



PathFinder Test Configurations and Best Practices for Mobile Money

Document: PF_MM_v1.3

October 2017

Security Classification Category (see next page)		
Restricted	Internal GSMA	
Restricted	Members	
Restricted	Associate Members	
Restricted	Public Release	



Restricted Information

Access to and distribution of this document is restricted to the persons listed under the heading Security Classification Category. This document is confidential to the Association and is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those listed under Security Classification Category without the prior written approval of the Association. The GSM Association (“Association”) makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Copyright Notice

Copyright © 2017 GSM Association

GSM and the GSM logo are the registered property of the GSM Association

Antitrust Notice

The information contain herein is in full compliance with the GSM Association’s Antitrust Compliance policy.

Product Release: 5.15.x

Document History

<i>Version</i>	<i>Date</i>	<i>Brief Description</i>	<i>Editor</i>
1.0	19 May 2017	Initial version	Neustar
1.1	14 April 2017	Added Caching mechanism and priority setting	Neustar
1.2	11 Sept 2017	Added AWS Openswan example configuration.	Neustar
1.3	11 Oct 2017	Updated Security Classification for public release.	Neustar



Table of Contents

PATHFINDER TEST CONFIGURATIONS AND BEST PRACTICES FOR MOBILE MONEY	1
DOCUMENT: PF_MM_V1.3	1
LIST OF TABLES	4
1 INTRODUCTION	5
1.1 AUDIENCE	6
1.2 ACRONYMS	6
1.3 DOCUMENT CONVENTIONS	7
1.4 RELATED DOCUMENTATION	8
2 QUERY INTERFACE OVERVIEW	9
2.1 QUERY INTERFACE TEST	10
3 ENUM CLIENT QUERY DESIGN	12
3.1 UDP TRANSPORT	12
3.2 TCP TRANSPORT	14
3.3 SSL TRANSPORT	16
3.4 MANAGING QUERY CONCURRENCY	18
3.4.1 <i>The Synchronous Query Design</i>	18
3.4.2 <i>The Asynchronous Query Design</i>	19
4 PROVISIONING INTERFACE API COMMANDS	21
4.1 DEFINEDNSPROFILE	21
4.1.1 <i>Create a Profile</i>	21
4.2 UPDATEDNSPROFILE	22
4.2.1 <i>Add URIs to a Profile</i>	22
4.3 QUERYDNSPROFILE	23
4.3.1 <i>Display Profile Information</i>	23
4.4 ACTIVATE	24
4.4.1 <i>Valid TNs</i>	24
4.5 DEACTIVATE	24
4.5.1 <i>TN Block</i>	24
4.6 CHANGETN	25
4.6.1 <i>Inactivate TN</i>	25
4.7 QUERYTN	26
4.7.1 <i>List Profiles Associated with TNs</i>	26
4.7.2 <i>List TNs in Profiles</i>	27
4.8 TRANSACTIONMODE INDICATOR	27
4.8.1 <i>Activate TNs Asynchronously</i>	28
4.9 GETTRANSACTIONSTATUS	28
4.9.1 <i>Display Status of Asynchronous Request</i>	28
4.9.2 <i>Unauthorized IP Address</i>	29
4.10 SET CACHE CONFIGURATION	29



4.11 SET PRIORITY.....30

5 CONNECTIVITY BEST PRACTICES32

5.1 CTE CONNECTIVITY OPTIONS FOR IPSEC VPN32

5.2 PRODUCTION CONNECTIVITY OPTIONS FOR IPSEC VPN33

5.3 PRODUCTION CONNECTIVITY OPTIONS FOR GRX/IPX VLAN.....35

List of Figures

FIGURE 1. PATHFINDER QUERY INTERFACE9

FIGURE 2: SAMPLE IPSEC VPN CONNECTIVITY DIAGRAM34

List of Tables

TABLE 1: ACRONYMS AND DEFINITIONS6

TABLE 2: DOCUMENT CONVENTIONS.....7

Appendix

AWS GUIDELINE AND LIBRESWAN CONFIGUTATION EXAMPLE.....37



Chapter 1

In This Chapter:

1	INTRODUCTION	5
1.1	AUDIENCE.....	6
1.2	ACRONYMS	6
1.3	DOCUMENT CONVENTIONS	7
1.4	RELATED DOCUMENTATION	8

1 Introduction

This document provides the information needed to understand and implement the PathFinder Query Interface (QI). The QI is the information exchange component of PathFinder. It handles real-time query/response processing when establishing communication sessions between two endpoints.

This document provides the information needed to understand and implement the PathFinder Query Interface. It includes the following:

- A description of the Customer Test Environment and Production Environment;
- A step-by-step view of how query data flows through the interface;
- An overview of the open source *dig* tool that lets you test and implement new instances of the interface;
- A description of how the Query Interface processes DNS ENUM queries;
- A series of examples and test scenarios for DNS ENUM queries that demonstrate various aspect of the interface.



1.1 Audience

This document is written for individuals who need to know the configuration and best practices details of the GSMA PathFinder Provisioning and Query Interface.

1.2 Acronyms

The following table provides a list of acronyms used in this document:

Table 1: Acronyms and Definitions

Acronym	Definition
API	Application Programming Interface
CTE	Customer Test Environment
dig	Domain Information Groper
dip	Query
DNS	Domain Name System
E2U	ENUM to URI
ENUM	Electronic Numbering Mapping
FQDN	Fully Qualified Domain Name
GSMA	Global System for Mobile communications Association
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
IPX	IP Packet Exchange
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
LRN	Location Routing Number
MCC	Mobile Country Code
MNC	Mobile Network Code
MMS	Multimedia Message Service



Acronym	Definition
NAPTR	Naming Authority Pointer
NPAC	Number Portability Administration Center
NPD	Number Portability Discovery
NPDI	Number Portability Dip Indicator
PI	Provisioning Interface
QI	Query Interface
RFC	Request for Comment
RR	Resource Record
SIP	Session Initiation Protocol
SMS	Short Message Service
SOA	Start of Authority
SPN	Service Provider Number
SSL	Secure Sockets Layer
TN	Telephone Number
URI	Uniform Resource Identifier


1.3 Document Conventions

This document uses the following conventions:

Table 2: Document Conventions

Convention	Description	Example
Italics	Indicates that a term or phrase is being introduced and that its definition is in the vicinity (either right before or right after).	The <i>engine</i> is the order management system.
Constant width	Used to indicate commands, file names, and file and code samples. Might be emphasized with bold.	<code>% xterm -sb -title osagent</code>
Italics, Constant width	Used within command, code, and file samples. Indicates file names or text that should be replaced with words or names that	<code>% perl copy2web.prl <directory></code>



Convention	Description	Example
	are appropriate to the customer's installation or environment. Might be emphasized with bold.	<i>AdminPhn=Administrator's phone number</i>
< >	Encloses a directory, file name or other information that will need to be replaced. The actual name should not be enclosed within angle brackets.	D:\<installation root directory>\
Hypertext link	Used to indicate a hypertext link that, if clicked, will take the user to either an HTML page or a URL. A default browser must be specified.	You can find some examples in the use cases at http://www.gsmworld.com/oneapi
Notations	A note symbol is used to provide supporting information that may not be explicitly addressed in the accompanying text.	 NOTE This symbol indicates supporting information.
XML Notation	It is Neustar standard to represent data values via Node attributes named "value".	<DSENT value="11-03-2003-0900AM"/>

1.4 Related Documentation

The following are PathFinder-related documents:

- **PATHFINDER ADMIN AND CUSTOMER USER GUIDES:** provides step-by-step procedures for tasks performed using the PathFinder GUI for Neustar Administrators and Customers;
- **PATHFINDER PROVISIONING INTERFACE GUIDE:** describes the structure and available commands;
- **PATHFINDER QUERY INTERFACE GUIDE:** This Guide which describes the PathFinder environments and how the Query Interface processes DNS ENUM queries;
- **PATHFINDER CONNECTIVITY AND DEPLOYMENT GUIDE:** This Guide summarizes the various options available for connectivity in the customer test (CTE) and production environments for a customer who has subscribed to a PathFinder Query Interface (QI) service, and to provide the sequence for deployment. It also discusses the availability of a Neustar-designed reference Java client to aid customer QI API development.

