Defence Research and Development Canada
Recherche et développement pour la défense Canada


# Policy Negotiation Proxy (PNP) Software Design Document

J. Spagnolo, D. Cayer



**Defence R&D Canada – Ottawa**
CONTRACT REPORT
DRDC Ottawa CR 2005-111
July 2005

Canada

# Policy Negotiation Proxy (PNP) Software Design Document

J. Spagnolo, D. Cayer
NRNS Incorporated

NRNS Incorporated
4043 Carling Avenue
Ottawa, ON
K2K 2A3

Contract Number: W7714-3-800/001/SV

Contract Scientific Authority: Dr. S. Zeber (613) 991-1388

Contract Scientific Advisor: Mr. Tim Symchych, CRC, (613) 949 - 3070

The work described in this report was sponsored jointly by the Department of National Defence under the work unit 15BF and by the Communications Research Center Canada under the work units 15CS and 15CV.

## Defence R&D Canada – Ottawa

Contract Report

DRDC Ottawa CR 2005-111

July 2005

**Terms of release:** The scientific or technical validity of this Contract Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada or the Communications Research Centre.

# Document Revision History

| | | |
|---|---|---|
| Version 0.1 | 31-Mar-2005 | Complete DRAFT submitted Scientific Authority for review. |
| Version 1.0 | 31-Mar-2005 | Incorporated comments from Defence R&D Canada and the Communications Research Centre. |

# Table of Contents

# List of Figures

# 1. Introduction

Policy Based Network Management (PBNM) systems provide an automated means to configure and administer Policy Enforcement Point (PEP) devices such as virtual private network (VPN) gateways, firewalls and routers. The Policy Decision Point (PDP) takes high level policies as input and produces lower level PEP specific policies as output. The PBNM system can process different types of policies. When evaluating policies, the PDP must identify and resolve conflicts within competing policies as well as take into consideration external factors such as the time-of-day and the current threat level.

A PBNM system alleviates the need for network administrators to manually configure numerous network devices in order to implement local policy changes. We also introduce the concept of policy negotiation for inter-domain policies[1] such as inter-domain security policies. Negotiable policies are not complete policy documents and therefore the PDP cannot directly implement them. Instead, the local PDP must exchange policy proposals with a PDP in a remote administrative domain. Policy proposals contain all the negotiation parameters needed by the other party to correctly evaluate the proposed policy against the local policy. A PDP can accept a proposed policy in whole or in part or it can reject the proposed policy. If both parties accept the other party's proposed policy in whole or in part, each party merges the local and remote policy proposals to form a complete policy document that the PDP can implement. The PBNM system automatically reconfigures network devices as required to implement negotiated policies.

# 2. Purpose

This document presents a software design for the Policy Negotiation Proxy (PNP) component of PBNM system described in the "Policy Based Network Management - System Design Document" [PBNM]. The system facilitates the exchange and negotiation of policies on behalf of the Policy Decision Point (PDP) component.

The PNP is implemented in Java. The primary goal of this software design is to create an extensible policy negotiation framework that can support different types of policies without requiring significant modifications to the existing software. Moreover, this software design makes extensive use of Java software interfaces to define the functions of the major software components. This allows for different implementations of the same software component to be substituted into the system without affecting the remainder of the system.

# 3. Software Architecture

Figure 1 illustrates major software components that comprise software architecture for the PNP. Most of the functionality initially allocated to the PNP has since been absorbed by the PDP component. Most software components are defined by a Java interface that describes their intended purpose. This hides the implementation details and allows other objects to interact with the software component using the generic Java interface.

[1] Note that inter-domain policies may also be intra-organizational policies as some "domains" will be identified parts of larger organizations.

