Stream: Independent Submission
RFC: 8921
Category: Informational
Published: October 2020
ISSN: 2070-1721
Authors: M. Boucadair, Ed. *Orange*; C. Jacquenet *Orange*; D. Zhang *Huawei Technologies*; P. Georgatsos *CERTH*

RFC 8921
Dynamic Service Negotiation: The Connectivity Provisioning Negotiation Protocol (CPNP)

Abstract

This document defines the Connectivity Provisioning Negotiation Protocol (CPNP), which is designed to facilitate the dynamic negotiation of service parameters.

CPNP is a generic protocol that can be used for various negotiation purposes that include (but are not necessarily limited to) connectivity provisioning services, storage facilities, Content Delivery Networks, etc.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8921.

Copyright Notice

Table of Contents

1. Introduction

This document defines the Connectivity Provisioning Negotiation Protocol (CPNP) that is meant to dynamically exchange and negotiate connectivity provisioning parameters and other service-specific parameters between a Customer and a Provider. CPNP is a tool that introduces automation to the service negotiation and activation procedures, thus fostering the overall service provisioning process. CPNP can be seen as a component of the dynamic negotiation metadomain described in Section 2.4 of [RFC7149].

CPNP is a generic protocol that can be used for negotiation purposes other than connectivity provisioning. For example, CPNP can be used to request extra storage resources, to extend the footprint of a Content Delivery Network (CDN), to enable additional features from a cloud Provider, etc. CPNP can be extended with new Information Elements (IEs). Sample negotiation use cases are described in Section 5. Section 4 introduces several order processing models and defines those that are targeted by CPNP. The CPNP negotiation model is then detailed in Section 7.

[RFC7297] describes a Connectivity Provisioning Profile (CPP) template to capture connectivity requirements to be met by a transport infrastructure for the delivery of various services such as Voice over IP (VoIP), IPTV, and Virtual Private Network (VPN) services [RFC4026]. The CPP document defines the set of IP transfer parameters that reflect the guarantees that can be provided by the underlying transport network together with reachability scope and capacity needs. CPNP uses the CPP template to encode connectivity provisioning clauses that are subject to negotiation. The accepted CPP will then be passed to other functional elements that are responsible for the actual service activation and provisioning. For example, Network Configuration Protocol (NETCONF) [RFC6241] or RESTCONF [RFC8040] can be used to activate

adequate network features that are required to deliver the accepted service. How the outcome of CPNP negotiation is translated into service and network provisioning actions is out of scope of this document.

As a reminder, several proposals have been made in the past by the (research) community (e.g., Common Open Policy Service protocol for supporting Service Level Specification [COPS-SLS], Service Negotiation Protocol [SrNP], Dynamic Service Negotiation Protocol [DSNP], Resource Negotiation and Pricing Protocol [RNAP], Service Negotiation and Acquisition Protocol [SNAP]). CPNP leverages the authors' experience with SrNP by separating the negotiation primitives from the service under negotiation. Moreover, careful examination of the other proposals revealed certain deficiencies that were easier to address through the creation of a new protocol rather than the modification of existing protocols. For example:

- COPS-SLS relies upon the COPS usage for policy provisioning (COPS-PR) [RFC3084], which is a Historic RFC.
- DSNP is tightly designed with one specific service in mind (QoS) and does not make any distinction between a quotation phase and the actual service-ordering phase.

One of the primary motivations of this document is to provide a permanent reference to exemplify how service negotiation can be automated.

Implementation details are out of scope. An example of required modules and interfaces to implement this specification is sketched in Section 4 of [AGAVE]. This specification builds on that effort.

2. Terminology

This document makes use of the following terms:

Customer: Is a business role that denotes an entity that is involved in the definition and the possible negotiation of an order, including a Connectivity Provisioning Agreement, with a Provider. A connectivity provisioning document is captured in a dedicated CPP template-based document, which may specify (among other information) the sites to be connected, border nodes, outsourced operations (e.g., routing, traffic steering).

The right to invoke the subscribed service may be delegated by the Customer to third-party end users or brokering services.

A Customer can be a Service Provider, an application owner, an enterprise, a user, etc.

Network Provider (or Provider): Owns and administers one or many transport domain(s) (typically Autonomous Systems (ASes)) composed of (IP) switching and transmission resources (e.g., routing, switching, forwarding, etc.). Network Providers are responsible for delivering and operating connectivity services (e.g., offering global or restricted reachability at specific rates). Offered connectivity services may not necessarily be restricted to IP.

The policies to be enforced by the connectivity service delivery components can be derived from the technology-specific clauses that might be included in agreements with the Customers. If no such clauses are included in the agreement, the mapping between the connectivity requirements and the underlying technology-specific policies to be enforced is deployment specific.

Quotation Order: Denotes a request made by the Customer to the Provider that includes a set of requirements. The Customer may express its service-specific requirements by assigning (strictly or loosely defined) values to the information items included in the commonly understood template (e.g., CPP template) describing the offered service. These requirements constitute the parameters to be mutually agreed upon.

Offer: Refers to a response made by the Provider to a Customer's quotation order that describes the ability of the Provider to satisfy the order at the time of its receipt. Offers reflect the capability of the Provider in accommodating received Customer orders beyond monolithic 'yes/no' answers.

An offer may fully or partially meet the requirements of the corresponding order. In the latter case, it may include alternative suggestions that the Customer may take into account by issuing a new order.

Agreement: Refers to an order placed by the Customer and accepted by the Provider. It signals the successful conclusion of a negotiation cycle.

3. CPNP Functional Elements

The following functional elements are defined:

CPNP client (or client): Denotes a software instance that sends CPNP requests and receives CPNP responses. The current operations that can be performed by a CPNP client are listed below:

1. Create a quotation order (Section 9.2.1).
2. Cancel an ongoing quotation order under negotiation (Section 9.2.7).
3. Accept an offer made by a server (Section 9.2.4).
4. Withdraw an agreement (Section 9.2.8).
5. Update an agreement (Section 9.2.9).

CPNP server (or server): Denotes a software instance that receives CPNP requests and sends back CPNP responses accordingly. The CPNP server is responsible for the following operations:

1. Process a quotation order (Section 9.2.2).
2. Make an offer (Section 9.2.3).
3. Cancel an ongoing quotation order (Section 11.2.3).

4. Process an order withdrawal (Section 11.2.3).

4. Order Processing Models

For preparing their service orders, Customers may need to be aware of the offered services. Therefore, Providers should first proceed with the announcement (or the exposure) of the services they can provide. The service announcement process may take place at designated global or Provider-specific service markets or through explicit interactions with the Providers. The details of this process are outside the scope of this document.

With or without such service announcement/exposure mechanisms in place, the following order processing models can be distinguished:

Frozen model:
: The Customer cannot actually negotiate the parameters of the service(s) offered by a Provider. After consulting the Provider's service portfolio, the Customer selects the service offer to which he or she wants to subscribe and places an order to the Provider. Order handling is quite simple on the Provider side because the service is not customized per Customer's requirements, but rather designed to address a Customer base that shares the same requirements (i.e., these Customers share the same Connectivity Provisioning Profile). This mode can be implemented using existing tools such as [RFC8309].

Negotiation-based model:
: Unlike the frozen model, the Customer documents his/her requirements in a request for a quotation, which is then sent to one or several Providers. Solicited Providers check whether they can address these requirements or not, and get back to the Customer accordingly, possibly with an offer that may not exactly match the Customer's requirements (e.g., a 100 Mbps connection cannot be provisioned given the amount of available resources, but an 80 Mbps connection can be provided). A negotiation between the Customer and the Provider(s) then follows until both parties reach an agreement (or do not).

Both frozen and negotiation-based models require the existence of appropriate service templates like a CPP template and their instantiation for expressing specific offerings from Providers and service requirements from Customers, respectively. CPNP can be used in either model for automating the required Customer-Provider interactions. The frozen model can be seen as a special case of the negotiation-based model. This document focuses on the negotiation-based model. Not only 'yes/no' answers but also counterproposals may be offered by the Provider in response to Customer orders.

Order processing management on the Network Provider's side usually solicits features supported by the following functional blocks:

- Network provisioning (including order activation, Network Planning, etc.)
- Authentication, authorization, and accounting (AAA)
- Network and service management (performance measurement and assessment, fault detection, etc.)

- Sales-related functional blocks (e.g., billing, invoice validation)
- Network impact analysis

CPNP does not assume any specific knowledge about these functional blocks, drawing an explicit line between protocol operation and the logic for handling connectivity provisioning requests. An order processing logic is typically fed with the information manipulated by the aforementioned functional blocks. For example, the resources that can be allocated to accommodate the Customer's requirements may depend on network availability estimates as calculated by the planning functions and related policies, as well as the number of orders to be processed simultaneously over a given period of time.

This document does not elaborate on how Customers are identified and subsequently managed by the Provider's information system.

5. Sample Use Cases

A non-exhaustive list of CPNP use cases is provided below:

1. [RFC4176] introduces the Layer 3 VPN (L3VPN) Service Order Management functional block, which is responsible for managing the requests initiated by the Customers and tracks the status of the completion of the related operations. CPNP can be used between the Customer and the Provider to negotiate L3VPN service parameters.

   A CPNP server could therefore be part of the L3VPN Service Order Management functional block discussed in [RFC4176]. A L3VPN Service YANG data model (L3SM) is defined in [RFC8299]. Once an agreement is reached, the service can be provisioned using, e.g., the L3VPN Network YANG data model specified in [L3VPN-NETWORK-YANG].

   Likewise, a CPNP server could be part of the Layer 2 VPN (L2VPN) Service Order Management functional block. A YANG data model for L2VPN service delivery is defined in [RFC8466]. Once an agreement is reached, the L2VPN service can be provisioned using, e.g., the L2VPN Network YANG data model specified in [L2VPN-NETWORK-YANG].

2. CPNP can be used between two adjacent domains to deliver IP interconnection services (e.g., enable, update, disconnect). For example, two Autonomous Systems (ASes) can be connected via several interconnection points. CPNP can be used between these ASes to upgrade existing links, request additional resources, provision a new interconnection point, etc.

   See, for example, the framework documented in [ETICS].

3. An integrated Provider can use CPNP to rationalize connectivity provisioning needs related to its service portfolio. A CPNP server function is used by network operations teams. A CPNP interface to trigger CPNP negotiation cycles is exposed to service management teams.
4. Service Providers can use CPNP to initiate connectivity provisioning requests towards a number of Network Providers so as to optimize the cost of delivering their services. Although multiple CPNP ordering cycles can be initiated by a Service Provider towards multiple Network Providers, a subset of these orders may actually be put into effect.

For example, a cloud Service Provider can use CPNP to request more resources from Network Providers.

5. CPNP can also be used in the context of network slicing [NETSLICES-ARCH] to request network resources together with a set of requirements that need to be satisfied by the Provider. Such requirements are not restricted to basic IP forwarding capabilities, but may also include a characterization of a set of service functions that may be invoked. For the network slicing case, the instances of a CPP template could be derived from the network slice template documented in [TEAS-SLICE-NBI].
6. CPNP can be used in Machine-to-Machine (M2M) environments to dynamically subscribe to M2M services (e.g., access data retrieved by a set of sensors, extend sensor coverage, etc.).

   Also, Internet of Things (IoT) [RFC6574] domains may rely on CPNP to enable dynamic access to data produced by involved objects, according to their specific policies, to various external stakeholders such as data analytics and business intelligence companies. Direct CPNP-based interactions between IoT domains and interested parties enable open access to diverse sets of data across the Internet, e.g., from multiple types of sensors, user groups, and/or geographical areas.

7. CPNP can be used in the context of Interface to Network Security Functions (I2NSF) [RFC8329] to capture the Customer-driven policies to be enforced by a set of Network Security Functions.
8. A Provider offering cloud services can expose a CPNP interface to allow Customers to dynamically negotiate typical data center resources, such as additional storage, processing and networking resources, enhanced security filters, etc.

   Cloud computing Providers typically structure their computation service offerings by bundling CPU, RAM, and storage units as quotas, instances, or flavors that can be consumed in an ephemeral or temporal fashion during the lifetime of the required function. A similar approach is followed by CPNP (see for example, Section 9.2.11).

9. In the inter-cloud context (also called cloud of clouds or cloud federation), CPNP can be used to reserve computing and networking resources hosted by various cloud infrastructures.
10. CDN Providers can use CPNP to extend their footprint by interconnecting their respective CDN infrastructures [RFC6770] (see Figure 1).