Chapter 2

In This Chapter:

2 QUERY INTERFACE OVERVIEW..... 9

 2.1 QUERY INTERFACE TEST..... 10

2 Query Interface Overview

The Query Interface supports standard DNS Electronic Number Mapping System (ENUM) per IETF RFC 6116 (formerly RFC 3761). PathFinder appears as a DNS server in the customer’s ENUM topology. The figure below illustrates the Query Interface:

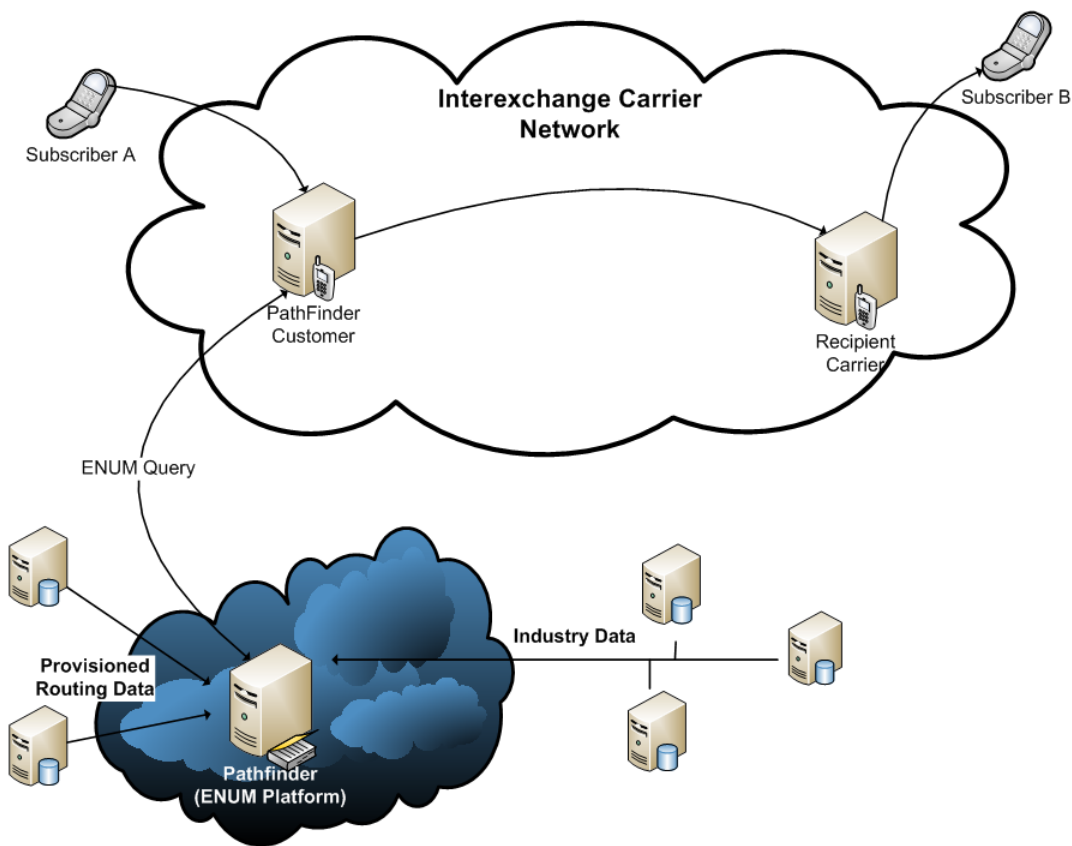


Figure 1. PathFinder Query Interface

The Query Interface is supported in the production environment as well as the Customer Test Environment (CTE) for testing and integration activities. Commercial PathFinder operations occur in the Neustar production environment. Pre-production efforts, such as testing and integration, are conducted in the CTE, which mimics the production environment.

The production environment and the CTE are kept isolated from each other. This separation preserves the performance and stability of production activities. Once a business relationship is



established, Neustar creates a profile for the customer that includes at least one customer IP address authorized to access the Query Interface. (Only IP addresses stored in the profile may connect to the Query Interface). After the profile has been created, Neustar provides both the CTE and production environment IP addresses to the customer.

Note: Customers who meet certain prerequisites can also consider a Secure Sockets Layer (SSL)-based connection method which doesn't require identifying IP addresses. Please refer to the *PathFinder Connectivity and Deployment Guide* for additional information.

New PathFinder customers perform pre-production testing and integration in the CTE. When they are ready to move to production, Neustar provides guidance to ensure a smooth transition. Existing PathFinder customers can use the CTE as a sandbox to conduct ongoing tests, trials and integration checks.

2.1 Query Interface Test

Query Interface testing lets PathFinder customers confirm that the querying host receives NAPTR RRs with the appropriate routing information and that these can be processed, as desired, within their own environments. The following sections list test examples that cover a representative range of the type of results that can be expected.

Successful Query with Multiple Results

TN	15714345785
Query Type	NAPTR
ENUM Root	e164enum.net

Purpose	To verify the ability to handle multiple NAPTRs. Not applicable for just PathFinder NPD Service.
Description	PathFinder receives a request to find matching answers for a valid TN. PathFinder sends multiple URI NAPTRs in the response. Customer should be able to parse all the NAPTRs without any errors.
Expected Behavior	PathFinder receives a request to a valid TN which has multiple NAPTRs. Customer should be able to handle multiple NAPTRs without any errors.
Preconditions	The originating IP address has been associated with a valid PathFinder customer. The TN in the request is a valid E.164 number. The ENUM root or the terminating domain in the query string is valid.
ENUM Query	Type=NAPTR Class=IN 5.8.7.5.4.3.4.1.7.5.1.e164enum.net



ENUM Answer	<pre>; <<>> DiG 9.2.4 <<>> @pathfinder-cte-qi.neustar.biz. 5.8.7.5.4.3.4.1.7.5.1.10025.e164enum.net in NAPTR ;; global options: printcmd ;; Got answer: ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50641 ;; flags: qr aa rd; QUERY: 1, ANSWER: 7, AUTHORITY: 2, ADDITIONAL: 0 ;; QUESTION SECTION: ;5.8.7.5.4.3.4.1.7.5.1.e164enum.net. IN NAPTR ;; ANSWER SECTION: 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 1 "u" "E2U+h323" "!^(.*)\$!h323:\\1@test1.neustarlab.biz!" . 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 1 "u" "E2U+im" "!^(.*)\$!im:\\1@test2.neustarlab.biz!" . 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 1 "u" "E2U+mailto" "!^(.*)\$!mailto:\\1@test3.neustarlab.biz!" . 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 1 "u" "E2U+pres" "!^(.*)\$!pres:\\1@test4.neustarlab.biz!" . 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 1 "u" "E2U+sip" "!^(.*)\$!sip:\\1@test5.neustarlab.biz!" . 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 1 "u" "E2U+sms" "!^(.*)\$!sms:\\1@test6.neustarlab.biz!" . 5.8.7.5.4.3.4.1.7.5.1.e164enum.net. 900 IN NAPTR 10 50 "u" "E2U+pstn:tel" "!^(.*)\$!tel:\\1\\;npdi\\;spn=7966!" . ;; AUTHORITY SECTION: 4.3.4.1.7.5.1.e164enum.net. 900 IN NS peabody.neustarlab.biz. 4.3.4.1.7.5.1.e164enum.net. 900 IN NS ns1.net.gprs. ;; Query time: 16 msec ;; SERVER: 10.41.5.142#53(lab2app3.neustarlab.biz) ;; WHEN: Tue Jul 7 19:37:22 2009 ;; MSG SIZE rcvd: 568</pre>
-------------	---



Chapter 3

In This Chapter:

3	ENUM CLIENT QUERY DESIGN	12
3.1	UDP TRANSPORT.....	12
3.2	TCP TRANSPORT.....	14
3.3	SSL TRANSPORT	16
3.4	MANAGING QUERY CONCURRENCY	18
3.4.1	<i>The Synchronous Query Design.....</i>	<i>18</i>
3.4.2	<i>The Asynchronous Query Design.....</i>	<i>19</i>

3 ENUM Client Query Design

Client design is critical to optimizing value from the PF service. To this end, Neustar provides a sample JAVA Client reference implementation (described in Chapter 5) that demonstrates how to issue an ENUM query using the UDP, TCP, and SSL transports.