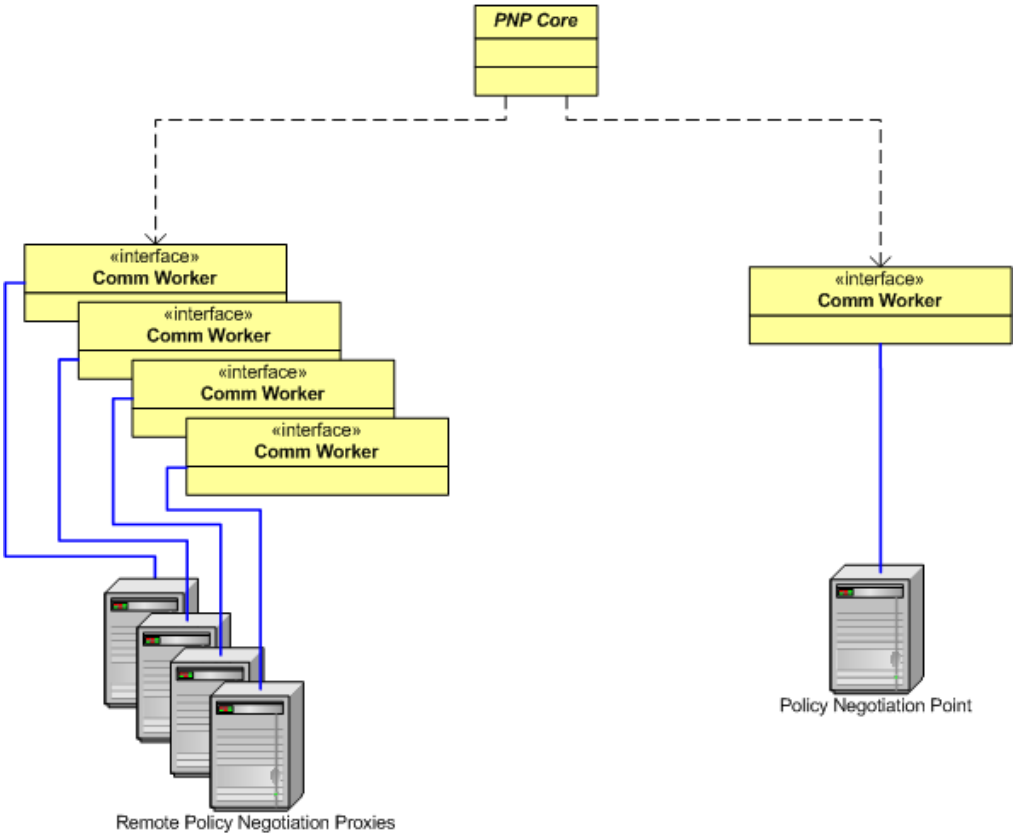


**Figure 1 - PNP High Level Software Architecture**

The PNP simply facilitates the negotiation with remote PNP devices. The PNP is truly generic, since it does not interpret the payload carried within the policy negotiation objects. The PNP can facilitate the concurrent negotiation of different types of policies.

The *PNP Core* forms the core of the PNP system. It makes use of a *Comm Worker* to interact with the PDP system and employs numerous *Comm Worker* instances to communicate with PNPs in remote administrative domains – one *Comm Worker* per remote administrative domain.

Figure 1 does not show the *Status Code Manager*, which maps PBNM status codes to informative strings. Like the other major software components, the *Status Code Manager* is also a Java interface that can realize many different implementations. Internationalization can be easily achieved by creating different implementations of the *Status Code Manager* interface.

# 4. Compliance with System Design

The software design described in this document implements the PNP portion of the PBNM system described in [PBNM] – with the following exceptions:

1. The communication between the PDP and the PNP as well as the inter-PNP communication is implemented using JGroups [JGROUPS].

2. Due to limitations with JGroups, the PDP/PNP communication channel and the inter-PNP communication channels are not authenticated or secured. The system is easily extensible

however and could easily incorporate different communication frameworks (i.e. Transport Layer Security, Session Initiation Protocol).

3. The PNP system does not currently perform status checks on certificates used to authenticate communication channels.

# 5. Software Design

This section presents a high-level design for each PNP software component. It provides an overview of each software component, it lists the requirements for each software component, and it outlines the design for each software component in terms of processing, thread synchronization and exceptions.