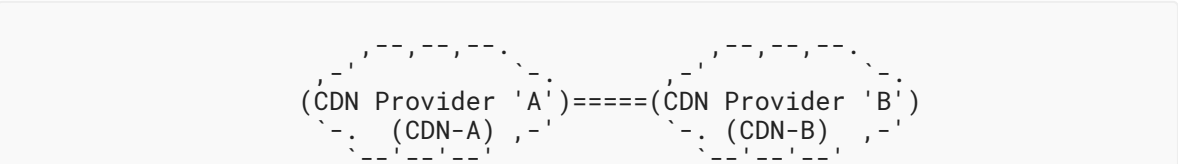

Figure 1: CDN Interconnection

11. Mapping Service Providers (MSPs) [RFC7215] can use CPNP to enrich their mapping database by interconnecting their mapping system (see Figure 2). This interconnection allows the relaxation of the constraints on PxTR (Proxy Ingress/Egress Tunnel Router) in favour of

native LISP (Locator/ID Separation Protocol) forwarding [RFC6830]. Also, it prevents the fragmentation of the LISP mapping database. A framework is described in [LISP-MS-DISCOVERY].

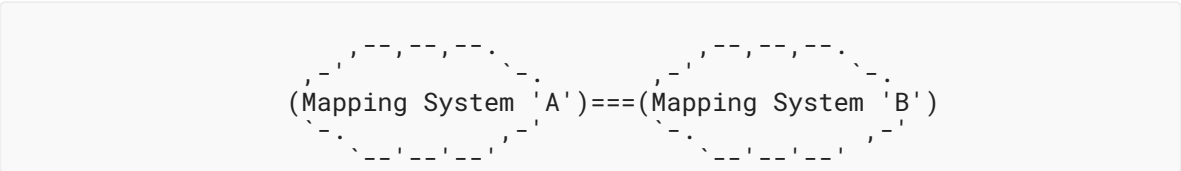

Figure 2: LISP Mapping System Interconnect

12. CPNP may also be used between SDN (Software-Defined Networking) controllers in contexts where Cooperating Layered Architecture for Software-Defined Networking (CLAS) is enabled [RFC8597].

6. CPNP Deployment Models

Several CPNP deployment models can be envisaged. Two examples are listed below:

- The Customer deploys a CPNP client while one or several CPNP servers are deployed by the Provider. A CPNP client can discover its CPNP servers using a variety of means (static, dynamic, etc.).
- The Customer does not enable any CPNP client. The Provider maintains a Customer Order Management portal. The Customer can initiate connectivity provisioning quotation orders via the portal; appropriate CPNP messages are then generated and sent to the relevant CPNP server. In this model, both the CPNP client and CPNP server are under the responsibility of the same administrative entity (i.e., Network Provider).

Once the negotiation of connectivity provisioning parameters is successfully concluded, that is, an order has been placed by the Customer, the actual network provisioning operations are initiated. The specification of related dynamic resource allocation and policy enforcement schemes, as well as how CPNP servers interact with the network provisioning functional blocks on the Provider side, are out of the scope of this document.

This document does not make any assumptions about the CPNP deployment model either.

7. CPNP Negotiation Model

CPNP runs between a Customer and a Provider, carrying service orders from the Customer and corresponding responses from the Provider in order to reach a service provisioning agreement. As the services offered by the Provider are well described, by means of the CPP template for connectivity matters, the negotiation process is essentially a value-settlement process, where an agreement is pursued on the values of the commonly understood information items (service parameters) included in the service description template (Section 9.1.9).

The content that CPNP carries and the negotiation logic invoked at Customer and Provider sides to manipulate the content (i.e., the information carried in CPNP messages to proceed with the negotiation) is transparent to the protocol.

The protocol aims to facilitate the execution of the negotiation logic by providing the required generic communication primitives.

Since negotiations are initiated and primarily driven by the Customer's negotiation logic, it is reasonable to assume that the Customer is the only party that can call for an agreement. An implicit approach is adopted for not overloading the protocol with additional messages. In particular, the acceptance of an offer made by the Provider signals a call for agreement from the Customer. Note that it is almost certain the Provider will accept this call since it refers to an offer that the Provider made. Of course, at any point the Provider or the Customer may quit the negotiations, each on its own grounds.

Based on the above, CPNP adopts a quotation order/offer/answer model, which proceeds through the following basic steps (Figure 3):

1. The CPNP client specifies its service requirements in a Provisioning Quotation Order (PQO). The order may include strictly or loosely defined values in the clauses describing service provisioning characteristics.
2. The CPNP server declines the PQO, or makes an offer to address the requirements of the PQO, or suggests a counterproposal that partially addresses the requirements of the PQO in case specific requirements cannot be accommodated.
3. The CPNP client either accepts or declines the offer. The acceptance of the offer by the CPNP client implies a call for agreement and, thus, the agreement between both parties and the conclusion of the negotiation.

Figure 3: Simplified Service Negotiation

Multiple instances of CPNP may run at a Customer's or a Provider's domains. A CPNP client may be engaged in multiple, simultaneous negotiations with the same or different CPNP servers (parallel negotiations, see Section 8.10), and a CPNP server may need to negotiate with other Provider(s) as part of negotiations that are ongoing with a CPNP client (cascaded negotiations, see Section 8.8).

CPNP relies on various timers to run its operations. Two types of timers are defined: those that are specific to CPNP message transmission and those that are specific to the negotiation logic. The latter are used to guide the negotiation logic at both CPNP client and CPNP server sides, particularly in cases where the CPNP client is involved in parallel negotiations with several CPNP servers or in cases where the CPNP server is, in turn, involved in negotiations with other

Providers for processing a given Customer-originated quotation order. CPNP allows a CPNP server to request extra time to proceed with the negotiation. This request may be accepted or rejected by the CPNP client.

Providers may need to publish available services to the Customers (see Section 4). CPNP may optionally support this functionality. Dedicated templates can be defined for the purpose of service announcement, which will be used by the CPNP clients to initiate their CPNP negotiation cycles.

For the sake of simplicity, a single offer/answer stage is assumed within one CPNP negotiation cycle. Nevertheless, as already stated, multiple CPNP negotiation cycles can be undertaken by a CPNP client (see Figure 4).

The model is flexible enough to accommodate changing conditions during the lifetime of a service (e.g., the introduction of an additional VPN site).

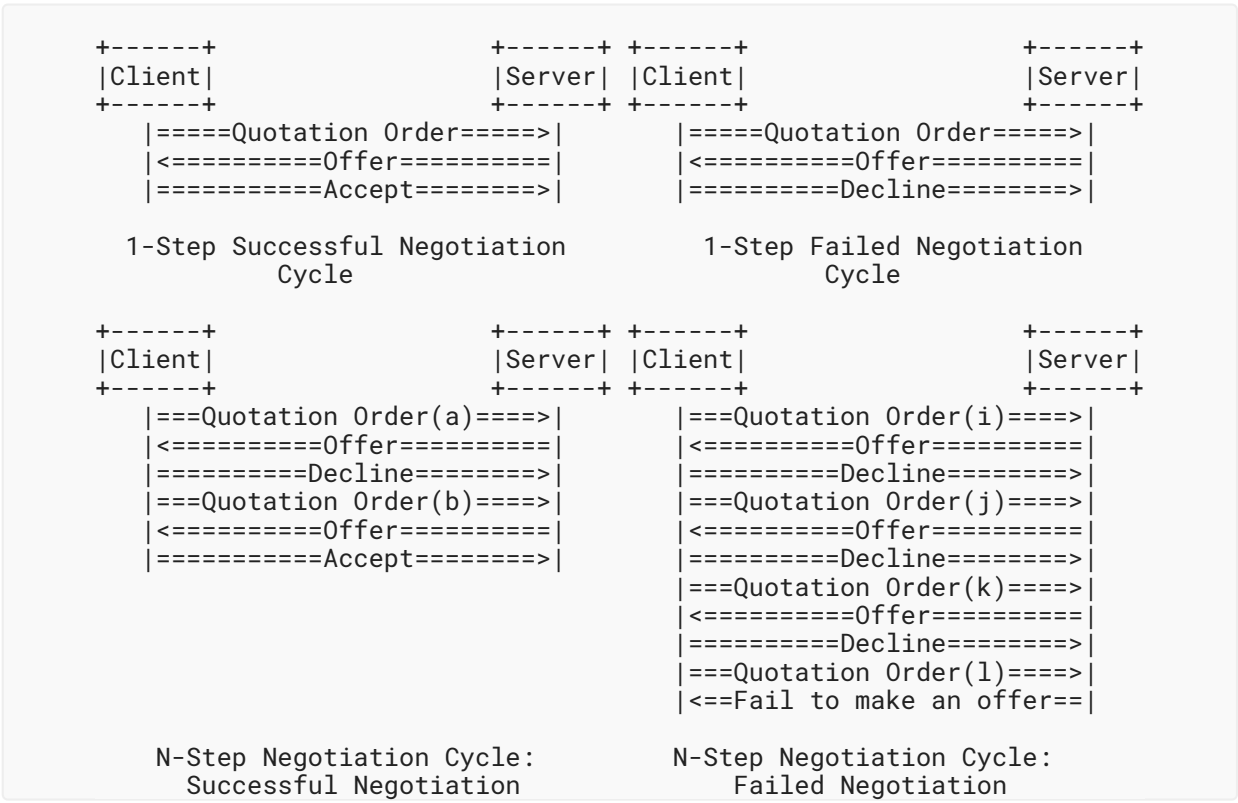

Figure 4: Overall Negotiation Process

The means used by a CPNP client to retrieve a list of active/accepted offers are not defined in this document.

An order can be implicitly or explicitly activated. Section 3.11 of [RFC7297] specifies a dedicated clause called Activation Means. Such a clause indicates the required action(s) to be undertaken to activate access to the (IP connectivity) service. This document defines a dedicated CPNP message that can be used for explicit activation (Section 9.2.11).

8. Protocol Overview

8.1. Client/Server Communication

CPNP is a client/server protocol that can run over any transport protocol. The default transport mode is UDP secured with Datagram Transport Layer Security (DTLS) [RFC6347]. No permanent CPNP transport session needs to be maintained between the client and the server.

The CPNP client can be configured with the CPNP server(s). Typically, the CPNP client is configured with an IP address together with a port number using manual or dynamic configuration means (e.g., DHCP). Alternatively, a Provider may advertise the port number (CPNP_PORT) it uses to bind the CPNP service using SRV [RFC2782].

The CPNP client may be provided with a domain name of the CPNP server for PKIX-based authentication purposes. CPNP servers should prefer the use of DNS-ID and SRV-ID over CN-ID identifier types in certificate requests (Section 2.3 of [RFC6125]). URI-IDs should not be used for CPNP server identity verification.

The client sends CPNP requests using CPNP_PORT as the destination port number. The same port number used as the source port number of a CPNP request sent to a CPNP server is used by the server to reply to that request.

CPNP is independent of the IP address family.

CPNP retransmission for unreliable transports is discussed in Section 11.4.

Considerations related to mutual authentication are discussed in Section 13.

8.2. Policy Configuration on the CPNP Server

As an input to its decision-making process, the CPNP server may be connected to various external modules such as Customer Profiles, Network Topology, Network Resource Management, Order Repositories, AAA, and Network Provisioning Manager (an example is shown in Figure 5).

These external modules provide inputs to the CPNP server so that it can do the following:

- Check whether a Customer is entitled to initiate a provisioning quotation request.
- Check whether a Customer is entitled to cancel an ongoing order.
- Check whether administrative data (e.g., billing-related information) have been verified before the processing of the request starts.
- Check whether network capacity is available or additional capacity is required.

- Receive guidelines from network design and sales blocks (e.g., pricing, network usage levels, thresholds associated with the number of CPP templates that can be processed over a given period of time as a function of the nature of the service to be delivered, etc.).
- Transfer completed orders to network provisioning blocks (referred to as "Network Provisioning Manager" in Figure 5). For example, the outcome of CPNP may be passed to modules such as Application-Based Network Operations (ABNO) [RFC7491] or network controllers. These controllers will use protocols such as NETCONF [RFC6241] to interact with the appropriate network nodes and functions for the sake of proper service activation and delivery.

The above list of CPNP server operations is not exhaustive.