Pathfinder service supports the DNS ENUM interface to enable the client application to send a TN query and receive the route information back in the response. ENUM is a standards-based query mechanism (RFC 6116) that uses DNS as its underlying protocol. Users have a choice to transmit TN queries using one of three transports: UDP, TCP, or SSL (with mandatory mutual authentication using a Public Key Infrastructure). The choice of UDP or TCP only applies to IPsec VPN connectivity. Conversely, TCP is only applicable if the Customer chooses to connect to PF using the SSL binding.

UDP is the preferred transport protocol for a PF query. TCP falls next in line. If UDP or TCP transport over IPsec VPN connectivity is not applicable, then the SSL query option is the fallback option.

3.1 UDP Transport

A TN query over a UDP transport is the least resource-intensive method to query the service. By design, UDP is a connectionless protocol. The reliability aspect of the UDP transport is less of a concern since the data packets are flowing through the managed IPsec VPN environment.

In order to make a TN query, the client application creates the required socket binding and sends the request message to PF. After processing the request, PF returns the response to the waiting client application.

In Figure 3 below, it is assumed that the client application is making a synchronous query, such that the client application will wait for the response to arrive before sending a new TN request using the same socket binding. This is the recommended approach.

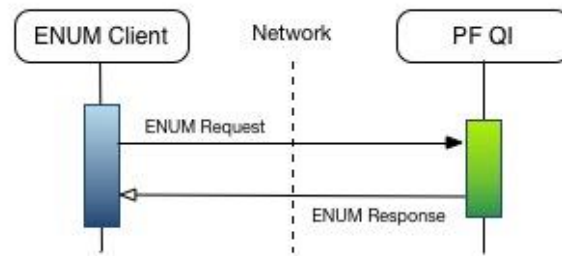


Figure 3: UDP Transport Flow Diagram

To process multiple TN lookups concurrently, the client application can utilize multiple UDP sockets, each processing a single synchronous TN request/response.

Further, in order to avoid potential issues with the PF firewall (FW) session-based configuration, it is recommended that the client application acquires a new source port each time a query is ready to be sent to PF. As an example, the pseudo-code of the operation to perform a PF TN query in the client application would be as shown below:

```

Result performTnQuery()
{
    // create a UDP socket for network communication

    // bind the socket to the local network

    // send the query to the remote PF server

    // receive the response from the remote PF server

    // close the socket
}
  
```

In some environments, if the source port is re-used for UDP transport to issue subsequent queries, the FW and/or load balancer (LB) creates a time-based session association between the originating source IP:Port and the terminating PF server IP:Port in order to facilitate the response from the

server back to the correct origination. The defaults for such time-based session association for UDP transport have been found to be typically 4 seconds. Once a response from the server is returned to the source, the session is “destroyed”. This may impact the subsequent TN query from the source in flight. Further, once the session is destroyed, subsequent responses from the PF server for the same originating Source:IP may get dropped. The latter would be a concern if the client application is attempting to send back-to-back TN queries over the same source IP:Port and hoping to receive the responses in asynchronous fashion. In order to avoid this situation, the client application should use a different source port for each new query.

3.2 TCP Transport

TCP transport provides an alternative approach to query PF. TCP is a connection-oriented protocol and it is designed to provide guarantees of packet delivery to both ends of the connection between the client application and PF.

In Figure 4 below, when the client application is ready to make a TN query, besides creating the socket binding, an explicit TCP connect method is called to make sure that the server reserves a unique “handle” to communicate with the client.

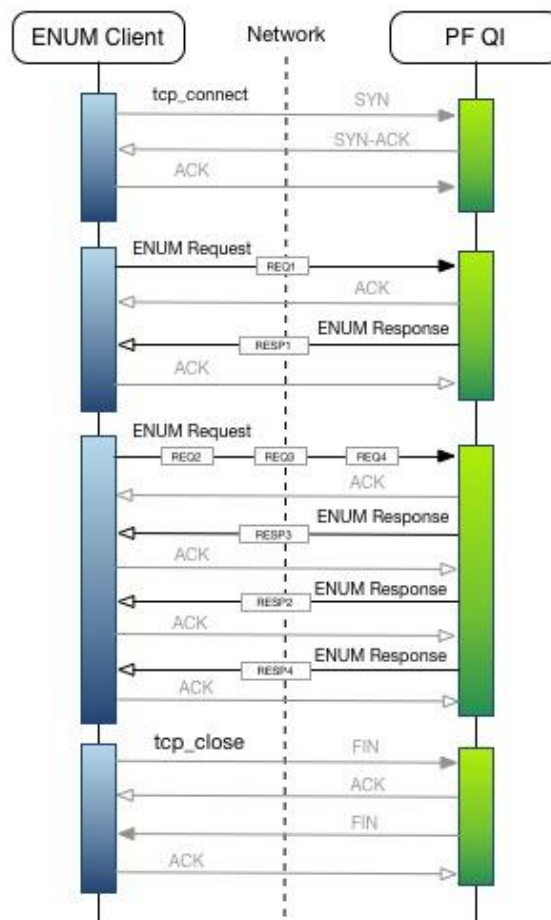


Figure 4: TCP Transport Flow Diagram

Next, the client application sends the TN query request to PF. After processing the request, PF returns the response back to the waiting client application. If there are no more TN lookups to process, the client application may choose to close the TCP connection, signaling the end of the communication to the PF server. In addition to the active client application initiated actions, Figure 4 also depicts the underlying TCP-level communication that makes for the added reliability aspect of TCP transport.

In the flow diagram, it is assumed that the client application is making a synchronous TN query. In this case, the client application has the single TN as the query context and the response received from PF will relate to it. The diagram denotes this request as REQ1 and the corresponding response as RESP1.

Also, unlike the UDP transport, since TCP is a connection-oriented protocol, there is an opportunity to send back-to-back TN queries to the PF server for processing over the same TCP connection and the response can be received asynchronously from the PF server. The client application will have to actively maintain the identifying information from the multiple requests that were sent to the PF server application until a response is received. In the case of ENUM, the query string - composed of an application unique string (the TN) and the terminating domain - often suffices as the identifying information used for mapping the request and the corresponding response.

If available to the client application code, the MsgID attribute of the DNS packet, in addition to the query string, makes for even stronger unique identifying information to map the related request and response combinations. In Figure 4 above, the REQ1, REQ2, and REQ3 are buffered requests that are transmitted as a single request. After processing the requests, the PF server returns the corresponding response messages back to the client application. Please note, in order to return the responses back to the waiting client application as quickly as possible, the responses may not follow the same sequence as the requests sent earlier.

Establishing a TCP connection is both resource-intensive and consumes time that could be spent for making actual TN queries. Therefore, it is imperative that, once established, the client application re-uses the TCP connection for multiple query lookups. In case of a network error, or an inactivity timeout detected by PF, standard TCP semantics will help discover the error and the client application will have an opportunity to re-create the connection before sending the new TN query.

In case of persistent TCP connections, the PF server implements a mechanism to detect idle connections and disconnect them after a period of inactivity. It is recommended that the client application sends a dummy query every 45 seconds as a “keepalive” indicator in order to keep the PF server from closing the TCP connection. The supported PF dummy query needs to be constructed



with the query string as the valid terminating domain associated with the client profile. An example of a valid query string is “e164enum.net”. Use of the supported PF “keepalive” query ensures that such queries are not billed.

The PF environment facilitates access to many customers in a manner that guarantees high availability and low latency response times. Each Customer is restricted to a finite number of TCP connections. The request-per-connection strategy doesn’t scale and it is highly discouraged. Also, a strategy to re-use the TCP connection through the use of a resource pool, with a well-defined maximum resource pool limit, is encouraged.

3.3 SSL Transport

PathFinder has native support for SSL PKI connectivity. Use of SSL enables confidentiality, data integrity, and authentication. SSL PKI connectivity is applicable in situations when the IPsec VPN connectivity is not an option for the Customer.

