Contract Proxy HTTP/TLS/JSON

System Description

Abstract

This document describes a system useful for managing digital legal contracts, such as for deliveries, payments or licensing, between distinct stakeholders represented by their own local clouds. In particular, the system takes responsibility for digital signing and publishing contractual events via the *Event Publish* service. The tasks of responding to and initiating contract offers and acceptances are delegated to so-called *agent systems*.

ARTEMIS Innovation Pilot Project: Arrowhead
THEME [SP1-JTI-ARTEMIS-2012-AIPP4 SP1-JTI-ARTEMIS-2012-AIPP6]
[Production and Energy System Automation Intelligent-Built environment and urban infrastructure for sustainable and friendly cities]

Contents

1 Overview

This document describes the HTTP/TLS/JSON Contract Proxy (CoPx) Eclipse Arrowhead system, which exsits to help enable distinct parties to digitalize their contractual interactions. Examples of such interactions could be taking on the obligation to deliver a good, negotiating insurance, agreeing to pay by invoice in exchange for access to a particular Eclipse Arrowhead service, among many other. Concretely, the CoPx system acts as a proxy, or representative, for systems from a particular Arrowhead local cloud, as exemplified in Figure 1.

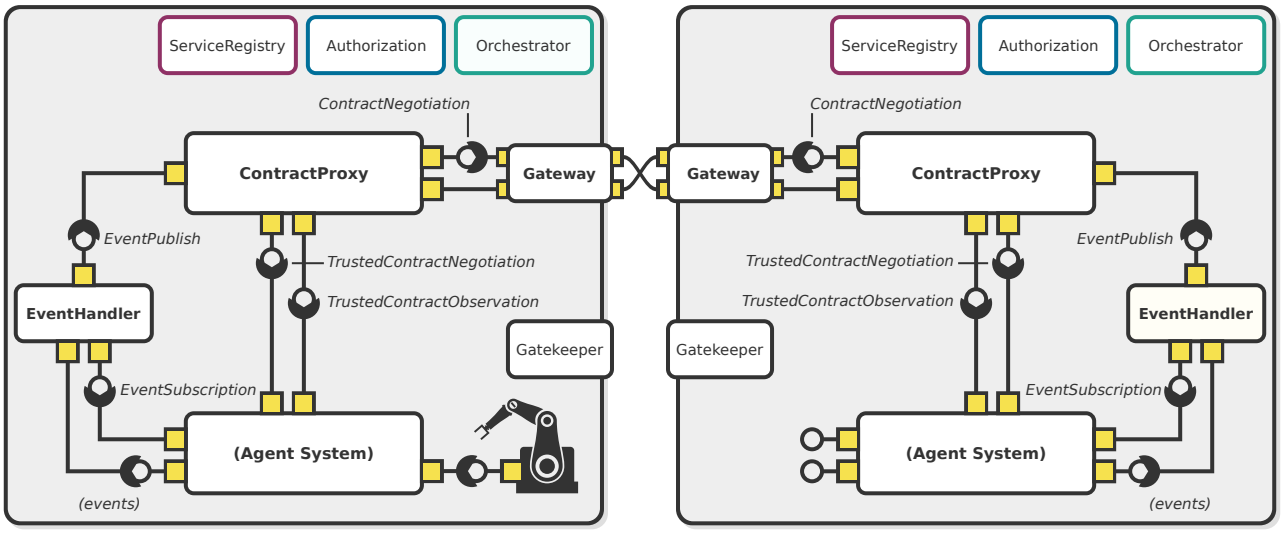

Figure 1: An example of two Arrowhead local clouds, both of which contain an agent system that uses a CoPx system to relay its contract offers, acceptances and rejections. From the perspective of each agent system, the party interacted with is the CoPx system in the other cloud, as it is what signs each interaction started from within its cloud.