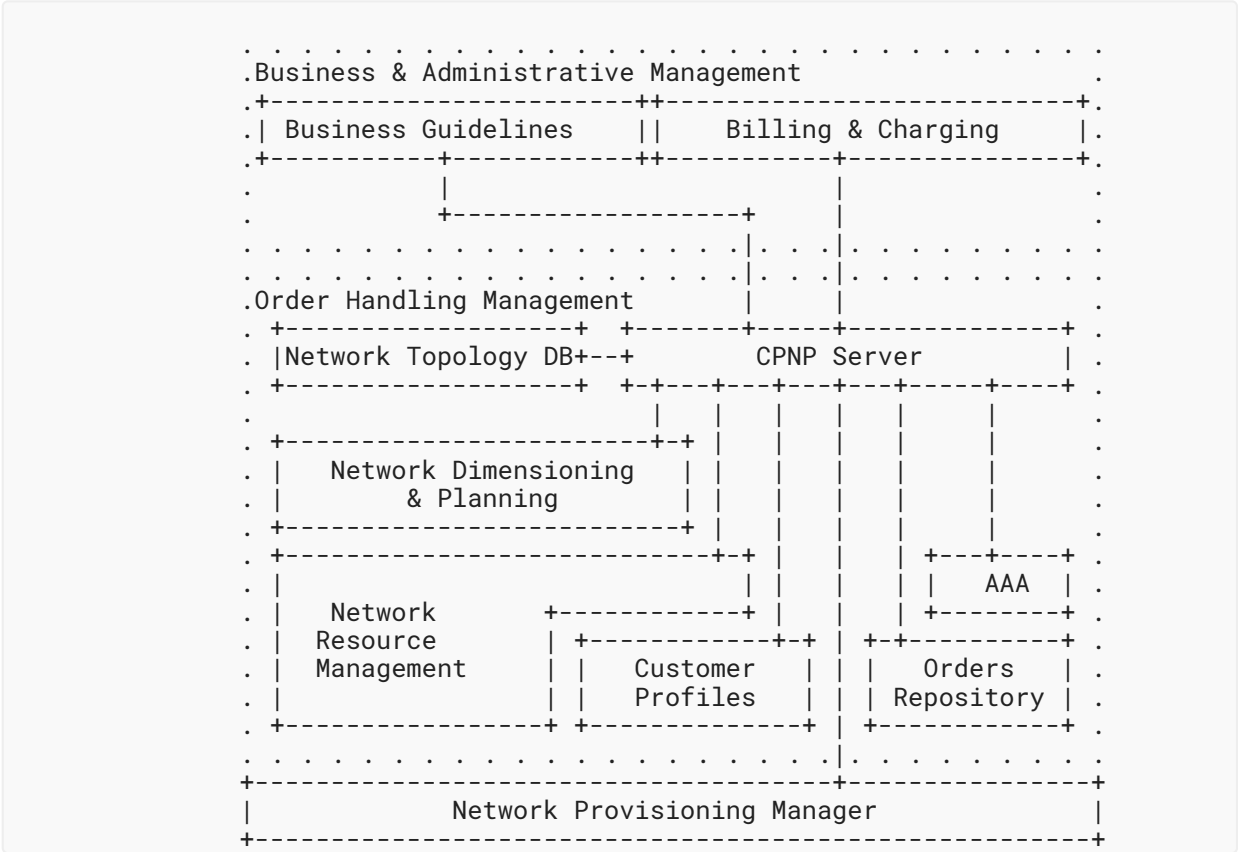

Figure 5: Order Handling Management Functional Block (Focus on Internal Interfaces)

The following order-handling modes can also be configured on the server:

Fully automated mode: This mode does not require any action from the administrator when receiving a request for a service. The server can execute its decision-making process related to the orders received and can generate corresponding offers.

Administrative validation checking: Some or all of the server's operations are subject to administrative validation procedures. This mode requires an action from the administrator for every request received. To that aim, the CPNP methods that can be automatically handled by the server (or are subject to one or several validation administrative checks) can be configured on the server.

8.3. CPNP Session Entries

A CPNP session entry is represented by a tuple defined as follows:

- Transport session (typically, the IP address of the CPNP client, the client's port number, the IP address of the CPNP server, and the CPNP server's port number).
- Incremented sequence number (Section 11.3).
- Customer agreement identifier: This is a unique identifier assigned to the order under negotiation by the CPNP client (Section 9.1.1). This identifier is also used by the client to identify the agreement that will result from a successful negotiation.
- Provider agreement identifier: This is a unique identifier assigned to the order under negotiation by the CPNP server (Section 9.1.2). This identifier is also used by the server to identify the agreement that will result from a successful negotiation.
- Transaction-ID (Section 8.4).

8.4. CPNP Transactions

A CPNP transaction occurs between a client and a server for completing, modifying, or withdrawing a service agreement, and comprises all CPNP messages exchanged between the client and the server, from the first request sent by the client to the final response sent by the server. A CPNP transaction is bound to a CPNP session (Section 8.3).

Because multiple CPNP transactions can be maintained by the CPNP client, the client must assign an identifier to uniquely identify a given transaction. This identifier is the Transaction-ID.

The Transaction-ID must be randomly assigned by the CPNP client, according to the best current practice for generating random numbers [RFC4086] that cannot be guessed easily. The Transaction-ID is used for validating CPNP responses received by the client.

In the context of a transaction, the client needs to select a sequence number randomly and then needs to assign it to the first CPNP message to send. This number is then incremented for each request message that is subsequently sent within the ongoing CPNP transaction (see Section 11.3).

8.5. CPNP Timers

CPNP adopts a simple retransmission procedure that relies on a retransmission timer represented by RETRANS_TIMER and a maximum retry threshold. The use of RETRANS_TIMER and a maximum retry threshold are described in Section 11.

The response timer (EXPECTED_RESPONSE_TIME) is set by the client to denote the time, in seconds, the client will wait to receive a response from the server to a PQO request (see Section 9.1.6). If the timer expires, the respective PQO is cancelled by the client, and a CANCEL message is generated accordingly.

The expected offer timer (EXPECTED_OFFER_TIME) is set by the server to indicate the time by when the CPNP server is expected to make an offer to the CPNP client (see Section 9.1.7). If no offer is received by then, the CPNP client will consider the order as rejected.

An offer expiration timer (VALIDITY_OFFER_TIME) is set by the server to represent the time, in minutes, after which an offer made by the server becomes invalid (see Section 9.1.8).

8.6. CPNP Operations

CPNP operations are listed below. They may be augmented depending on the nature of some transactions or because of security considerations that may necessitate a distinct CPNP client/server authentication phase before negotiation begins.

QUOTATION (Section 9.2.1):
: This operation is used by the client to initiate a PQO. Upon receipt of a QUOTATION request, the server may respond with a PROCESSING, OFFER, or a FAIL message. A QUOTATION-initiated transaction can be terminated by a FAIL message.

PROCESSING (Section 9.2.2):
: This operation is used to inform the remote party that its message (the order quotation or the offer) was received and it is being processed. This message can also be issued by the server to request more time, in which case, the client may reply with an ACK or FAIL message depending on whether extra time can or cannot be granted.

OFFER (Section 9.2.3):
: This operation is used by the server to inform the client about an offer that can best accommodate the requirements indicated in the previously received QUOTATION message.

ACCEPT (Section 9.2.4):
: This operation is used by the client to confirm the acceptance of an offer made by the server. This message implies a call for agreement. An agreement is reached when an ACK is subsequently received from the server, which is likely to happen if the message is sent before the offer validity time expires; the server is unlikely to reject an offer that it has already made.

DECLINE (Section 9.2.5):
: This operation is used by the client to reject an offer made by the server. The ongoing transaction may not be terminated immediately, e.g., the client may issue another order or the server may issue another offer.

ACK (Section 9.2.6):
: This operation is used by the server to acknowledge the receipt of an ACCEPT or WITHDRAW message or by the client to confirm the server's request for a time extension (conveyed in a PROCESSING message) in order to process the last received quotation order.

CANCEL (Section 9.2.7):
: This operation is used by the client to cancel (quit) the ongoing transaction.

WITHDRAW (Section 9.2.8):
: This operation is used by the client to withdraw a completed order (i.e., an agreement).

UPDATE (Section 9.2.9):
: This operation is used by the client to update an existing agreement. For example, this method can be invoked to add a new VPN site. This method will trigger a new negotiation cycle.

FAIL (Section 9.2.10):
: This operation is used by the server to indicate that it cannot accommodate the requirements documented in the PQO conveyed in the QUOTATION message or to inform the client about an error encountered when processing the received message. In either case, the message implies that the server is unable to make offers, and, as a consequence, it terminates the ongoing transaction.

  This message is also used by the client to reject a time extension request in a PROCESSING message received from the server. The message includes a status code that provides explanatory information.

The above CPNP primitives are service independent. CPNP messages may transparently carry service-specific objects that are handled by the negotiation logic at either side.

The document defines the service objects that are required for connectivity provisioning negotiation purposes (see Section 8.7). Additional service-specific objects for CPNP messages to accommodate alternative deployment schemes or other service provisioning needs can be defined in the future.

8.7. Connectivity Provisioning Documents

CPNP makes use of several flavors of Connectivity Provisioning Documents (CPD). These documents follow the same CPP template described in [RFC7297].

Requested CPD:
: Refers to the CPD included by a CPNP client in a QUOTATION request.

Offered CPD:
: This document is included by a CPNP server in an OFFER message. Its information reflects the proposal of the server to accommodate all or a subset of the clauses depicted in a Requested CPD. A validity time is associated with the offer made.

Accepted CPD:
: If the client accepts an offer made by the server, the Offered CPD is included in an ACCEPT message. This CPD is also included in an ACK message. Thus, a three-way handshake procedure is followed for successfully completing the negotiation.

Figure 6 shows a typical CPNP negotiation cycle and the use of the different types of CPDs.

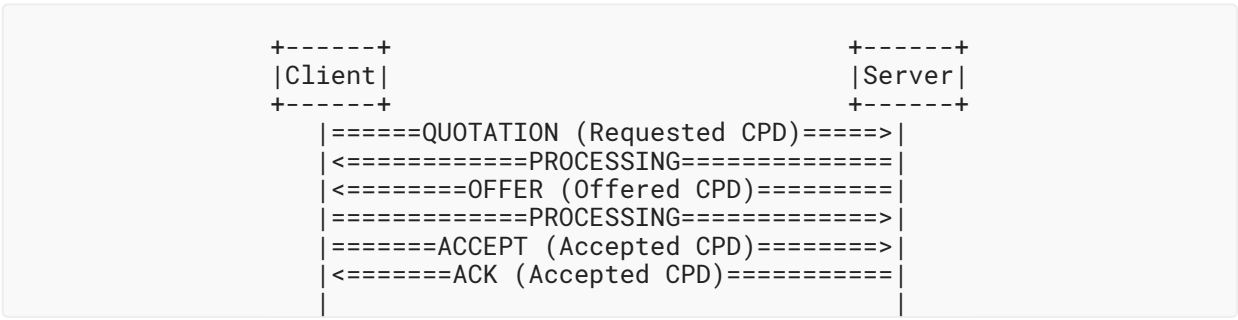

Figure 6: Connectivity Provisioning Documents

A CPD can include parameters with fixed values, loosely defined values, or any combination thereof. A CPD is said to be concrete if all clauses have fixed values.

A typical evolution of a negotiation cycle would start with a quotation order with loosely defined parameters, and then, as offers are made, it would conclude with a concrete CPD for calling for the agreement.

8.8. Child PQOs

If the server detects that network resources from another Network Provider need to be allocated in order to accommodate the requirements described in a PQO (e.g., in the context of an inter-domain VPN service, additional Provider Edge (PE) router resources need to be allocated), the server may generate child PQOs to request the appropriate network provisioning operations (see Figure 7). In such a situation, the server also behaves as a CPNP client. The server associates the parent order with its child PQOs. How this is achieved is implementation specific (e.g., this can be typically achieved by locally adding the reference of the child PQO to the parent order).

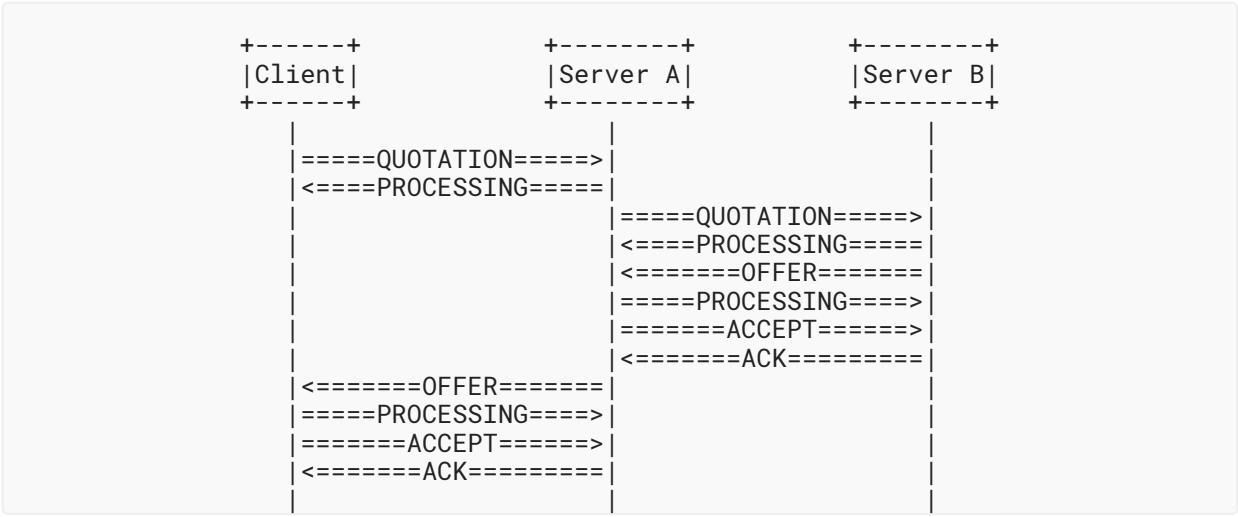

Figure 7: Example of Child Orders

Note that the server must not activate recursion for an order if the client includes a negotiation option to restrict the negotiation scope to the resources of the server's domain (Section 9.1.10.3).