Mutual authentication is the only supported SSL option in the PF environment. We demonstrate this in Figure 5 below.

First, in response to the client application SSL connection request, the PF SSL server asks for the client application certificate. If the client certificate is in the PF truststore, then the SSL connection is established.

Next, the PF server performs the authorization check by testing the client certificate attributes against the Customer profile table. An authorization failure will result in PF closing the SSL connection.

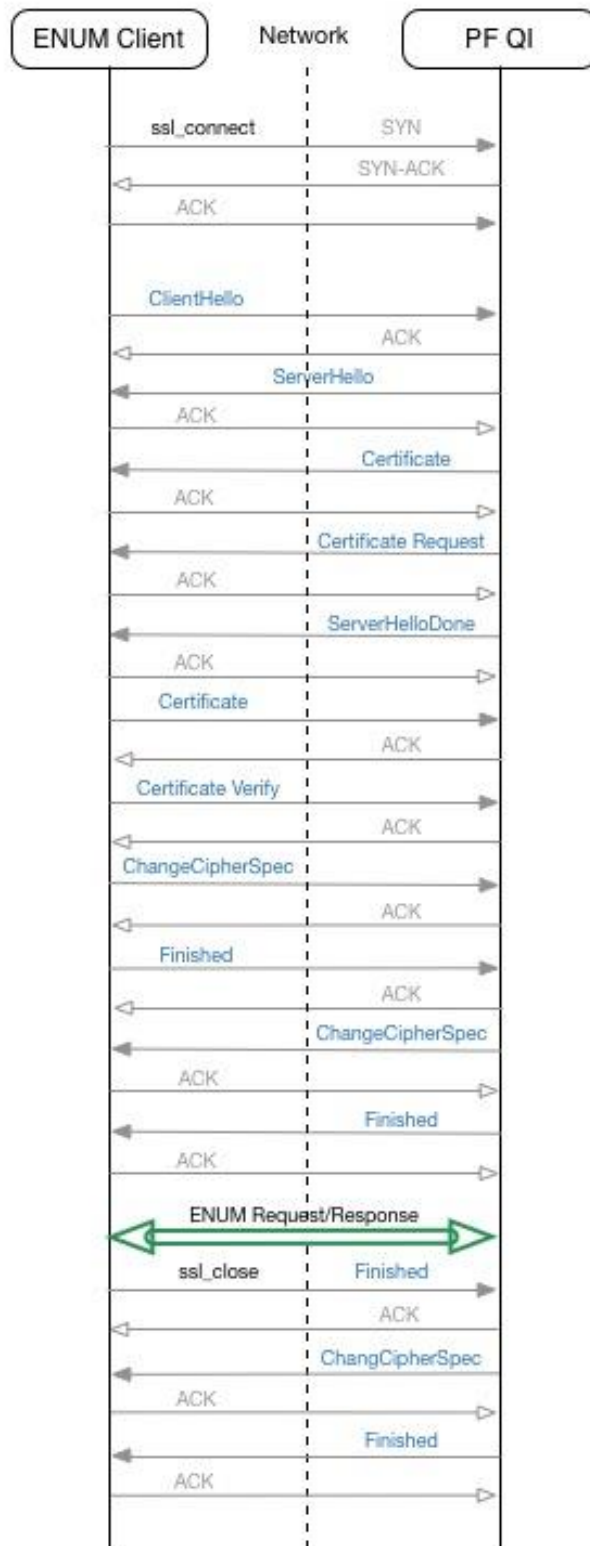


Figure 5: SSL Transport Flow Diagram

SSL is enabled over TCP, therefore, all the TCP-related aspects described in section 2.2 above apply equally to the SSL transport option.

3.4 Managing Query Concurrency

In the early testing phase, it is fine to have the client application create a new socket binding, make a TN query, and then close the socket. If the expectation is for the client application to be able to process a high query per second throughput, then a more improved strategy is required.

Since both the threads and file/socket handles are finite resources, invoking a new thread-per-connection model doesn't scale with an increasing query rate. Further, the number of CPUs on the host machine, as well as the network bandwidth, is also limiting factors that will cause the peak query rate to plateau at some point. Use of more machine resources than a certain "sweet spot" may not yield higher query throughput and a variety of errors may begin to surface.

Throughout the following sub-sections several design strategies are shared to help Customers achieve both higher query throughput and efficient use of resources. These design strategies are agnostic to the supported protocols: UDP, TCP, and SSL.

3.4.1 The Synchronous Query Design

In this method, as illustrated in Figure 6 below, the business domain makes *synchronous* calls to the Request Processor, which in turn relies on asynchronous request processing to resolve the TN query. Concurrent Queue helps with the messaging, such that, after pushing the request onto the queue, a "notify" method is used to wake up one of the Worker Threads.

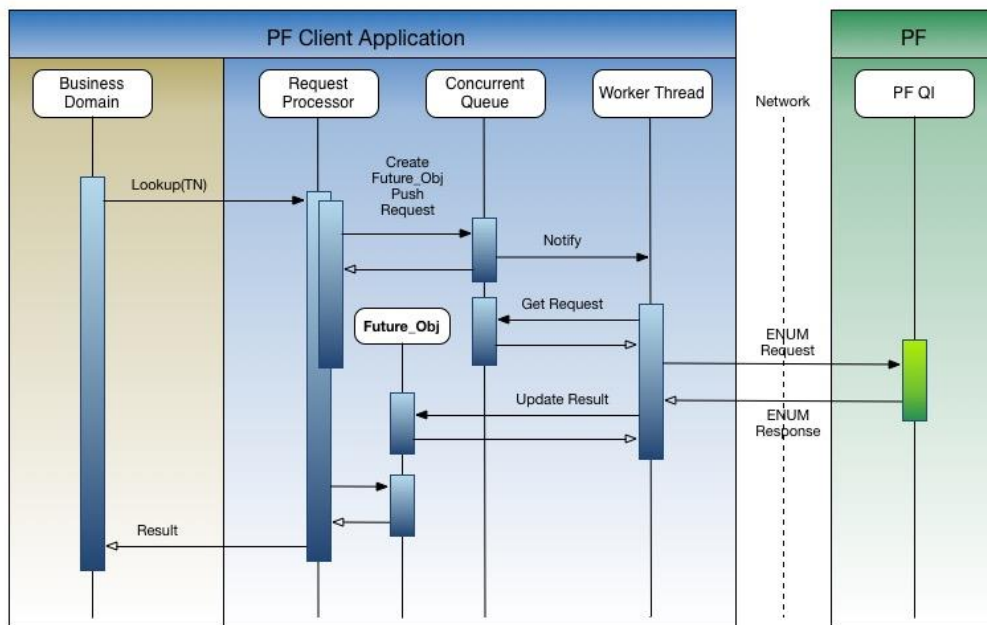


Figure 6: The Synchronous Query Flow Diagram

Next, the Worker fetches the waiting TN request from the Concurrent Queue, makes the ENUM query, and posts the response to the Future_Obj. The Request Processor discovers the result of the query and returns it to the waiting Business Domain.

In this case, each Worker Thread can have a one-to-one association with the socket handle. Further, a configurable pool of Worker Threads can be used to limit the allowable maximum.

In addition to the TN information, the Business Domain can provide a response timeout value to the Request Processor in order to keep from waiting too long for the response from PF. Further, a configurable request retry count attribute can allow the Worker Thread to retry the query against alternative PF sites in case of an error, before returning a query failure.

3.4.2 The Asynchronous Query Design

Figure 7 below illustrates the Asynchronous Query method. In this method, the Business Domain makes an *asynchronous* call to the Request Processor, which returns a Future_Obj for discovering updates at a later time. The remaining functionality of the Request Processor is unchanged from what was discussed in section 2.4.1 above.