## 5.1 Base PBNM Objects

This section provides a description of some of the base PBNM software objects referenced in the remainder of this document.

| | |
|---|---|
| AlertListener | An AlertListener is an interface that a class implements when it wants to be alerted of asynchronous events from objects executing in a different thread. The alert() method accepts a single object of type Object as an argument. |
| Reminder | A Reminder is an object that dispatches an alert at a specific time in the future. The recipient of the alert must be an AlertListener. |
| NetworkID | A NetworkID contains the information that is needed to communicate with another network entity. This includes network layer (IPv4 or IPv6) addresses and transport port numbers. |
| SynchronizedQueue | A SynchronizedQueue allows two threads to exchange objects/messages in a synchronized fashion. |
| PNU | A policy negotiation unit (PNU) carries policy negotiation objects between the PNP and the PPU responsible for its processing. A PNU includes a header that identifies the type of policy and the type of policy object contained within its payload. |

## 5.2 PNP

The PNP system acts as a proxy for the local PDP system in negotiating policies with remote administrative domains. Figure 2 shows a more detailed UML (Unified Modelling Language) class diagram for the PNP System. The *PNPCore* forms the core of the PNP system and executes in the context of the main program thread. It makes use of a *CommWorker* to interact with the PDP system and employs numerous *CommWorker* instances to communicate with PNPs in remote administrative domains – one *CommWorker* per remote administrative domain.

The *PNPCore* class implements the *AlertListener* interface, which mandates the implementation of the following public method:

| | |
|---|---|
| alert() | Submit a PNU from a *CommWorker* to the *PNPCore*. |

The alert() method executes in the context of the caller's thread. It performs its intended function without the use of blocking I/O operations and as such it will not cause the caller's thread to block for an extended period of time.

The *PNPCore* instantiates a single *CommWorker* object to accept and maintain the communication channel with the PNP. For this purpose, the PNP currently includes a single implementation of the *CommWorker* called the *JGroupsWorker*, which is based on the JGroups communication framework [JGROUPS]. The *PNPCore* also instantiates a distinct *CommWorker* object to interact with each remote PNP device. For this purpose, the PNP currently includes a single implementation of the *CommWorker* called the *JGroupsPeerWorker*, which also is based on the JGroups communication framework [JGROUPS]. In the future, different *CommWorker* implementations (i.e. Transport Layer Security, Session Initiation Protocol) can be added to the system.

The *CommWorker* interface requires that the implementing class provide the following public methods:

| | |
|---|---|
| isConnected() | Determine if the *CommWorker* is connected to the remote system – whether the local PDP device or a remote PNP device. |
| submit() | Submit a PNU from the *PNPCore* to the *CommWorker*. |
| close() | Close the communication channel to the remote device. |

These methods execute in the context of the caller's thread. They perform their intended functions without the use of blocking I/O operations and as such they will not cause the caller's thread to block for an extended period of time.


**Figure 2 - PNP UML Diagram**

### 5.2.1 Requirements

The PNP must fulfill the following requirements:

1. Process requests from the PDP to engage in communication with a remote administrative domain.

2. Process requests from the PDP to disengage from a remote administrative domain.

3. Forward PNUs received from the PDP to the remote PNP.

4. Retransmit PNUs to the remote PNP that the remote PNP did not acknowledge in a timely fashion.

5. Forward PNUs received from the remote PNP to the PDP.

### 5.2.2 Design

The *main()* method of the *PNPCore* class creates a single instance of the *PNPCore* class. The lone *PNPCore* constructor requires the network identity of the PDP and a reference to the *Status Code Manager* as its arguments. The *PNPCore* creates a single instance of a *CommWorker*, which in the current system is a *JGroupsWorker*. The *CommWorker* interface extends the java.lang.Runnable interface, which causes an implementing class to run in its own thread. This *CommWorker*, the PDP *CommWorker*, facilitates the communication between the PNP and the PDP.

The *PNPCore* processes control PNUs from the PDP to engage and disengage from a remote administrative domain. The control PNU specifies the type of policy to be negotiated, provides the network information required by the PNP to communicate with the remote administrative domain, and assigns a connection identifier for use as a reference within subsequent data PNUs. The *PNPCore* uses this information to create an internal routing table.

The *PNPCore* creates a distinct instance of a *CommWorker*, which in the current system is a *JGroupsPeerWorker,* to interact with each distinct remote PNP device. The *CommWorker* interface extends the java.lang.Runnable interface, which causes an implementing class to run in its own thread. The *PNPCore* employs a distinct *CommWorker*, the peer *CommWorker*, to exchange PNUs for numerous types of policies with a distinct remote PNP device. The *PNPCore* creates a peer *CommWorker* when no peer *CommWorker* is present and destroys a peer *CommWorker* when no policies require negotiation with a remote PNP device.