If recursion is not explicitly disabled, the server may notify the client when appropriate (Section 9.2.2). Such notification may depend on the nature of the service and also regulatory considerations.

8.9. Multi-Segment Service

A composite service (e.g., connectivity) requested by a Customer could imply multi-segment services (e.g., multi-segment connectivity spanning an end-to-end scope), in the sense that one single CPNP request is decomposed into multiple connectivity requests on the Provider's side (thereby leading to child orders). The Provider is in charge of handling the complexity of splitting the generic provisioning order in a multi-segment context. Such complexity is local to the Provider.

8.10. Negotiating with Multiple CPNP Servers

A CPNP client may undertake multiple negotiations in parallel with several servers for various reasons, such as cost optimization and fail-safety. These multiple negotiations may lead to one or many agreements.

The salient point underlining the parallel negotiation scenarios is that, although the negotiation protocol is strictly between two parties, this may not be the case of the negotiation logic. The CPNP client negotiation logic may need to collectively drive parallel negotiations, as the negotiation with one server may affect the negotiation with other servers; for example, it may need to use the responses from all servers as an input for determining the messages (and their content) to subsequently send within the course of each individual negotiation. Therefore, timing is an important aspect on the client's side. The CPNP client needs to have the ability to

synchronize the receipt of the responses from the servers. CPNP takes into account this requirement by allowing clients to specify in the QUOTATION message the time by which the server needs to respond (see Section 9.1.6).

8.11. State Management

Both the client and the server maintain repositories to store ongoing orders. How these repositories are maintained is deployment specific. It is out of scope of this document to elaborate on such considerations. Timestamps are also logged to track state change. Tracking may be needed for various reasons, including regulatory or billing ones.

In order to accommodate failures that may lead to the reboot of the client or the server, the use of permanent storage is recommended, thereby facilitating state recovery.

8.11.1. On the Client Side

This is the list of the typical states that can be associated with a given order on the client's side:

Created: The order has been created. It is not handled by the client until the administrator allows it to be processed.

AwaitingProcessing: The administrator has approved the processing of a created order, but the order has not been handled yet.

PQOSent: The order has been sent to the server.

ServerProcessing: The server has confirmed the receipt of the order.

OfferReceived: An offer has been received from the server.

OfferProcessing: A received offer is being processed by the client.

AcceptSent: The client has confirmed the offer to the server.

Completed: The offer has been acknowledged by the server.

Cancelled: The order has failed or was cancelled.

Sub-states may be defined (e.g., to track failed vs. cancelled orders), but those are not shown in Figure 8.

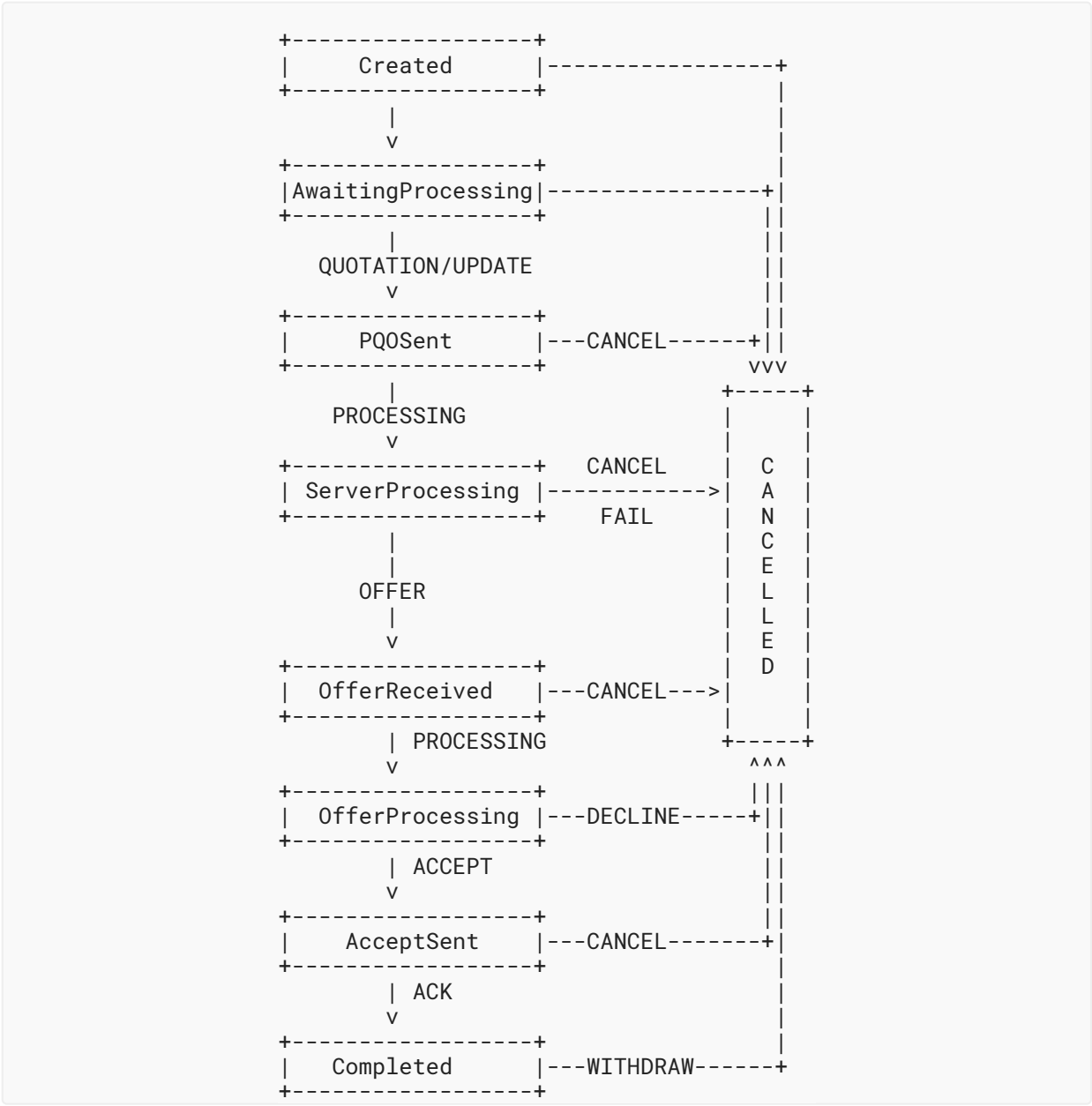

Figure 8: Example of a CPNP Finite State Machine (Client Side)

8.11.2. On the Server Side

The following lists the states on the server's side that can be associated with a given order and a corresponding offer:

PQOReceived: The order has been received from the client.

AwaitingProcessing: The order is being processed by the server. An action from the server administrator may be needed.

OfferProposed: The request has been successfully handled, and an offer has been sent to the client.

ProcessingReceived: The server has received a PROCESSING message for an offer sent to the client.

AcceptReceived: The server has received a confirmation for the offer from the client.

Completed: The server has acknowledged the offer (accepted by client) to the client. Transitioning to this state assumes that the ACK was received by the client (this can be detected by the server if it receives a retransmitted ACCEPT message from the client).

Cancelled: The order cannot be accommodated, or it has been cancelled by the client. Associated resources must be released in the latter case, if previously reserved.

ChildCreated: A child order has been created in cases where resources from another Network Provider are needed.

ChildPQOSent: A child order has been sent to the remote server.

ChildServerProcessing: A child order is being processed by the remote server.

ChildOfferReceived: The remote server has received an offer to a child order.

ChildOfferProcessing: A received offer to a child order is being processed.

ChildAcceptSent: The child offer (the offer received from the remote server in response to a child order) is confirmed to the remote server.

ChildCompleted: The accepted child offer has been acknowledged by the remote server.

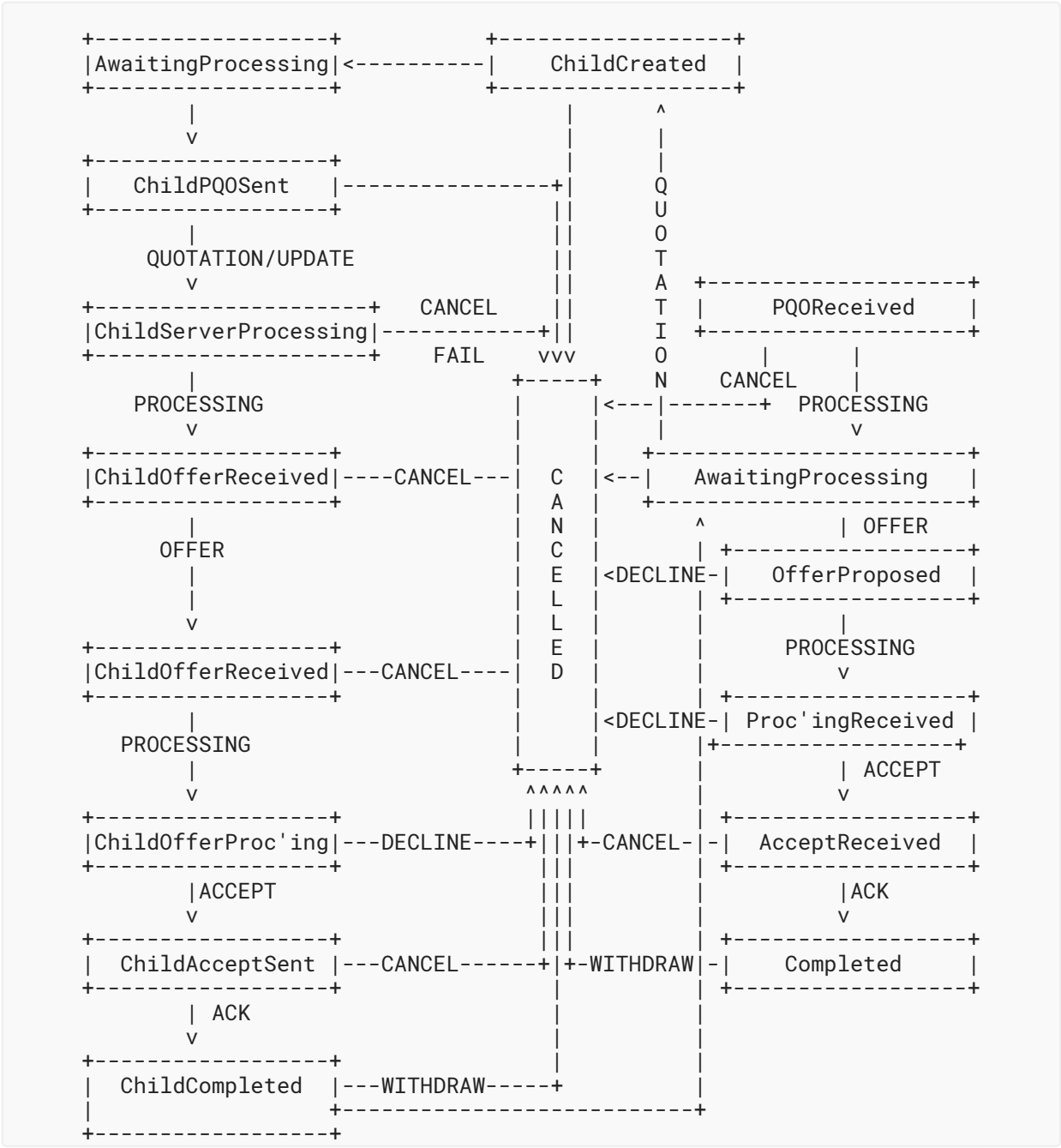

Figure 9: CPNP Finite State Machine (Server Side)

9. CPNP Objects

This section defines CPNP objects using the Routing Backus-Naur Form (RBNF) format defined in [RFC5511]. Please also note the following:

> Note 1: The formats of CPNP messages are provided using a generic format. Implementors can adapt RBNF definitions to their "favorite" message format. For example, JSON [RFC8259] or Concise Binary Object Representation (CBOR) [RFC7049] can be used.