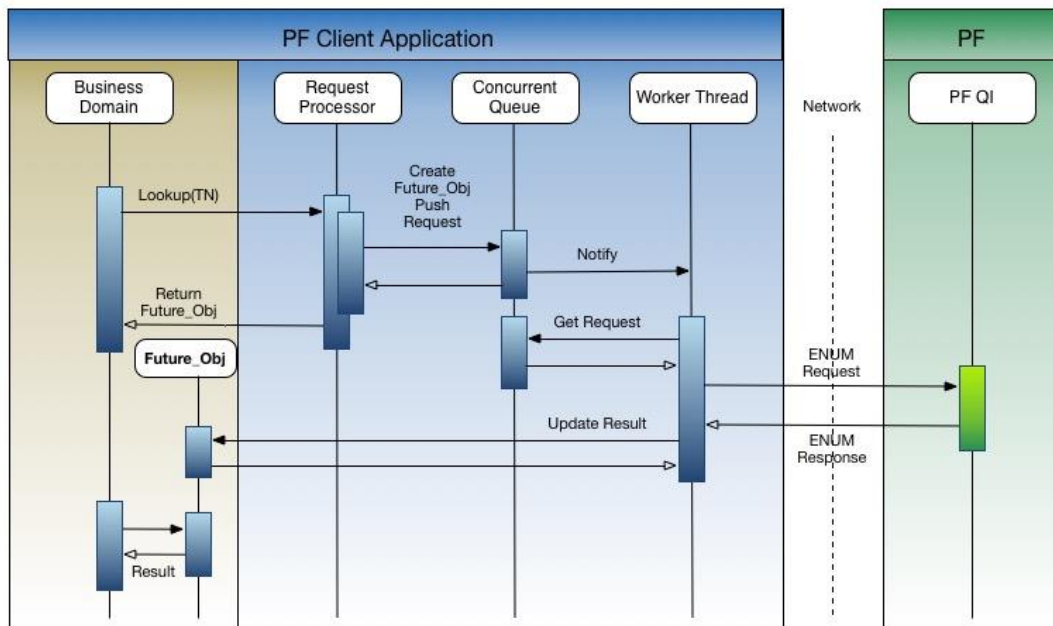


Figure 7: The Asynchronous Query Flow Diagram



Chapter 4

In This Chapter:


4	PROVISIONING INTERFACE API COMMANDS	21
4.1	DEFINEDNSPROFILE	21
4.1.1	<i>Create a Profile</i>	21
4.2	UPDATEDNSPROFILE	22
4.2.1	<i>Add URIs to a Profile</i>	22
4.3	QUERYDNSPROFILE	23
4.3.1	<i>Display Profile Information</i>	23
4.4	ACTIVATE	24
4.4.1	<i>Valid TNs</i>	24
4.5	DEACTIVATE	24
4.5.1	<i>TN Block</i>	24
4.6	CHANGETN	25
4.6.1	<i>Inactivate TN</i>	25
4.7	QUERYTN	26
4.7.1	<i>List Profiles Associated with TNs</i>	26
4.7.2	<i>List TNs in Profiles</i>	27
4.8	TRANSACTIONMODE INDICATOR	27
4.8.1	<i>Activate TNs Asynchronously</i>	28
4.9	GETTRANSACTIONSTATUS	28
4.9.1	<i>Display Status of Asynchronous Request</i>	28
4.9.2	<i>Unauthorized IP Address</i>	29
4.10	SET CACHE CONFIGURATION	29
4.11	SET PRIORITY	30



4 Provisioning Interface API Commands

4.1 DefineDNSProfile

Use the DefineDNSProfile command to create a Profile.

 **Note:** PathFinder lets you specify additional URI Schemes/Services and Attributes in URIs for Profiles. In the PathFinder GUI, the available URI Schemes/Services are shown on the **NAPTR** tab on the **Manage Tier 2 Profile** page. The available Attributes are shown on the **NAPTR Attributes** tab. To request that additional URI Schemes/Services or Attributes be added to PathFinder, contact *Customer Support* (on page **Error! Bookmark not defined.**).

4.1.1 Create a Profile

In the following example, the DefineDNSProfile request includes valid URIs:

Description	Define a Profile with two valid URIs.
Expected Result	Return code 201
SOAP Request	<pre><BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <DefinedDNSProfile> <TransactionID>1209553743270</TransactionID> <ProfileID>DNSProfW2NAPTRs</ProfileID> <Tier>2</Tier> <NAPTR ttl="900"> <DomainName>e164enum.net</DomainName> <Preference>1</Preference> <Order>10</Order> <Flags>u</Flags> <Service>E2U+sip</Service> <Regex pattern="^(.*)\$">sip:\1@iotlab1.com</Regex> <Replacement>.</Replacement> <Partner id="-1">ALL</Partner> </NAPTR> <NAPTR ttl="900"> <DomainName>e164enum.net</DomainName> <Preference>2</Preference> <Order>10</Order> <Flags>u</Flags> <Service>E2U+sip</Service> <Regex pattern="^(.*)\$">sip:\1@iotlab2.com</Regex> <Replacement>.</Replacement> <Partner id="-1">ALL</Partner> </NAPTR> </DefinedDNSProfile> </Request> </BatchRequest></pre>
SOAP Response	<pre><BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209553743270</TransactionID> <ReturnCode>201</ReturnCode></pre>



	<pre> <TextMessage>Created</TextMessage> <TextMessage>Profile DNSProfW2NAPTRs created successfully</TextMessage> <TextMessage>Date: Wed Apr 30 07:09:10 EDT 2010</TextMessage> </Response> </BatchResponse> </pre>
--	--

4.2 UpdateDNSProfile

Use the UpdateDNSProfile command to update the URIs in a Profile.

4.2.1 Add URIs to a Profile

In the following example, the UpdateDNSProfile request includes two URIs to be added to a Profile:

Description	Update a Profile with two valid URIs.
Expected Result	Return code 200
SOAP Request	<pre> <BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <UpdateDNSProfile> <TransactionID>1209550343658</TransactionID> <ProfileID>T2Profile2NAPTRs</ProfileID> <NAPTR> <DomainName>abc123.com</DomainName> <Preference>1</Preference> <Order>10</Order> <Flags>u</Flags> <Service>E2U+sip</Service> <Regexp pattern="^(.*)\$">sip:\1@T2Profile1.com </Regexp> <Replacement>.</Replacement> <Partner id="-1">ALL</Partner> </NAPTR> <NAPTR> <DomainName>xyz123.com</DomainName> <Preference>2</Preference> <Order>10</Order> <Flags>u</Flags> <Service>E2U+sip</Service> <Regexp pattern="^(.*)\$">sip:\1@T2Profile2.com </Regexp> <Replacement>.</Replacement> <Partner id="-1">ALL</Partner> </NAPTR> </UpdateDNSProfile> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209550343658</TransactionID> <ReturnCode>200</ReturnCode> <TextMessage>OK</TextMessage> </pre>



	<pre> <TextMessage>Profile DNSProfileForTier2 updated successfully</TextMessage> <TextMessage>Date: Wed Apr 30 06:12:30 EDT 2010</TextMessage> </Response> </BatchResponse> </pre>
--	--

4.3 QueryDNSProfile

Use the QueryDNS command to get a list of URIs included in a Profile.

4.3.1 Display Profile Information

In the following example, the QueryDNSProfile request specifies a valid Profile ID:

Description	Query a Profile to get list of URIs.
Expected Result	Return code 200
SOAP Request	<pre> <BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <QueryDNSProfile> <TransactionID>100001209557062141</TransactionID> <ProfileID>DNSProfW2NAPTRs</ProfileID> </QueryDNSProfile> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>100001209557062141</TransactionID> <ReturnCode>200</ReturnCode> <TextMessage>OK</TextMessage> <TextMessage>DNS profile queried successfully</TextMessage> <TextMessage>Date: Wed Apr 30 08:04:22 EDT 2010</TextMessage> <ResponseData> <DNSProfileData> <ProfileID> DNSProfW2NAPTRs </ProfileID> <Customer id="10000"/> <DateCreated>2010-01-28T20:33:37.641Z</DateCreated> <IsInUse>>false</IsInUse> <Tier>2</Tier> <NAPTR ttl="900"> <DomainName>e164enum.net</DomainName> <Preference>1</Preference> <Order>10</Order> <Flags>u</Flags> <Service>E2U+sip</Service> <Regexp pattern="^(.*)\$">sip:\1@iotlab1.com </Regexp> <Replacement>.</Replacement> <CountryCode>>false</CountryCode> <Partner id="10000"/> </NAPTR> </pre>



	<pre> </DNSProfileData> </ResponseData> </Response> </BatchResponse> </pre>
--	---

4.4 Activate

Use the Activate command to change the status of TNs in a Profile to active.