Application systems in a local cloud that use a particular CoPx system are referred to as its *agents*. The CoPx system will, in turn, communicate with other systems, referred to as *counter-parties* that implement the Contract Negotiation (CoNe) service. In Figure 1, the counter-party of the CoPx system in each cloud is another CoPx system. Any system providing and consuming the CoNe service could, however, be a counter-party.

Concretely, the CoPx system takes contract messages, both from its agents and counter-parties, and does two things with those messages. Firstly, it simplifies management of cryptographic legal idenitities for its agents by adding and removing signatures and certificate identifiers from the messages it receives. Secondly, it publishes information, via the *Event Publish* service, about those messages within its local cloud.

The rest of this document is organized as follows. In the remainder of this section we comment on the status of this document. In Section 2, we outline major delimitations of the system, which is a work-in-progress. In Section 3, we describe how the CoPx system simplifies management of cryptographic legal identities and how it publishes contract events. Finally, In Section 4, we describe what services the CoPx system consumes and produces. *Readers of this document are assumed to be familiar with the CoNe service. For more information about that service, please refer to the CoNe service description.*

1.1 Status of this Document

Designing an adequate infrastructure for digitized collaboration has proven to require a lot more thought and effort than simply ”throwing a blockchain” on the problem. The criticality and genuine complexity of the problem being addressed by this system means that there still are too many unknowns and unsures for this document to be close to finalization. This document, an all other such part of the same Eclipse Arrowhead Core proposal, are still to be considered early drafts and might to have to undergo several significant revisions before becoming sufficient for most kinds of industrial deployments. If evaluating this system, and any implementations that may come with it, please report back your findings and experiences to Emanuel Palm <emanuel.palm@ltu.se>, who will ensure they contribute to the further refinement of the system.

2 Important Delimitations

The primary purpose of the CoPx system is to help manage the complexity inherent to contractual collaboration between distinct stakeholders. As this system and the Contract Negotiation service represent rather new concepts, there is not yet any consensus regarding what kinds of abstractions are appropriate for reaching this aim. Consequently, the current state of this document represent what could be thought of as an ambition for a minimum viable product. In particular, no way is provided for managing contract templates, switching between legal identities, or deciding what external legal identities are trusted. If the core assumptions surrounding the system prove to be sufficient, future work will likely be spent to extend the system in various ways, both by creating complementary systems and by adding more functionality.

In particular, if the current roles of the CoPx system are decided to be adequate, then it might become relevant to make the system able to manage multiple legal identities, perhaps by mapping certain agents to certain certificates.

3 System Role

As already mentioned in Section 1, the CoPx system fulfills two primary roles. Firstly, it manages cryptographic legal identities, and, secondly, it publishes information about contract events within its local cloud. How these two functions are performed is exemplified in Figure 2 as six steps.

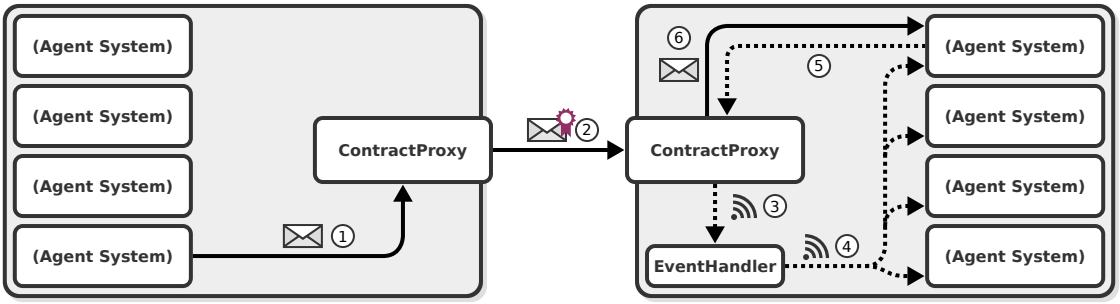

Figure 2: Two Arrowhead local clouds, described naively using only agents and CoPx systems. The figure is meant to illustrate the functions the CoPx system can perform. Concretely, it shows how contract messages, such as contract offers and acceptances, are relayed via CoPx systems. The envolopes in steps 1, 2 and 6 symbolize contract messages, out of which only the envelop in step 2 is cryptographically secured via signatures and certificate identifiers. Steps 3 and 4 represent a publish request and an actual announcement, respectively. Step 5 denotes how a particular agent system makes a request for the contract message announced to exist in step 4. While omitted for the sake of brevity, the leftmost local cloud would in a real-world setting also publish the fact that one agent system sent a message in step 1.