> Note 2: CPNP messages cannot be blindly mapped to RESTCONF messages with the target service being modelled as configuration data because such data is supposed to be manipulated by a RESTCONF client only. In such a model, the RESTCONF server cannot use a value other than the one set by the client (e.g., Section 9.2.3) or remove offers from its own initiative (e.g., Section 9.1.8). An alternate approach might be to map CPNP operations into RESTCONF actions (RPC). Assessing the feasibility of such approach is out of scope.

9.1. Attributes

9.1.1. CUSTOMER_ORDER_IDENTIFIER

The CUSTOMER_ORDER_IDENTIFIER (Customer Order Identifier) is an identifier that is assigned by a client to identify an agreement. This identifier must be unique to the client.

Rules for assigning this identifier (including the structure and semantics) are specific to the client (Customer). The value of CUSTOMER_ORDER_IDENTIFIER is included in all CPNP messages.

The client (Customer) assigns an identifier to an order under negotiation before an agreement is reached. This identifier will be used to unambiguously identify the resulting agreement at the client side (Customer).

The server handles the CUSTOMER_ORDER_IDENTIFIER as an opaque value.

9.1.2. PROVIDER_ORDER_IDENTIFIER

The PROVIDER_ORDER_IDENTIFIER (Provider Order Identifier) is an identifier that is assigned by a server to identify an order. This identifier must be unique to the server.

Rules for assigning this identifier (including the structure and semantics) are specific to the server (Provider). The PROVIDER_ORDER_IDENTIFIER is included in all CPNP messages except QUOTATION messages (because the state is only present at the client side).

The server (Provider) assigns an identifier to an order under negotiation before an agreement is reached. This identifier will be used to unambiguously identify the resulting agreement at the server side (Provider).

The client handles the PROVIDER_ORDER_IDENTIFIER as an opaque value.

9.1.3. TRANSACTION_ID

This object conveys the Transaction-ID introduced in Section 8.4.

9.1.4. SEQUENCE_NUMBER

The sequence number is a number that is monotonically incremented in every new CPNP message pertaining to a given CPNP transaction. This number is used to avoid replay attacks.

Refer to Section 11.3.

9.1.5. NONCE

The NONCE is a random value assigned by the CPNP server. Assigning a unique NONCE value for each order is recommended.

It is mandatory to then include the NONCE in subsequent CPNP client operations on the associated order (including the resulting agreement) such as withdrawing the order or updating the order.

If the NONCE validation checks fail, the server rejects the request with a FAIL message that includes the appropriate failure reason code.

9.1.6. EXPECTED_RESPONSE_TIME

This attribute indicates the time by when the CPNP client is expecting to receive a response from the CPNP server to a given PQO. If no offer is received by then, the CPNP client will consider the quotation order to be rejected.

The EXPECTED_RESPONSE_TIME follows the date format specified in [RFC3339].

9.1.7. EXPECTED_OFFER_TIME

This attribute indicates the time by when the CPNP server is expecting to make an offer to the CPNP client. If no offer is received by then, the CPNP client will consider the order rejected.

The CPNP server may propose an expected offer time that does not match the expected response time indicated in the quotation order message. The CPNP client can accept or reject the proposed expected time by when the CPNP server will make an offer.

The CPNP server can always request extra time for its processing, but this may be accepted or rejected by the CPNP client.

The EXPECTED_OFFER_TIME follows the date format specified in [RFC3339].

9.1.8. VALIDITY_OFFER_TIME

This attribute indicates the time of validity of an offer made by the CPNP server. If the offer is not accepted before this time expires, the CPNP server will consider the CPNP client as having rejected the offer; the CPNP server will silently remove this order from its base.

The VALIDITY_OFFER_TIME follows date format specified in [RFC3339].

9.1.9. SERVICE_DESCRIPTION

This document defines a machinery to negotiate any aspect subject to negotiation. Service clauses that are under negotiation are conveyed using this attribute.

The structure of the connectivity provisioning clauses is provided in the following subsection.

9.1.9.1. CPD

The RBNF format of the CPD is shown in Figure 10.

```
<CPD> ::=  <Connectivity Provisioning Component> ...
<Connectivity Provisioning Component> ::=
                           <CONNECTIVITY_PROVISIONING_PROFILE> ...
<CONNECTIVITY_PROVISIONING_PROFILE> ::=
                           <Customer Nodes Map>
                           <SCOPE>
                           <QoS Guarantees>
                           <Availability>
                           <CAPACITY>
                           <Traffic Isolation>
                           <Conformance Traffic>
                           <Flow Identification>
                           <Overall Traffic Guarantees>
                           <Routing and Forwarding>
                           <Activation Means>
                           <Invocation Means>
                           <Notifications>
<Customer Nodes Map> ::=  <Customer Node> ...
<Customer Node> ::=  <IDENTIFIER>
                     <LINK_IDENTIFIER>
                     <LOCALIZATION>
```

Figure 10: The RBNF format of the CPD

9.1.10. CPNP Information Elements

An Information Element (IE) is an optional object that can be included in a CPNP message.

9.1.10.1. Customer Description

The client may include administrative information such as the following:

- Name
- Contact Information

The format of this Information Element is as follows:

```
<Customer Description> ::= [<NAME>] [<Contact Information>]
<Contact Information> ::=  [<EMAIL_ADDRESS>] [<POSTAL_ADDRESS>]
                           [<TELEPHONE_NUMBER> ...]
```

9.1.10.2. Provider Description

The server may include administrative information in an offer such as the following:

- Name
- AS Number [RFC6793]
- Contact Information

The format of this Information Element is as follows:

```
<Provider Description> ::= [<NAME>][<Contact Information>]
                           [<AS_NUMBER>]
```

9.1.10.3. Negotiation Options

The client may include some negotiation options such as the following:

Setup purpose: A client may request the setup of a service (e.g., connectivity) only for testing purposes during a limited period. The order can be extended to become permanent if the client was satisfied during the test period. This operation is achieved using the UPDATE method.

Activation type: A client may request a permanent or scheduled activation type. If no activation type clause is included during the negotiation, this means that the order will be immediately activated right after the negotiation ends.

The format of this Information Element is as follows:

```
<Negotiation Options> ::= [<PURPOSE>]
```

9.2. Operation Messages

This section defines the RBNF format of CPNP operation messages. The following operation codes are used:

Code	Operation Message	Reference
1	QUOTATION	Section 9.2.1
2	PROCESSING	Section 9.2.2
3	OFFER	Section 9.2.3
4	ACCEPT	Section 9.2.4
5	DECLINE	Section 9.2.5

Code	Operation Message	Reference
6	ACK	Section 9.2.6
7	CANCEL	Section 9.2.7
8	WITHDRAW	Section 9.2.8
9	UPDATE	Section 9.2.9
10	FAIL	Section 9.2.10
11	ACTIVATE	Section 9.2.11

Table 1: CPNP Operation Message Codes

These codes are used to unambiguously identify a CPNP operation; the operation code is conveyed in the METHOD_CODE attribute mentioned in the following subsections.

In the following, VERSION refers to the CPNP version number. This attribute must be set to 1.

9.2.1. QUOTATION

The format of the QUOTATION message is shown below:

```
<QUOTATION Message> ::=  <VERSION>
                         <METHOD_CODE>
                         <SEQUENCE_NUMBER>
                         <TRANSACTION_ID>
                         <CUSTOMER_ORDER_IDENTIFIER>
                         [<EXPECTED_RESPONSE_TIME>]
                         <REQUESTED_CPD>
                         [<INFORMATION_ELEMENT>...]
```

A QUOTATION message must include an order identifier that is generated by the client (CUSTOMER_ORDER_IDENTIFIER). Because several orders can be issued to several servers, the QUOTATION message must also include a Transaction-ID.

The message may include an EXPECTED_RESPONSE_TIME, which indicates by when the client expects to receive an offer from the server. The QUOTATION message must also include a requested service description (that is, a Requested CPD for connectivity services).

The message may include ACTIVATION_TYPE to request a permanent or scheduled activation type (e.g., using the ACTIVATE method defined in Section 9.2.11). If no such clause is included, the default mode is to assume that the order will be active once the accepted activation means are successfully invoked (e.g., Section 3.11 of [RFC7297]).

When the client sends the QUOTATION message to the server, the state of the order changes to "PQOSent" at the client side.

9.2.2. PROCESSING

The format of the PROCESSING message is shown below:

```
<PROCESSING Message> ::= <VERSION>
                         <METHOD_CODE>
                         <SEQUENCE_NUMBER>
                         <TRANSACTION_ID>
                         <CUSTOMER_ORDER_IDENTIFIER>
                         <PROVIDER_ORDER_IDENTIFIER>
                         [<EXPECTED_OFFER_TIME>]
                         [<PROCESSING_SUBCODE>]
```

Upon receipt of a QUOTATION message, the server proceeds with the parsing rules (see Section 10). If no error is encountered, the server generates a PROCESSING response to the client to indicate the PQO has been received and it is being processed. The server must generate an order identifier that identifies the order in its local order repository. The server must copy the content of the CUSTOMER_ORDER_IDENTIFIER and TRANSACTION_ID fields as conveyed in the QUOTATION message. The server may include an EXPECTED_OFFER_TIME by when it expects to make an offer to the client.

Upon receipt of a PROCESSING message, the client verifies whether it has issued a PQO that contains the CUSTOMER_ORDER_IDENTIFIER and TRANSACTION_ID to that server. If no such PQO is found, the PROCESSING message must be silently ignored. If a PQO is found, the client may check whether it accepts the EXPECTED_OFFER_TIME, and then it changes to state of the order to "ServerProcessing".

If the server requires more time to process the quotation order, it may send a PROCESSING message that includes a new EXPECTED_OFFER_TIME. The client can answer with an ACK message if more time is granted (Figure 11) or with a FAIL message if the time extension request is rejected (Figure 12).

The server may provide more details in the PROCESSING_SUBCODE attribute about the reason for requesting more time to process the request. The following codes are defined:

Subcode	Description
1	Upgrade of local resources
2	Request external resources

Table 2: PROCESSING_SUBCODE Codes

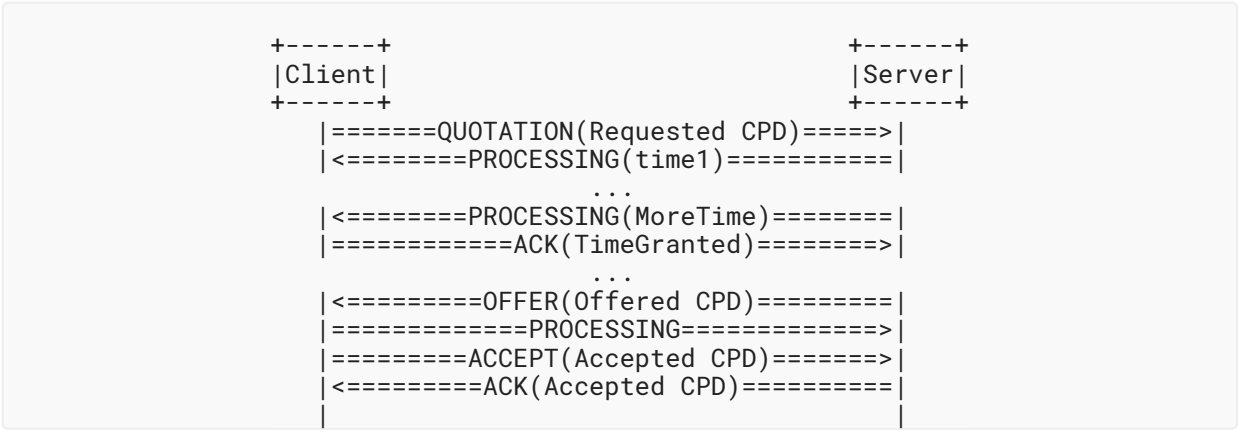

Figure 11: Request More Negotiation Time: Granted

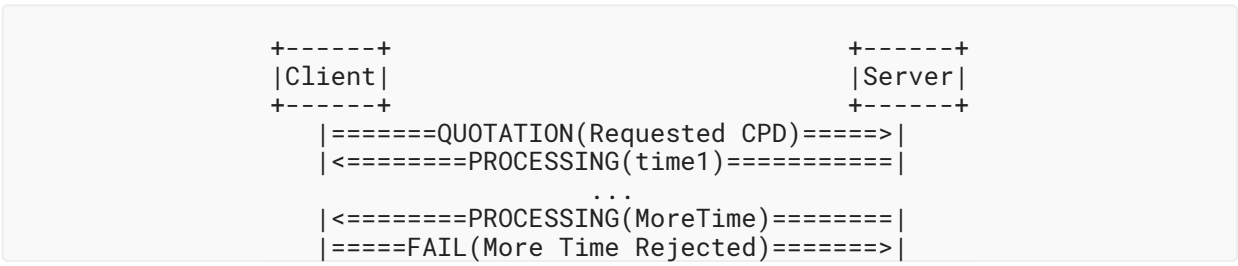

Figure 12: Request More Negotiation Time: Rejected

9.2.3. OFFER

The format of the OFFER message is shown below:

```
<OFFER Message> ::= <VERSION>
                    <METHOD_CODE>
                    <SEQUENCE_NUMBER>
                    <TRANSACTION_ID>
                    <CUSTOMER_ORDER_IDENTIFIER>
                    <PROVIDER_ORDER_IDENTIFIER>
                    <NONCE>
                    <VALIDITY_OFFER_TIME>
                    <OFFERED_CPD>
                    [<INFORMATION_ELEMENT>...]
```

The server answers a QUOTATION request received from the client with an OFFER message. The offer will be considered to be rejected by the client if no confirmation (i.e., an ACCEPT message sent by the client) is received by the server before the expiration of the validity time.