4.4.1 Valid TNs

In the following example, the Activate request includes a TN block that is associated with a Profile:

Description	Activate valid TNs with a valid Profile ID.
Expected Result	Return code 201
SOAP Request	<pre> <BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <Activate> <TransactionID>1209553748889</TransactionID> <TN> <Base>5679401000</Base> <Stop>5679401009</Stop> <CountryCode>1</CountryCode> </TN> <Status>active</Status> <DNSProfileID>DNSProfW2NAPTRs</DNSProfileID> <Tier>2</Tier> </Activate> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209553748889</TransactionID> <ReturnCode>201</ReturnCode> <TextMessage>Created</TextMessage> <TextMessage>TN activated successfully</TextMessage> <TextMessage>Date: Wed Apr 30 07:09:16 EDT 2010</TextMessage> </Response> </BatchResponse> </pre>

4.5 Deactivate

Use the Deactivate command to inactivate previously activated TNs.

4.5.1 TN Block

In the following example, the Deactivate request includes two TNs to be removed:

Description	Deactivate a valid TN block.
Expected Result	Return code 200



SOAP Request	<pre><BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <Deactivate> <TransactionID>1209558306031</TransactionID> <TN> <Base>5679401008</Base> <Stop>5679401009</Stop> <CountryCode>1</CountryCode> </TN> <Tier>2</Tier> </Deactivate> </Request> </BatchRequest></pre>
SOAP Response	<pre><BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209558306031</TransactionID> <ReturnCode>200</ReturnCode> <TextMessage>OK</TextMessage> <TextMessage>TNs deactivated successfully</TextMessage> <TextMessage>Date: Wed Apr 30 08:25:12 EDT 2010</TextMessage> </Response> </BatchResponse></pre>

4.6 ChangeTN

Use the ChangeTN command to change the status of a TN or its associated Profile.

4.6.1 Inactivate TN

In the following example, the ChangeTN request is to change the status of a TN to inactive:

Description	Change a TN or TN block status to inactive.
Expected Result	Return code 200
SOAP Request	<pre><BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <ChangeTN> <TransactionID>1209561222734</TransactionID> <TN> <Base>5679401020</Base> <Size>1</Size> <CountryCode>1</CountryCode> </TN> <Status>inactive</Status> <DNSProfileID>DNSProfW2NAPTRs</DNSProfileID> </ChangeTN> </Request> </BatchRequest></pre>
SOAP Response	<pre><BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209561222734</TransactionID> <ReturnCode>200</ReturnCode></pre>



	<pre> <TextMessage>OK</TextMessage> <TextMessage>TN profile updated successfully</TextMessage> <TextMessage>Date: Wed Apr 30 09:13:49 EDT 2010</TextMessage> </Response> </BatchResponse> </pre>
--	--

4.7 QueryTN

Use the QueryTN command to obtain a list of the Profiles that a TN is associated with or a list of the TNs that are associated with a Profile. The QueryTN command is helpful in verifying that a TN is associated with the correct Profile and that a Profile has the correct TNs associated with it.

4.7.1 List Profiles Associated with TNs

In the following example, the QueryTN request includes a TN and the response lists the Profile the TN is associated with:

Description	Query a TN or TN block that belongs to the customer.
Expected Result	Return code 200
SOAP Request	<pre> <BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <QueryTN> <TransactionID>1209555371693</TransactionID> <TN> <Base>5714341000</Base> <CountryCode>1</CountryCode> </TN> <Tier>2</Tier> </QueryTN> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209555371693</TransactionID> <ReturnCode>200</ReturnCode> <TextMessage>OK</TextMessage> <TextMessage>1 TN profile is queried successfully</TextMessage> <TextMessage>Date: Wed Apr 30 07:36:18 EDT 2010</TextMessage> <ResponseData> <TNData> <TN> <Base>5714341000</Base> <CountryCode>1</CountryCode> </TN> <Customer id="10000"/> <DateCreated>2010-03-13T09:46:33.395Z</DateCreated> <Status>active</Status> <DNSProfileID>DNSProfileId: DNSProfW2NAPTRs </DNSProfileID> </pre>



	<pre> <Tier>2</Tier> </TNData> </ResponseData> </Response> </BatchResponse> </pre>
--	--

4.7.2 List TNs in Profiles

In the following example, the QueryTN request includes a Profile ID and the response lists the TNs associated with the Profile:

Description	Query TNs associated with a valid Profile ID.
Expected Result	Return code 200
SOAP Request	<pre> <BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <QueryTN> <TransactionID>1209557071297</TransactionID> <DNSProfileID>DNSProfW2NAPTRs</DNSProfileID> </QueryTN> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209557071297</TransactionID> <ReturnCode>200</ReturnCode> <TextMessage>OK</TextMessage> <TextMessage>5 TN profiles are queried successfully</TextMessage> <TextMessage>Date: Wed Apr 30 08:04:38 EDT 2010</TextMessage> <ResponseData> <TNData> <TN> <Base>5714341020</Base> <Size>5</Size> <CountryCode>1</CountryCode> </TN> <Customer id="10000"/> <DateCreated>2010-04-30T16:01:28.687Z</DateCreated> <Status>active</Status> <DNSProfileID>DNSProfW2NAPTRs</DNSProfileID> <Tier>2</Tier> </TNData> </ResponseData> </Response> </BatchResponse> </pre>

4.8 TransactionMode Indicator

Use the **async** value in the TransactionMode tag to indicate that the transaction should be run asynchronously. By default, transactions are run synchronously.



4.8.1 Activate TNs Asynchronously

In the following example, the request is to activate TNs asynchronously:

Description	Activate TNs in async mode.
Expected Result	Return code 202
SOAP Request	<pre><BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <Activate> <TransactionID>1209557710725642000</TransactionID> <TN> <Base>5679403000</Base> <Stop>5679403004</Stop> <CountryCode>1</CountryCode> </TN> <Status>active</Status> <DNSProfileID>DNSProfW2NAPTRs</DNSProfileID> <Tier>2</Tier> <TransactionMode>async</TransactionMode> </Activate> </Request> </BatchRequest></pre>
SOAP Response	<pre><BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209557710725642000</TransactionID> <ReturnCode>202</ReturnCode> <TextMessage>Accepted</TextMessage> <TextMessage>TN activation request is accepted and pending</TextMessage> <TextMessage>Date: Wed Apr 30 08:15:10 EDT 2010</TextMessage> </Response> </BatchResponse></pre>

4.9 GetTransactionStatus

Use the GetTransactionStatus command to determine the progress of a transaction running in asynchronous mode.