The steps proceed as follows:

1. Some agent system in the left hand cloud decides to send an offer to buy component in exchange for some payment by invoice to the cloud on the right hand side. Concretely, it invokes the *Offer* function of the Trusted Contract Negotiation service produced by the CoPx system in its own cloud. Notably, the sent offer contains one internal identifier that only is valid within the left hand cloud, used to identify the CoPx system to receive the offer in the right hand cloud. When referring to x.509 certificates, the only kind currently supported, internal identifiers are always the Common Name of the certificate owner. Furthermore, the offer is not signed.
2. After having received the offer, the CoPx system in the left hand cloud substitutes the internal identifier for the hash of the certificate of the CoPx system in the right hand cloud, as well as adding its own certificate identifier and signature to the offer. The offer is then forwarded to the CoPx system in the right hand cloud by invoking the *Offer* function of the Contract Negotiation service it produces.
3. With the offer now received at the right hand CoPx system, its existance is announced by sending a publish request via the Event Publish service of some *Event Handler* system within the same cloud.

4. Having received the publish request, the *Event Handler* relays it to all systems subscribing to it, which in this case happens to be all agent systems in the right hand local cloud. Note that the event does not contain the actual offer, only the details outlined in Section 3.1.

5. At this point, one agent system decides that the received event represent something of interest, for which reason it calls the *GetNegotiationByNamesAndId* function of the Trusted Contract Observation service to request a copy of the offer.

6. The CoPx system in the right hand cloud then sends a copy of the session containing the offer to the requesting agent system. Before being sent, however, the offer is stripped of its signature, the hash of the certificate of the CoPx that received the offer, as well as having the hash of the certificate of the CoPx system in the left hand cloud substituted for an internal identifier. The agent system in question can now consider whether or not it wishes to accept the offer, reject the offer, or make a counter-offer, which would be sent back in the same manner as this offer was received.

3.1 Management of a Cryptographic Legal Identity

As already established, the CoPx system abstracts away details regarding signatures and certificates[1] from its agents. More precisely, the CoPx system accepts messages from its agents via the Trusted Contract Negotiation service, adds signatures and certificate identifiers, and then relays those messages to the Contract Negotiation service of the intended counter-party receiver. Conversely, the CoPx system takes messages from its counter-parties via the Contract Negotiation service, removes signatures and substitutes the certificate identifiers with one internal name, saves a copy of both the original and modified message, and then announces the existence of the new message via the Event Publish service, as described in more detail in Section 3.2.

There are three primary benefits to abstracting away signatures and certificate information from agent systems in this way, as follows:

1. If agent systems do not know what concrete certificate is used to sign their messages, that certificate can be updated as it expires or is superceded without those agent systems having to know about it.

2. Multiple agent systems can, unknowingly, securely share the same identity, which could be important for conformance to relevant laws, which might require that a single legal entity be represented by a single certificate, or certain prior agreements.

3. The agent systems do not need to be aware about the certificates of counter-parties changing over time, as the agents do not refer to counter-parties by their certificate identifiers, but by internal names.

[1] For more information about x.509 certificates in particular, which may or may not be the most suitable kind of certificate for legal signing, please consult [1].

3.2 Publishing of Contract Events

As also already illustrated in the beginning of this section, the CoPx system announces contract events to subscribers within its local cloud. It should be noted that those announcements do not contain the concrete messages received by the CoPx systems. Rather, the event messages managed by the Event Publish contain three particularly significant fields, which are

1. a UTF-8 string event identifier,
2. a collection of metadata and
3. a UTF-8 string payload.