The server may include ACTIVATION_TYPE to indicate whether the offer is about a permanent or scheduled activation type. The message may include ACTIVATION_SCHEDULE to indicate when the order is to be activated. If no such clause is included, the default mode is to assume that the order will be active once the accepted activation means are successfully invoked (e.g., Section 3.11 of [RFC7297] or Section 9.2.11).

9.2.4. ACCEPT

The format of the ACCEPT message is shown below:

```
<ACCEPT Message> ::= <VERSION>
                     <METHOD_CODE>
                     <SEQUENCE_NUMBER>
                     <TRANSACTION_ID>
                     <CUSTOMER_ORDER_IDENTIFIER>
                     <PROVIDER_ORDER_IDENTIFIER>
                     <NONCE>
                     <ACCEPTED_CPD>
                     [<INFORMATION_ELEMENT>...]
```

This message is used by a client to confirm the acceptance of an offer received from a server. The fields of this message must be copied from the received OFFER message. This message should not be sent after the validity time of the offer expires, as indicated by the server (Section 9.2.3).

9.2.5. DECLINE

The format of the DECLINE message is shown below:

```
<DECLINE Message> ::= <VERSION>
                      <METHOD_CODE>
                      <SEQUENCE_NUMBER>
                      <TRANSACTION_ID>
                      <CUSTOMER_ORDER_IDENTIFIER>
                      <PROVIDER_ORDER_IDENTIFIER>
                      <NONCE>
                      [<REASON>...]
```

The client may issue a DECLINE message to reject an offer. CUSTOMER_ORDER_IDENTIFIER, PROVIDER_ORDER_IDENTIFIER, TRANSACTION_ID, and NONCE are used by the server as keys to find the corresponding order. If an order matches, the server changes the state of this order to "Cancelled" and then returns an ACK with a copy of the Requested CPD to the requesting client.

A DECLINE message may include an Information Element to indicate the reason for declining an offer. The following codes are defined:

Code	Description
1	Unacceptable gap between the request and the offer
2	Conflict with another offer from another server
3	Activation type mismatch

Table 3: DECLINE Message Codes

If no order is found, the server returns a FAIL message to the requesting client. In order to prevent DDoS (Distributed Denial of Service) attacks, the server should restrict the number of FAIL messages sent to a requesting client. It may also rate-limit FAIL messages.

A flow example is shown in Figure 13.

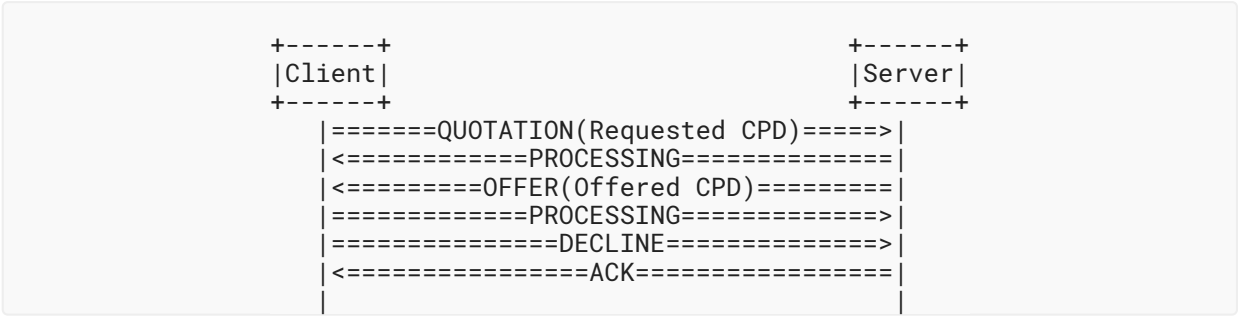

Figure 13: DECLINE Flow Example

9.2.6. ACK

The format of the ACK message is shown below:

```
<ACK Message> ::= <VERSION>
                  <METHOD_CODE>
                  <SEQUENCE_NUMBER>
                  <TRANSACTION_ID>
                  <CUSTOMER_ORDER_IDENTIFIER>
                  <PROVIDER_ORDER_IDENTIFIER>
                  [<EXPECTED_RESPONSE_TIME>]
                  [<CPD>]
                  [<INFORMATION_ELEMENT>...]
```

This message is issued by the server to close a CPNP transaction or by a client to grant more negotiation time to the server.

This message is sent by the server as a response to an ACCEPT, WITHDRAW, DECLINE, or CANCEL message. In this case, the ACK message must include the copy of the service description (i.e., CPD for connectivity services) as stored by the server. In particular, the following considerations are taken into account for connectivity provisioning services:

- A copy of the Requested/Offered CPD is included by the server if it successfully handled a CANCEL message.
- A copy of the Updated CPD is included by the server if it successfully handled an UPDATE message.
- A copy of the Offered CPD is included by the server if it successfully handled an ACCEPT message in the context of a QUOTATION transaction (refer to "Accepted CPD" in Section 8.7).
- An Empty CPD is included by the server if it successfully handled a DECLINE or WITHDRAW message.

A client may issue an ACK message as a response to a time extension request (conveyed in PROCESSING) received from the server. In such case, the ACK message must include an EXPECTED_RESPONSE_TIME that is likely to be set to the time extension requested by the server.

9.2.7. CANCEL

The format of the CANCEL message is shown below:

```
<CANCEL Message> ::= <VERSION>
                     <METHOD_CODE>
                     <SEQUENCE_NUMBER>
                     <TRANSACTION_ID>
                     <CUSTOMER_ORDER_IDENTIFIER>
                     [<CPD>]
```

The client can issue a CANCEL message at any stage during the CPNP negotiation process before an agreement is reached. The CUSTOMER_ORDER_IDENTIFIER and TRANSACTION_ID are used by the server as keys to find the corresponding order. If a quotation order matches, the server changes the state of this quotation order to "Cancelled" and then returns an ACK with a copy of the Requested CPD to the requesting client.

If no quotation order is found, the server returns a FAIL message to the requesting client.

9.2.8. WITHDRAW

The format of the WITHDRAW message is shown below:

```
<WITHDRAW Message> ::= <VERSION>
                       <METHOD_CODE>
                       <SEQUENCE_NUMBER>
                       <TRANSACTION_ID>
                       <CUSTOMER_ORDER_IDENTIFIER>
                       <PROVIDER_ORDER_IDENTIFIER>
                       <NONCE>
                       [<ACCEPTED_CPD>]
                       [<INFORMATION_ELEMENT>...]
```

This message is used to withdraw an offer already accepted by the Customer. Figure 14 shows a typical usage of this message.

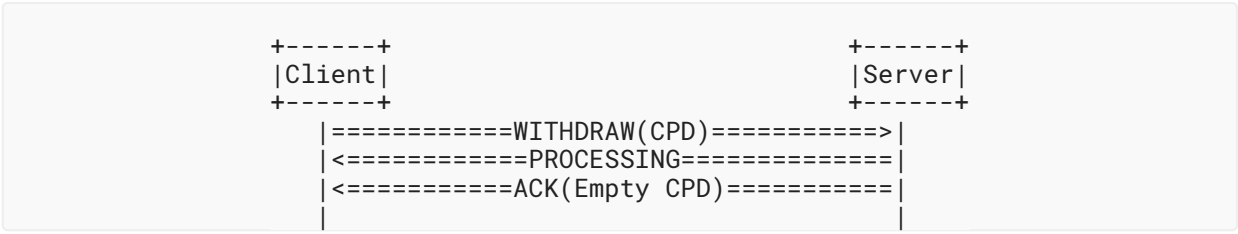

Figure 14: WITHDRAW Flow Example

The WITHDRAW message must include the same CUSTOMER_ORDER_IDENTIFIER, PROVIDER_ORDER_IDENTIFIER, and NONCE as those used when creating the order.

Upon receipt of a WITHDRAW message, the server checks whether an order matching the request is found. If an order is found, the state of the order is changed to "Cancelled", and an ACK message including an Empty CPD is returned to the requesting client. If no order is found, the server returns a FAIL message to the requesting client.

9.2.9. UPDATE

The format of the UPDATE message is shown below:

```
<UPDATE Message> ::= <VERSION>
                     <METHOD_CODE>
                     <SEQUENCE_NUMBER>
                     <TRANSACTION_ID>
                     <CUSTOMER_ORDER_IDENTIFIER>
                     <PROVIDER_ORDER_IDENTIFIER>
                     <NONCE>
                     <EXPECTED_RESPONSE_TIME>
                     <REQUESTED_CPD>
                     [<INFORMATION_ELEMENT>...]
```

This message is sent by the CPNP client to update an existing service agreement (e.g., Accepted CPD). The UPDATE message must include the same CUSTOMER_ORDER_IDENTIFIER, PROVIDER_ORDER_IDENTIFIER, and NONCE as those used when creating the order. The CPNP client includes a new service description (e.g., Updated CPD) that integrates the requested modifications. A new Transaction_ID must be assigned by the client.

Upon receipt of an UPDATE message, the server checks whether an order, having state "Completed", matches CUSTOMER_ORDER_IDENTIFIER, PROVIDER_ORDER_IDENTIFIER, and NONCE.

- If no order is found, the CPNP server generates a FAIL error with the appropriate error code (Section 9.2.10).
- If an order is found, the server checks whether it can honor the request:
  - A FAIL message is sent to the client if the server cannot honor the request. The client may initiate a new PQO negotiation cycle (that is, send a new UPDATE message).
  - An OFFER message including the updated clauses (e.g., Updated CPD) is sent to the client. For example, the server maintains an order for provisioning a VPN service that connects sites A, B, and C. If the client sends an UPDATE message to remove site C, only sites A and B will be included in the OFFER sent by the server to the requesting client.

    Note that the cycle that is triggered by an UPDATE message is also considered to be a negotiation cycle.

A flow chart that illustrates the use of UPDATE operation is shown in Figure 15.

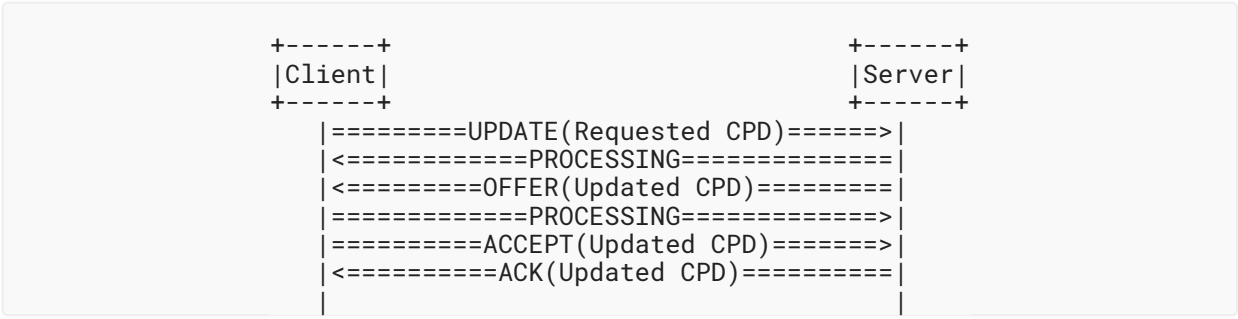

Figure 15: UPDATE Flow Example

9.2.10. FAIL

The format of the FAIL message is shown below:

```
<FAIL Message> ::=  <VERSION>
                    <METHOD_CODE>
                    <SEQUENCE_NUMBER>
                    <TRANSACTION_ID>
                    <CUSTOMER_ORDER_IDENTIFIER>
                    <PROVIDER_ORDER_IDENTIFIER>
                    <STATUS_CODE>
```

This message is sent in the following cases:

- The server cannot honor an order received from the client (i.e., received in a QUOTATION or UPDATE request).
- The server encounters an error when processing a CPNP request received from the client.
- The client cannot grant more time to the server. This is a response to a time extension request carried in a PROCESSING message.