4.9.1 Display Status of Asynchronous Request

In the following example, the GetTransactionStatus request includes the transaction ID of an ActivateTN request:

Description	Check the status of TNs being activated in async mode.
Expected Result	Return code 201
SOAP Request	<pre><BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <GetTransactionStatus> <TransactionID>1209557710725642000</TransactionID> </GetTransactionStatus> </Request> </BatchRequest></pre>



	<pre> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209557710725642000</TransactionID> <ReturnCode>201</ReturnCode> <TextMessage>Created</TextMessage> <TextMessage>TN activated successfully</TextMessage> <TextMessage>Request Type: Activate</TextMessage> <TextMessage>Request Time: Wed Apr 30 08:15:05 EDT 2010</TextMessage> <TextMessage>Process Time: 1380 ms</TextMessage> <TextMessage>TNs Affected: 5</TextMessage> <TextMessage>Completed(%): 100</TextMessage> <TextMessage>Date: Wed Apr 30 08:15:17 EDT 2010</TextMessage> </Response> </BatchResponse> </pre>

4.9.2 Unauthorized IP Address

In the following example, the GetTransactionStatus request is sent from an IP address that is not authorized to access PathFinder:

Description	Send a GetTransactionStatus request from an unauthorized IP address.
Expected Result	Return code 401
SOAP Request	<pre> <BatchRequest xmlns="http://www.neustar.biz/sip_ix/prov"> <Request> <GetTransactionStatus> <TransactionID>1209563015367309000</TransactionID> </GetTransactionStatus> </Request> </BatchRequest> </pre>
SOAP Response	<pre> <BatchResponse xmlns="http://www.neustar.biz/sip_ix/prov"> <Response> <TransactionID>1209563015367309000</TransactionID> <ReturnCode>401</ReturnCode> <TextMessage>Unauthorized</TextMessage> <TextMessage>Not authorized from IP address 203.199.18.91</TextMessage> <TextMessage>Date: Wed Apr 30 09:44:06 EDT 2010</TextMessage> </Response> </BatchResponse> </pre>

4.10 Set Cache Configuration

The API queries from section 4.1 through section 4.9 are executed on the Provisioning Interface (PI). As part of a distributed architecture, the PathFinder PI pushes the data to the Query Interface (QI) nodes which reside on the periphery of the network using 3rd party propagation tools. This



propagation can take anywhere from a few minutes to 6 hours. To ensure that user validation happens correctly and no data is lost, Neustar recommends caching the provisioned information for a Time-To-Live (TTL) period of 6 hours. This should be configured so as to change the depth of the cache life. At the end of this configured period of the cache the system shall refer to Pathfinder for any further data validations.

4.11 Set Priority

The API's for creating a profile and to add URI's to the profile have a parameter named "Preference" which can be used to set priority of users who will be provisioned under multiple profiles. This can be used to remove any ambiguity of having a single user being associated with more than one bank.

<Preference>1</Preference>

If the queried API responds back with multiple records then the system shall take the multiple records and mark the record with preference=1 as the preferred account.

Description	Retrieve multiple records associated with one user
Expected Result	Get Preferred record
Requested Phone number	+15158675307
Response	{ name: '7.0.3.5.7.6.8.5.1.5.1.e164enum.net', type: 35, class: 1, ttl: 900, order: 10, preference: 1, flags: 'u', service: 'E2U+mm', regexp: '!^.*\$!mm:001.504@leveloneproject.org!', replacement: " }, { name: '7.0.3.5.7.6.8.5.1.5.1.e164enum.net', type: 35, class: 1, ttl: 900, order: 10,



	<pre>preference: 2, flags: 'u', service: 'E2U+mm', regexp: '!^.*\$!mm:001.499@leveloneproject.org!', replacement: "" }</pre>
Conclusion	The URI associated with Preference=1 is preferred over the other records.



Chapter 5

In This Chapter:

5 CONNECTIVITY BEST PRACTICES 32

5.1 CTE CONNECTIVITY OPTIONS FOR IPSEC VPN 32

5.2 PRODUCTION CONNECTIVITY OPTIONS FOR IPSEC VPN 33

5.3 PRODUCTION CONNECTIVITY OPTIONS FOR GRX/IPX VLAN..... 35

5 Connectivity Best Practices

5.1 CTE Connectivity Options for IPsec VPN

The service options for the QI CTE for IPsec VPN are listed below:

<i>Services</i>	<i>IPsec VPN¹</i>
<i>Number Portability Discovery</i>	✓
<i>ENUM Hosting</i>	✓

The Pathfinder QI CTE resides in the same data center as production using the same VPN peer in Ashburn, Virginia. In order to establish Virtual Private Network (VPN) connectivity, the customer needs to have an IPsec-capable router or firewall (see Production Environment section 3.2 below for more detailed VPN requirements).

An IPsec-capable router or firewall is required for the Production Environment. If the customer plans on using an open-source solution such as Openswan, it is recommended that it is first reviewed with their Account Executive and/or Deployment manager at Neustar. Once reviewed, the VPN should be deployed and tested in CTE before implementation in production. Openswan (<http://www.openswan.org/>) is a software-only implementation of IPsec for Linux.


¹ Internet Protocol Security (IPsec) is a protocol suite for securing IP communications by authenticating and encrypting each IP packet of a communication session. IPsec VPN is a gateway to gateway VPN. IPsec is officially standardized by the [Internet Engineering Task Force](#) (IETF) in a series of [Request for Comments](#) documents addressing various components and extensions.




5.2 Production Connectivity Options for IPsec VPN


The Production Environment supports live PathFinder customers after they have completed their testing in the CTE. This environment is designed and operated to meet contractual SLAs as defined in the Standard Services Agreement between the customer and GSMA.


The primary production connectivity option is via an IPsec Virtual Private Network (VPN).

 **Note:** The term ‘primary’ also refers to recommended traffic distribution. North American customers should route 100% of portability traffic to Ashburn/Denver and split their remote traffic evenly to EUNet (Amsterdam) and Teleticity (Amsterdam). International customers should split all their traffic evenly to EUNet and Teleticity. Ashburn/Denver can serve as a backup in the case of any issues.

 **Note:** *Neustar does not provide technical support for open source VPN software (e.g., Openswan). Further, Neustar can only provide limited technical support for cloud-based web services environments (e.g., Amazon Elastic Compute Cloud).* Consequently, Neustar has extended an SSL PKI connectivity option – which should be considered as a last resort option for customers. Using SSL PKI, similar PathFinder NPD query per second throughput can be achieved relative to IPsec VPNs. However, such responses can be expected to be up to twenty percent slower on average.

For ENUM Hosting service, if the result for a TN query is a non-terminal record, PathFinder will currently not recurse and perform a subsequent query in attempt to discover a terminal answer. PathFinder will return the non-terminal answer (the delegation information) back to the Customer’s application client, which is assumed, in turn, to make the subsequent query in order to obtain the terminal answer. The same will be true for other related queries, such as find the A resource record associated with a FQDN contained in a response from PathFinder.

 **Note:** For IPsec VPN connectivity, the Customer is responsible for failing over to a secondary site in the unlikely event of a failure.

 **Note:** CTE is hosted at one site. In the Production environment, there are currently three Query Interface sites. Each site has multiple query interface applications running behind a load balancer. The Customer can simply establish an IPsec VPN connection to get to the site Virtual IP Address used to issue a query.

The service connectivity options for the production environment for IPsec VPN are summarized below.

<i>Services</i>	<i>IPsec VPN</i>
<i>Number Portability Discovery</i>	✓
<i>ENUM Hosting</i>	✓

In order to establish IPsec VPN connectivity to PathFinder’s Number Portability Discovery and/or ENUM Hosting services, the customer must have an IPsec-capable router or firewall in order to build LAN-to-LAN (also known as site-to-site VPNs) to Cisco ASA or Juniper firewalls hosted in the PathFinder nodes. The advantage to this method of connectivity is that the data exchange is over a secure connection but still over the public Internet. All communications are secured at a minimum using AES algorithms. As shown in Figure 8 below, an example of a commercial IPsec-capable VPN device could be a Juniper SSG or others from vendors like CheckPoint, Cisco, Fortinet and Sonicwall. For any clarifications or questions regarding the VPN requirements/devices supported, please contact your Account Executive and/or Deployment Manager at Neustar.

Note: Due to these VPN requirements, the customer must provide a Public IP address to Pathfinder Support for both the VPN peer and the Encryption Domain of the customer’s VPN. Elastic IP and RFC 1918 addresses are not supported. Customers using RFC 1918 addresses will need to employ source NAT to a public address before traffic enters into the VPN tunnel.