The *PNPCore* forwards a data PNU to the responsible peer *CommWorker* using its internal routing table. When the *PNPCore* receives a data PNU from the PDP, it locates the instance of the peer *CommWorker* in its internal routing table and invokes the peer *CommWorker* submit() method to deliver the data PNU. When the *PNPCore* receives a data PNU from the remote PDP via a peer *CommWorker*, it locates the instance of the peer *CommWorker* in its internal routing table to ensure that the data PNU was received by a known source and invokes the PDP *CommWorker* submit() method to deliver the data PNU to the PDP.

The *PNPCore* acknowledges the receipt of data PNUs from the remote PDP. When the remote PNP does not acknowledge the receipt of a data PNU in a timely fashion, the *PNPCore* retransmits the pending data PNU. This retransmission occurs until the acknowledgement arrives or the PNU validity period expires[2]. The *PNPCore* processes a PNU carrying Policy Refresh objects in a slightly different fashion. The *PNPCore* periodically retransmits and acknowledges a PNU carrying Policy Refresh objects until validity period expires.

The PDP *CommWorker*, the *JGroupsWorker,* establishes and maintains the communication channel between the PNP and the PDP. The *JGroupsWorker* simply relays PNUs between the *PDP* and the *PNPCore*. It accepts outgoing PNUs from the PDP and delivers them to the *PNPCore*, and it accepts incoming PNUs from the *PNPCore* and delivers them to the PDP.

The peer *CommWorker*, the *JGroupsPeerWorker,* establishes and maintains the communication channel between the local PNP and a remote PNP device. The *JGroupsPeerWorker* simply relays PNUs between the remote PNP and the *PNPCore*. It accepts incoming PNUs from the remote PNP and delivers them to the *PNPCore,* and it accepts outgoing PNUs from the *PNPCore* and delivers them to the remote PNP.

[2] The PNU header contains the validity period.

#### 5.2.2.1 Synchronization

The *PNPCore* makes us of a *SynchronizedQueue*, which is automatically synchronized.

The *JGroupsWorker* class synchronizes access to its request queue where incoming PNUs are deposited by the *PNPCore* thread and retrieved by the *JGroupsWorker* thread.

The *JGroupsPeerWorker* class synchronizes access to its request queue where outgoing PNUs are deposited by the *PNPCore* thread and retrieved by the *JGroupsPeerWorker* thread.

#### 5.2.2.2 Exceptions

The *PNPCore* class does not throw any exceptions.

The *JGroupsWorker* class does not throw any exceptions.

The *JGroupsPeerWorker* class does not throw any exceptions.

# 6. References

[PBNM] "Policy Based Network Management – System Design Document", Version DRAFT 0.1, NRNS Incorporated, March 2005

[JGROUPS] http://www.jgroups.org

## DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

NRNS Incorporated
4043 Carling Avenue
Ottawa K2K 2A3

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Policy Negotiation Proxy (PNP) Software Design Document (U)

4. AUTHORS (Last name, first name, middle initial)

Spagnolo, J., Cayer D.

5. DATE OF PUBLICATION (month and year of publication of document)

March 2005

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

8

6b. NO. OF REFS (total cited in document)

2

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DRDC Ottawa/NIO Section
3701 Carling Avenue
Ottawa K1A 0Z4

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

15BF27

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

W7714-3-800/001/SV

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

DRDC Ottawa CR 2005-111

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

( x ) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Full Unlimited

13. ABSTRACT ( a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

(U) This report presents the software design for the Policy Negotiation Proxy (PNP) component of a policy-based network management (PBNM) system. The PNP component facilitates the exchange and negotiation of policies on behalf of the Policy Decision Point (PDP) component of a PBNM system.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Inter-domain security policy
Network Managament
Policy
Policy-based network management (PBNM)
Policy Decision Point (PDP)
Policy enforcement
Policy Enforcement Point (PEP)
Policy negotiation
Policy Negotiation Proxy (PNP)
Policy object
Policy Processing Unit (PPU)

**Defence R&D Canada**

Canada's leader in Defence
and National Security
Science and Technology

**R & D pour la défense Canada**

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale


**www.drdc-rddc.gc.ca**