The status code indicates the error code. The following codes are supported:

Status Code	Error Code	Description
1	Message Validation Error	The message cannot be validated (see Section 10).
2	Authentication Required	The request cannot be handled because authentication is required.
3	Authorization Failed	The request cannot be handled because authorization failed.
4	Administratively prohibited	The request cannot be handled because of administrative policies.

Status Code	Error Code	Description
5	Out of Resources	The request cannot be honored because resources (e.g., capacity) are insufficient.
6	Network Presence Error	The request cannot be honored because there is no network presence.
7	More Time Rejected	The request to extend the time for negotiation is rejected by the client.
8	Unsupported Activation Type	The request cannot be handled because the requested activation type is not supported.

Table 4: FAIL Message Error Codes

9.2.11. ACTIVATE

The format of the ACTIVATE message is shown below:

```
<ACTIVATE Message> ::= <VERSION>
                       <METHOD_CODE>
                       <SEQUENCE_NUMBER>
                       <TRANSACTION_ID>
                       <CUSTOMER_ORDER_IDENTIFIER>
                       <PROVIDER_ORDER_IDENTIFIER>
                       <NONCE>
                       <ACTIVATION_SCHEDULE>
                       [<INFORMATION_ELEMENT>...]
```

This message is sent by the CPNP client to request the activation of an existing service agreement. The message must include the same CUSTOMER_ORDER_IDENTIFIER, PROVIDER_ORDER_IDENTIFIER, and NONCE as those used when creating the order. The CPNP client may include a schedule target for activating this order. A new Transaction_ID must be assigned by the client.

Upon receipt of an ACTIVATE message, the server checks whether an order, having state "Completed", matches CUSTOMER_ORDER_IDENTIFIER, PROVIDER_ORDER_IDENTIFIER, and NONCE.

- If no completed order is found, the CPNP server generates a FAIL error with the appropriate error code (Section 9.2.10).
- If an order is found, the server checks whether it can honor the request:
  - A FAIL message is sent to the client if the server cannot honor the request (e.g., out of resources or explicit activation wasn't negotiated with this client).
  - An ACK is sent to the client to confirm that the immediate activation (or deactivation) of the order or its successful scheduling if a non-null ACTIVATION_SCHEDULE was included

in the request. Note that setting ACTIVATION_SCHEDULE to 0 in an ACTIVATE request has a special meaning: it is used to request a deactivation of an accepted order.

Figure 16 illustrates the use of the ACTIVATE operation.

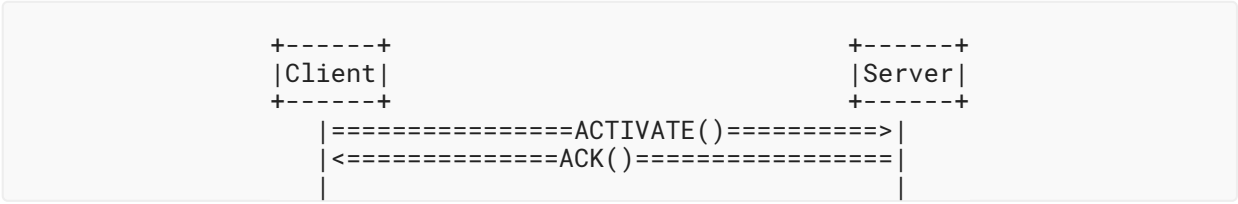

Figure 16: ACTIVATE Flow Example

10. CPNP Message Validation

Both the client and the server proceed with CPNP message validation. The following tables summarize the validation checks to be followed.

10.1. On the Client Side

Operation	Validation Checks
PROCESSING	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier} must match an existing PQO with a state set to "PQOSent". The sequence number carried in the packet must be larger than the sequence number maintained by the client.
OFFER	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier} must match an existing order with state set to "PQOSent", or {Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier} must match an existing order with a state set to "ServerProcessing". The sequence number carried in the packet must be larger than the sequence number maintained by the client.
ACK (QUOTATION Transaction)	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier, Offered Connectivity Provisioning Document} must match an order with a state set to "AcceptSent". The sequence number carried in the packet must be larger than the sequence number maintained by the client.

Operation	Validation Checks
ACK (UPDATE Transaction)	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier, Updated Connectivity Provisioning Document} must match an order with a state set to "AcceptSent". The sequence number carried in the packet must be larger than the sequence number maintained by the client.
ACK (WITHDRAW Transaction)	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier, Empty Connectivity Provisioning Document} must match an order with a state set to "Cancelled". The sequence number carried in the packet must be larger than the sequence number maintained by the client.

Table 5: Client Side Validation Checks

10.2. On the Server Side

Method	Validation Checks
QUOTATION	The source IP address passes existing access filters (if any). The sequence number carried in the packet must not be lower than the sequence number maintained by the server.
PROCESSING	The sequence number carried in the packet must be greater than the sequence number maintained by the server.
CANCEL	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier} must match an order with state set to "PQOReceived" or "OfferProposed" or "ProcessingReceived" or "AcceptReceived". The sequence number carried in the packet must be greater than the sequence number maintained by the server.
ACCEPT	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier, Nonce, Offered Connectivity Provisioning Document} must match an order with state set to "OfferProposed" or "ProcessingReceived". The sequence number carried in the packet must be greater than the sequence number maintained by the server.
FAIL	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier} must match an order with state set to "AwaitingProcessing" and for which a request to grant more time to process an offer was requested. The sequence number carried in the packet must be greater than the sequence number maintained by the server.

Method	Validation Checks
DECLINE	{Source IP address, source port number, destination IP address, destination port number, Transaction-ID, Customer Order Identifier, Provider Order Identifier, Nonce} must match an order with state set to "OfferProposed" or "ProcessingReceived". The sequence number carried in the packet must be greater than the sequence number maintained by the server.
UPDATE	The source IP address passes existing access filters (if any), and {Customer Order Identifier, Provider Order Identifier, Nonce} must match an existing order with state "Completed".
WITHDRAW	The source IP address passes existing access filters (if any), and {Customer Order Identifier, Provider Order Identifier, Nonce} must match an existing order with state "Completed".
ACTIVATE	The source IP address passes existing access filters (if any), and {Customer Order Identifier, Provider Order Identifier, Nonce} must match an existing order with a state of "Completed" and its activation procedure set to explicit.

Table 6: Server Side Validation Checks

11. Theory of Operation

Both the CPNP client and server proceed with message validation checks as specified in Section 10.

11.1. Client Behavior

11.1.1. Order Negotiation Cycle

To place a PQO, the client first initiates a local quotation order object identified by a unique identifier assigned by the client (Client Order Identifier). The state of the quotation order is set to "Created". The client then generates a QUOTATION request that includes the assigned identifier, possibly an expected response time, a Transaction-ID, and a requested service (e.g., Requested CPD). The client may include additional Information Elements such as Customer Description or Negotiation Options.

The client may be configured to not enforce negotiation checks on EXPECTED_OFFER_TIME; if so, the client should either not include the EXPECTED_RESPONSE_TIME attribute in the PQO or it should set the attribute to infinite.

Once the request is sent to the server, the state of the request is set to "PQOSent", and if a response time is included in the quotation order, a timer is set to the expiration time as included in the QUOTATION request. The client also maintains a copy of the CPNP session entry details used to generate the QUOTATION request. The CPNP client must listen on the same port number that it used to send the QUOTATION request.

If no answer is received from the server before the retransmission timer expires (i.e., RETRANS_TIMER, Section 8.5), the client retransmits the message until maximum retry is reached (e.g., three times). The same sequence number is used for retransmitted packets.

If a FAIL message is received, the client may decide to issue another (corrected) request towards the same server, cancel the local order, or contact another server. The behavior of the client depends on the error code returned by the server in the FAIL message.

If a PROCESSING message matching the CPNP session entry (Section 8.3) is received, the client updates the CPNP session entry with the PROVIDER_ORDER_IDENTIFIER information. If the client does not accept the expected offer time that may have been indicated in the PROCESSING message, the client may decide to cancel the quotation order. If the client accepts the EXPECTED_OFFER_TIME, it changes the state of the order to "ServerProcessing" and sets a timer to the value of EXPECTED_OFFER_TIME. If no offer is made before the timer expires, the client changes the state of the order to "Cancelled".

As a response to a time extension request (conveyed in a PROCESSING message that included a new EXPECTED_OFFER_TIME), the client may either grant this extension by issuing an ACK message or reject the time extension by issuing a FAIL message with a status code set to "More Time Rejected".

If an OFFER message matching the CPNP session entry is received, the client checks if a PROCESSING message having the same PROVIDER_ORDER_IDENTIFIER has been received from the server. If a PROCESSING message was already received for the same order, but the PROVIDER_ORDER_IDENTIFIER does not match the identifier included in the OFFER message, the client silently ignores the message. If a PROCESSING message with the same PROVIDER_ORDER_IDENTIFIER was already received and matches the CPNP transaction identifier, the client changes the state of the order to "OfferReceived" and sets a timer to the value of VALIDITY_OFFER_TIME indicated in the OFFER message.

If an offer is received from the server (i.e., as documented in an OFFER message), the client may accept or reject the offer. The client accepts the offer by generating an ACCEPT message that confirms that the client agrees to subscribe to the offer documented in the OFFER message; the state of the order is passed to "AcceptSent". The transaction is terminated if an ACK message is received from the server. If no ACK is received from the server, the client proceeds with the retransmission of the ACCEPT message until the maximum retry is reached (Section 11.4).

The client may also decide to reject the offer by sending a DECLINE message. The state of the order is set by the client to "Cancelled". If an offer is not acceptable to the client, the client may decide to contact a new server or submit another order to the same server. Guidelines to issue an updated order or terminate the negotiation are specific to the client.

An order can be activated (or deactivated) using the ACTIVATE message or other accepted activation means (Section 3.11 of [RFC7297]).

11.1.2. Order Withdrawal Cycle

A client may withdraw a completed order. This is achieved by issuing a WITHDRAW message. This message must include the Customer Order Identifier, Provider Order Identifier, and Nonce returned during the order negotiation cycle, as specified in Section 11.1.1.

If no ACK is received from the server, the client proceeds with the retransmission of the message. If no ACK is received after the maximum retry is exhausted, the client should log the information and must send an alarm to the administrator. If there is no specific instruction from the administrator, the client should schedule another Withdrawal cycle. The client must not retry this Withdrawal cycle more frequently than every 300 seconds and must not retry more frequently than every 60 seconds.

11.1.3. Order Update Cycle

A client may update a completed order. This is achieved by issuing an UPDATE message. This message must include the Customer Order Identifier, Provider Order Identifier, and Nonce returned during the order negotiation cycle specified in Section 11.1.1. The client must include in the UPDATE message an Updated CPD with the requested changes.

The subsequent message exchange is similar to what is documented in Section 11.1.1.

11.2. Server Behavior

11.2.1. Order Processing

Upon receipt of a QUOTATION message from a client, the server sets a CPNP session, stores the Transaction-ID, and generates a Provider Order Identifier. Once preliminary validation checks are completed (Section 10), the server may return a PROCESSING message to inform the client that the quotation order is received and it is under processing; the server may include an expected offer time to notify the client by when an offer will be proposed. An order with state "AwaitingProcessing" is created by the server. The server runs its decision-making process to decide which offer it can make to honor the received order. The offer should be made before the expected offer time expires.

If the server cannot make an offer, it sends backs a FAIL message with the appropriate error code (Section 9.2.10).

If the server requires more negotiation time, it must send a PROCESSING message with a new EXPECTED_OFFER_TIME. The client may grant this extension by issuing an ACK message or reject the time extension by issuing a FAIL message with the status code set to "More Time Rejected". If the client doesn't grant more time, the server must answer before the initial expected offer time; otherwise, the client will decline the quotation order.

If the server can honor the request, or if it can make an offer that meets only some of the requirements, it creates an OFFER message. The server must indicate the Transaction-ID, the Customer Order Identifier as indicated in the QUOTATION message, and the Provider Order Identifier generated for this order. The server must also include the Nonce and the offered

service document (e.g., Offered CPD). The server includes an offer validity time as well. Once sent to the client, the server changes the state of the order to "OfferProposed", and a timer set to the validity time is initiated.

If the server determines that additional network resources from another Network Provider are needed to accommodate a quotation order, it will create child PQO(s) and will behave as a CPNP client to negotiate child PQO(s) with possible partnering Providers (see Figure 7).