Concretely, those fields are populated as follows:

Field	Description
Event Identifier	Must be set to "`ContractNegotiation.Update`".
Metadata	Must contain the "`offeror`", "`receiver`", "`templates`" and "`status`" fields. "`offeror`" and "`receiver`" contain the internal identifiers that identify the offering and receiving parties, respectively. "`templates`" contains a comma-separated list of internal names of the contract templates invoked by the message. Lastly, the "`status`" value must be one of "`OFFERING`", "`ACCEPTED`" or "`REJECTED`".
Payload	The identifier of the negotiation session containing the message.

When a subscriber is notified about the existence of some message of interest, it uses the Trusted Contract Observation service of the CoPx system to retrieve an actual copy of that message.

4 Services

The CoPx systems produces and consumes the following services, as described in Figure 3.

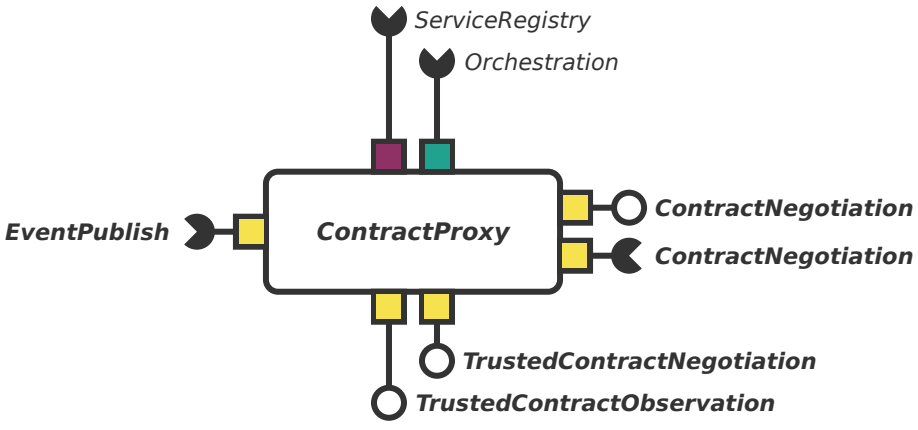

Figure 3: The Arrowhead Contract Proxy system. Note how the Contract Negotiation service is both produced and consumed. This is what enables the system to both send and receive contracts offers and acceptances. When offers and acceptances are received, they are announced via the Event Publish service and made available via the Trusted Contract Observation service. When trusted systems want to make or accept offers, they are to use the produced Trusted Contract Negotiation service.

More details regarding the consumed and produced services are given in the following subsection.

4.1 Consumed Services

4.1.1 Contract Negotiation

This service is consumed to relay messages originally received via the Trusted Contract Negotiation service. As the original message can identify one out of any number of receivers, the CoPx system may have to be able to consume this service from multiple distinct systems.

4.1.2 Event Publish

This service is used to announce the availability of new messages via the Trusted Contract Observation.

4.1.3 Service Registry

This service is consumed to make sure that the CoPx system becomes accessible to other systems.

4.1.4 Orchestration

This service is used to determine what Event Publish and Contract Negotiation services to utilitze.

4.2 Produced Services

4.2.1 Contract Negotiation

This service is produced to allow other systems to send contract offers, acceptances and rejections to the CoPx system, which will then announce their existence via the Event Publish service and make them available via the produced Trusted Contract Observation service.

4.2.2 Trusted Contract Negotiation

This service is produced to allow for authorized systems, likely located in the same Arrowhead local cloud, send contract offers, acceptances and rejections to counter-party systems, known to the CoPx system, that also produce the Contract Negotiation service.

4.2.3 Trusted Contract Observation

This service is to make contract negotiation session information available to authorized systems.

5 References

[1] D. Cooper *et al.*, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, 2008. [Online]. Available: https://doi.org/10.17487/RFC5280

6 Revision History

6.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1				

6.2 Quality Assurance

No.	Date	Version	Approved by
1			