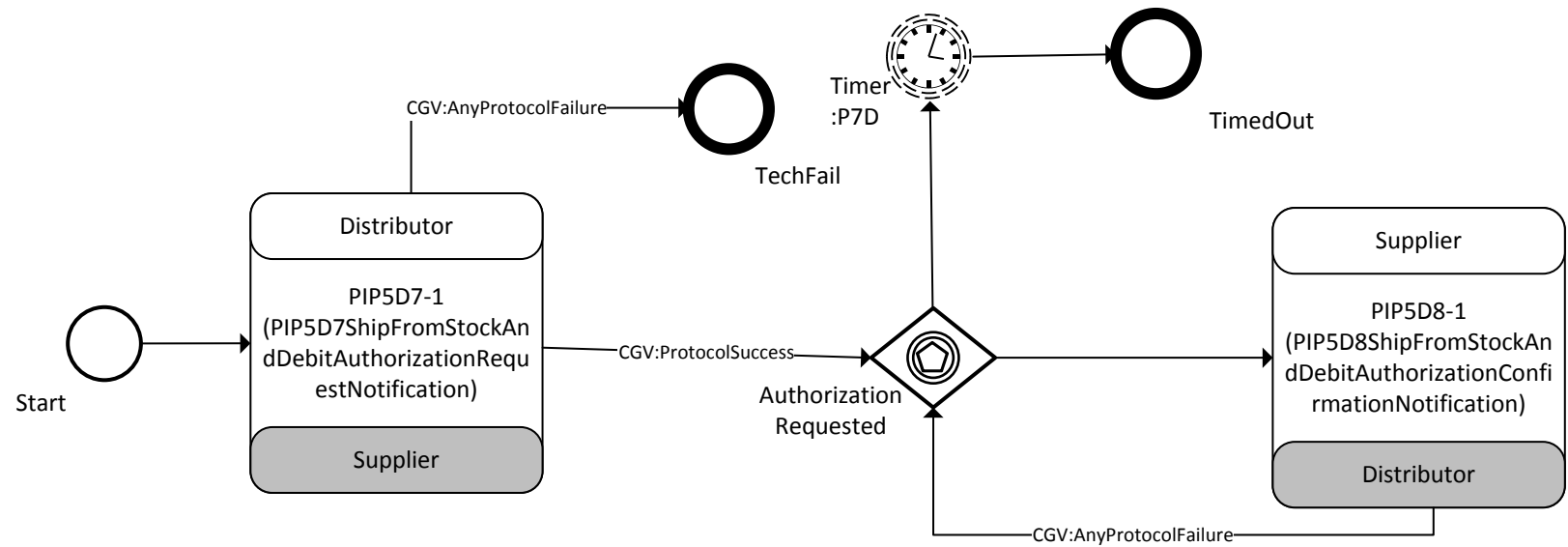
8.10.2 Quote-Exchange



8.10.3 QuoteAndAcknowledge



8.10.4 SFStockAndDebit



8.10.5 SeqMP-MultiPartyCollaboration