If no PROCESSING, ACCEPT, or DECLINE message is received before the expiry of the RETRANS_TIMER, the server resends the same offer to the client. This procedure is repeated until maximum retry is reached.

If an ACCEPT message is received before the offered validity time expires, the server proceeds with validation checks as specified in Section 10. The state of the corresponding order is passed to "AcceptReceived". The server sends back an ACK message to terminate the order processing cycle.

If a CANCEL or a DECLINE message is received, the server proceeds with the cancellation of the order. The state of the order is then passed to "Cancelled".

11.2.2. Order Withdrawal

A client may withdraw a completed order by issuing a WITHDRAW message. Upon receipt of a WITHDRAW message, the server proceeds with the validation checks, as specified in Section 10:

- If the checks fail, a FAIL message is sent back to the client with the appropriate error code (e.g., 1 (Message Validation Error), 2 (Authentication Required), or 3 (Authorization Failed)).
- If the checks succeed, the server clears the clauses of the CPD, changes the state of the order to "Cancelled", and sends back an ACK message with an Empty CPD.

11.2.3. Order Update

A client may update an order by issuing an UPDATE message. Upon receipt of an UPDATE message, the server proceeds with the validation checks as specified in Section 10:

- If the checks fail, a FAIL message is sent back to the client with the appropriate error code (e.g., 1 (Message Validation Error), 2 (Authentication Required), 3 (Authorization Failed), or 6 (Network Presence Error)).
- The exchange of subsequent messages is similar to what is specified in Section 11.1.1. The server should generate a new Nonce value to be included in the offer made to the client.

11.3. Sequence Numbers

In each transaction, sequence numbers are used to protect the transaction against replay attacks. Each communicating partner of the transaction maintains two sequence numbers, one for incoming packets and one for outgoing packets. When a partner receives a message, it will check whether the sequence number in the message is larger than the incoming sequence number maintained locally. If not, the message will be discarded. If the message is proved to be legitimate, the value of the incoming sequence number maintained locally will be replaced by

the value of the sequence number in the message. When a partner sends out a message, it will insert the value of the outgoing sequence number into the message and increase the outgoing sequence number maintained locally by 1.

11.4. Message Retransmission

If a transaction partner sends out a message and does not receive any expected reply before the retransmission timer expires (i.e., RETRANS_TIMER), a transaction partner will try to retransmit the message. The procedure is reiterated until a maximum retry is reached (e.g., three times). An exception is the last message (e.g., ACK) sent from the server in a transaction. After sending this message, the retransmission timer will be disabled since no additional feedback is expected.

In addition, if the partner receives a retransmission of the last incoming packet it handled, the partner can resend the same answer to the incoming packet with a limited frequency. If an answer cannot be generated right after the request is received, the partner needs to generate a PROCESSING message as the answer.

To optimize message retransmission, a partner could also store the last incoming packet and the associated answer. Note that the times of retransmission could be decided by the local policy, and retransmission will not cause any change of sequence numbers.

12. Some Operational Guidelines

12.1. CPNP Server Logging

The CPNP server should be configurable to log various events and associated information. Such information may include the following:

- Client's IP address
- Any event change (e.g., new quotation order, offer sent, order confirmation, order cancellation, order withdrawal, etc.)
- Timestamp

The exact logging details are deployment specific.

12.2. Business Guidelines and Objectives

The CPNP server can operate in the following modes:

Fully automated mode:
: The CPNP server is provisioned with a set of business guidelines and objectives that will be used as an input to the decision-making process. The CPNP server will service received orders that fall into these business guidelines; otherwise, requests will be escalated to an administrator that will formally validate or invalidate an order request. The set of policies to be configured to the CPNP server are specific to each administrative entity managing a CPNP server.

Administrative-based mode:

This mode assumes some or all of the CPNP server's operations are subject to a formal administrative validation. CPNP events will trigger appropriate validation requests that will be forwarded to the contact person(s) or department that is responsible for validating the orders. Administrative validation messages are relayed using another protocol (e.g., SMTP) or a dedicated tool.

Business guidelines are local to each administrative entity. How validation requests are presented to an administrator are out of scope of this document; each administrative entity may decide the appropriate mechanism to enable for that purpose.

13. Security Considerations

Means to defend the server against denial-of-service attacks must be enabled. For example, access control lists can be enforced on the client, the server, or the network in between to allow a trusted client to communicate with a trusted server.

The client and the server must be mutually authenticated. Authenticated encryption must be used for data confidentiality and message integrity.

The protocol does not provide security mechanisms to protect the confidentiality and integrity of the packets transported between the client and the server. An underlying security protocol such as (e.g., Datagram Transport Layer Security (DTLS) [RFC6347], Transport Layer Security (TLS) [RFC8446]) must be used to protect the integrity and confidentiality of protocol messages. In this case, if it is possible to provide automated key management (Section 2.1 of [RFC4107]) and associate each transaction with a different key, inter-transaction replay attacks can naturally be addressed. If the client and the server use a single key, an additional mechanism should be provided to protect against inter-transaction replay attacks between them. Clients must implement DTLS record replay detection (Section 3.3 of [RFC6347]) or an equivalent mechanism to protect against replay attacks.

DTLS and TLS with a cipher suite offering confidentiality protection and the guidance given in [RFC7525] must be followed to avoid attacks on (D)TLS.

The client must silently discard CPNP responses received from unknown CPNP servers. The use of a randomly generated Transaction-ID makes it hard to forge a response from a server with a spoofed IP address belonging to a legitimate CPNP server. Furthermore, CPNP demands that messages from the server must include the correct identifiers of the orders. Two order identifiers are used: one generated by the client and a second one generated by the server. Both the CPNP client and server maintain the local identifier they assigned and the one assigned by the peer for a given order. Means to detect swapping of these identifiers (even when such swapping occurs inadvertently at the client or the server) should be enabled by CPNP clients/servers. For example, the CPNP server should not assign a Provider agreement identifier that is equal to a Customer agreement identifier used by the CPNP client.

The Provider must enforce the means to protect privacy-related information included in the documents (see Section 8.7) exchanged in CPNP messages [RFC6462]. In particular, this information must not be revealed to external parties without the consent of Customers. Providers should enforce policies to make Customer fingerprinting difficult to achieve (e.g., in a recursion request). For more discussion about privacy, refer to [RFC6462] [RFC6973].

The Nonce and the Transaction-ID attributes provide sufficient randomness and can effectively tolerate attacks raised by off-path adversaries, who do not have the capability of eavesdropping and intercepting the packets transported between the client and the server. Only authorized clients must be able to modify accepted CPNP orders. The use of a randomly generated Nonce by the server makes it hard to modify an agreement on behalf of a malicious third party.

14. IANA Considerations

This document has no IANA actions.

15. References

15.1. Normative References

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, DOI 10.17487/RFC5511, April 2009, <https://www.rfc-editor.org/info/rfc5511>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <https://www.rfc-editor.org/info/rfc7297>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

15.2. Informative References

[AGAVE] Boucadair, M., Georgatsos, P., Wang, N., Griffin, D., Pavlou, G., Howarth, M., and A. Elizondo, "The AGAVE Approach for Network Virtualization: Differentiated Services Delivery", Annals of Telecommunication, Volume 64, 277-288, DOI 10.1007/s12243-009-0103-4, April 2009, <https://rd.springer.com/article/10.1007/s12243-009-0103-4>.

[COPS-SLS] Nguyen, T., "COPS Usage for SLS negotiation (COPS-SLS)", Work in Progress, Internet-Draft, draft-nguyen-rap-cops-sls-03, 5 July 2002, <https://tools.ietf.org/html/draft-nguyen-rap-cops-sls-03>.

[DSNP] Chen, J., "Dynamic Service Negotiation Protocol (DSNP)", Work in Progress, Internet-Draft, draft-itsumo-dsnp-03, 2 March 2006, <https://tools.ietf.org/html/draft-itsumo-dsnp-03>.

[ETICS] EU FP7 ETICS Project, "Economics and Technologies of Inter-Carrier Services", January 2014, <https://cordis.europa.eu/project/id/248567>.

[L2VPN-NETWORK-YANG] Barguil, S., Dios, O. G. D., Boucadair, M., Munoz, L. A., Jalil, L., and J. Ma, "A Layer 2 VPN Network YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-l2nm-00, 2 July 2020, <https://tools.ietf.org/html/draft-ietf-opsawg-l2nm-00>.

[L3VPN-NETWORK-YANG] Barguil, S., Dios, O. G. D., Boucadair, M., Munoz, L. A., and A. Aguado, "A Layer 3 VPN Network YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-l3sm-l3nm-05, 16 October 2020, <https://tools.ietf.org/html/draft-ietf-opsawg-l3sm-l3nm-05>.

[LISP-MS-DISCOVERY] Boucadair, M. and C. Jacquenet, "LISP Mapping Service Discovery at Large", Work in Progress, Internet-Draft, draft-boucadair-lisp-idr-ms-discovery-01, 9 March 2016, <https://tools.ietf.org/html/draft-boucadair-lisp-idr-ms-discovery-01>.

[NETSLICES-ARCH] Geng, L., Dong, J., Bryant, S., Makhijani, K., Galis, A., Foy, X. D., and S. Kuklinski, "Network Slicing Architecture", Work in Progress, Internet-Draft, draft-geng-netslices-architecture-02, 3 July 2017, <https://tools.ietf.org/html/draft-geng-netslices-architecture-02>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC3084] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, DOI 10.17487/RFC3084, March 2001, <https://www.rfc-editor.org/info/rfc3084>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <https://www.rfc-editor.org/info/rfc4026>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/info/rfc4107>.

[RFC4176] El Mghazli, Y., Ed., Nadeau, T., Boucadair, M., Chan, K., and A. Gonguet, "Framework for Layer 3 Virtual Private Networks (L3VPN) Operations and Management", RFC 4176, DOI 10.17487/RFC4176, October 2005, <https://www.rfc-editor.org/info/rfc4176>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6462] Cooper, A., "Report from the Internet Privacy Workshop", RFC 6462, DOI 10.17487/RFC6462, January 2012, <https://www.rfc-editor.org/info/rfc6462>.

[RFC6574] Tschofenig, H. and J. Arkko, "Report from the Smart Object Workshop", RFC 6574, DOI 10.17487/RFC6574, April 2012, <https://www.rfc-editor.org/info/rfc6574>.

[RFC6770] Bertrand, G., Ed., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, DOI 10.17487/RFC6770, November 2012, <https://www.rfc-editor.org/info/rfc6770>.

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <https://www.rfc-editor.org/info/rfc6793>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, DOI 10.17487/RFC7215, April 2014, <https://www.rfc-editor.org/info/rfc7215>.

[RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <https://www.rfc-editor.org/info/rfc7491>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

[RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <https://www.rfc-editor.org/info/rfc8309>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

[RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <https://www.rfc-editor.org/info/rfc8466>.

[RFC8597] Contreras, LM., Bernardos, CJ., Lopez, D., Boucadair, M., and P. Iovanna, "Cooperating Layered Architecture for Software-Defined Networking (CLAS)", RFC 8597, DOI 10.17487/RFC8597, May 2019, <https://www.rfc-editor.org/info/rfc8597>.

[RNAP] Wang, X., "A Resource Negotiation and Pricing Protocol (RNAP)", <http://www.cs.columbia.edu/~xinwang/public/projects/protocol.html>.

[SNAP] Czajkowski, K., Foster, I., Kesselman, C., Sander, V., and S. Tuecke, "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems", DOI 10.1.1.19.5907, 2002, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.5907>.

[SrNP] Georgatsos, P. and G. Giannakopoulos, "Service Negotiation Protocol (SrNP)", <https://www.ist-tequila.org/presentations/srnp-pipcm.pdf>.

[TEAS-SLICE-NBI] Contreras, L. M., Homma, S., and J. A. Ordonez-Lucena, "Considerations for defining a Transport Slice NBI", Work in Progress, Internet-Draft, draft-contreras-teas-slice-nbi-02, 13 July 2020, <https://tools.ietf.org/html/draft-contreras-teas-slice-nbi-02>.

Acknowledgements

Thanks to Diego R. Lopez, Adrian Farrel, Éric Vyncke, Eric Kline, and Benjamin Kaduk for the comments.

Thanks to those that reviewed this document for publication in the Independent Stream.

Special thanks to Luis Miguel Contreras Murillo for the detailed review.

Authors' Addresses

Mohamed Boucadair (EDITOR)
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Christian Jacquenet
Orange
35000 Rennes
France
Email: christian.jacquenet@orange.com

Dacheng Zhang
Huawei Technologies
Email: dacheng.zhang@huawei.com

Panos Georgatsos
Centre for Research and Innovation Hellas
78, Filikis Etairias str.
38334 Volos
Greece
Phone: +302421306070
Email: pgeorgat@gmail.com