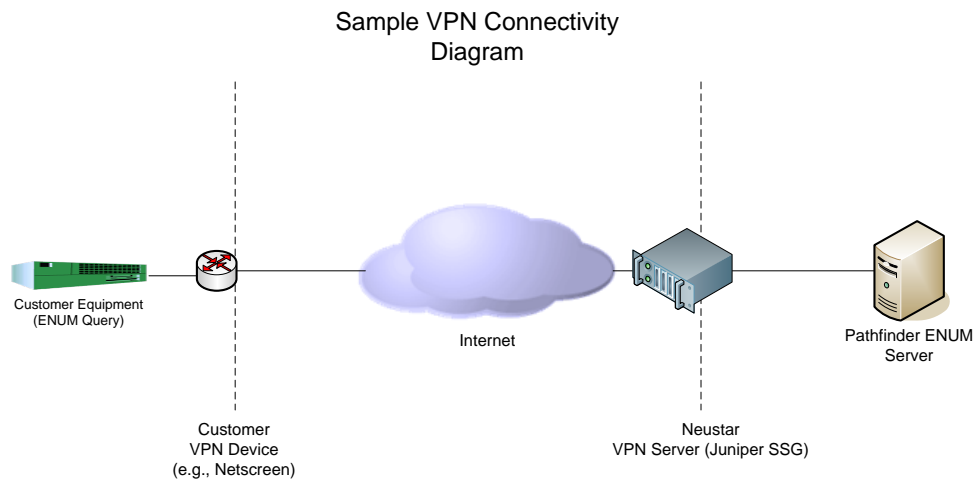


Figure 2: Sample IPsec VPN Connectivity Diagram


The IPsec VPN device should minimally support the following:

- Tunnel mode
- 3DES encryption
- Site to site VPN
- SHA hash
- ISAKMP
- Pre-Shared Keys


A sample list of LAN-to-LAN IPsec-compatible devices includes:




- CheckPoint R7x,
- Fortinet Fortigate-200A
- Juniper SSG5, SSG140, ISG-1000
- Juniper SRX240, SRX550
- Cisco ASA 5500-X
- Cisco IOS router (k9 crypto version)
- Sonicwall NSA series

 **Note:** This list is not meant to be exhaustive by any means as models and vendors change frequently with new devices coming to market regularly.

Where geographical resiliency in Production is required, the customer needs to implement additional VPN connections to secondary sites. ENUM queries may be sent to all active PathFinder query nodes, however if one is “unreachable”, the customer (ENUM client) is responsible for re-directing queries to an alternative query node. In Production, there are currently three active QI nodes to which the customer can connect.

 **Note:** Although UDP and TCP are supported over an IPsec VPN connection, Neustar recommends using UDP instead of TCP as TCP has connection overhead (see also Chapter 2).

5.3 Production Connectivity Options for GRX/IPX VLAN

 **Note:** IPX, or Internet Protocol Exchange, is a telecommunications interconnection model for the exchange of IP-based traffic between customers of separate mobile and fixed operators, as well as other types of service providers via an IP-based Network-to-Network Interface. IPX was developed by GSM Association.

PathFinder’s NPD and ENUM Hosting services are also accessible via a GRX/IPX VLAN. The GRX (Global Roaming Exchange) provides a private IP network, name space, address space and routing between members of the Global Roaming Exchange and provides signaling transport between them. In order to establish GRX/IPX connectivity, the PathFinder customer needs to have a presence in the GRX/IPX cloud directly or via GRX/IPX providers who will establish the connectivity for the PathFinder customer. With GRX/IPX connectivity, it is no longer the responsibility of the Customer to failover to a secondary site whenever the primary site is “unreachable” as it is automatically taken care of by the system design in the GRX/IPX environment.

For ENUM Hosting, each of the Customer’s host servers must be visible and routable over the GRX/IPX VLAN. A Zone Transfer mechanism is available to the Customer to download and locally store the Tier 0 data. It is the Customer’s responsibility to set up a recursive server for resolving the NAPTR RRs into routable IP addresses as required.




The service connectivity options for the production environment for GRX/IPX VLAN are summarized below.

<i>Services</i>	<i>GRX/IPX VLAN</i>
<i>Number Portability Discovery</i>	✓
<i>ENUM Hosting</i>	✓



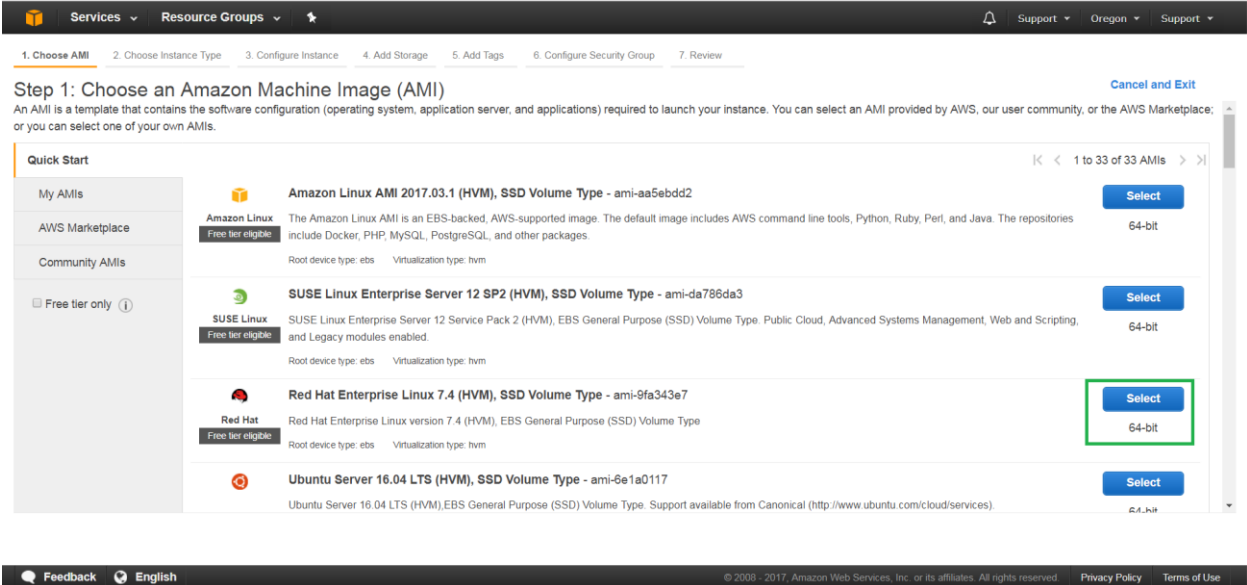
APPENDIX A: AWS VPN GUIDELINE AND OPENSWAN CONFIGURATION

If you are connecting through AWS with an open source vpn, then the following serves as a useful reference.

 **Note:** Note: Neustar does not provide technical support for open source VPN software (e.g., Openswan). Further, Neustar can only provide limited technical support for cloud-based web services environments (e.g., Amazon Elastic Compute Cloud).

1. In AWS launch a RHEL AWS instance

Services => EC2 => Launch an instance



The screenshot shows the AWS console interface for selecting an Amazon Machine Image (AMI). The page title is "Step 1: Choose an Amazon Machine Image (AMI)". Below the title, there is a description: "An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs." The interface lists several AMIs under the "Quick Start" section:

- Amazon Linux AMI 2017.03.1 (HVM), SSD Volume Type - ami-aa5ebdd2**: Includes Docker, PHP, MySQL, PostgreSQL, and other packages. Root device type: ebs, Virtualization type: hvm. 64-bit.
- SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-da786da3**: SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled. Root device type: ebs, Virtualization type: hvm. 64-bit.
- Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type - ami-9fa343e7**: Red Hat Enterprise Linux version 7.4 (HVM), EBS General Purpose (SSD) Volume Type. Root device type: ebs, Virtualization type: hvm. 64-bit. This option is highlighted with a green box.
- Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-6e1a0117**: Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services). 64-bit.

The footer of the console shows "Feedback", "English", and copyright information: "© 2009 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use".



2. Create an elastic IP

Services => EC2 => Network & Security => Elastic IPs

The screenshot shows the AWS Management Console interface for Elastic IPs. The top navigation bar includes 'Services' and 'Resource Groups'. The left sidebar lists various services, with 'Elastic IPs' under 'NETWORK & SECURITY' highlighted. The main content area displays a message: 'You do not have any Addresses in this region. Click the Create Address button to create your first Address.' A blue 'Allocate new address' button is prominently displayed in the center of the page. Another 'Allocate new address' button is visible in the top navigation bar. The footer contains 'Feedback', 'English', and '© 2008 - 2017, Amazon Web Services, Inc'.

3. Allocate the elastic IP to the new instance

Select newly created Elastic IP

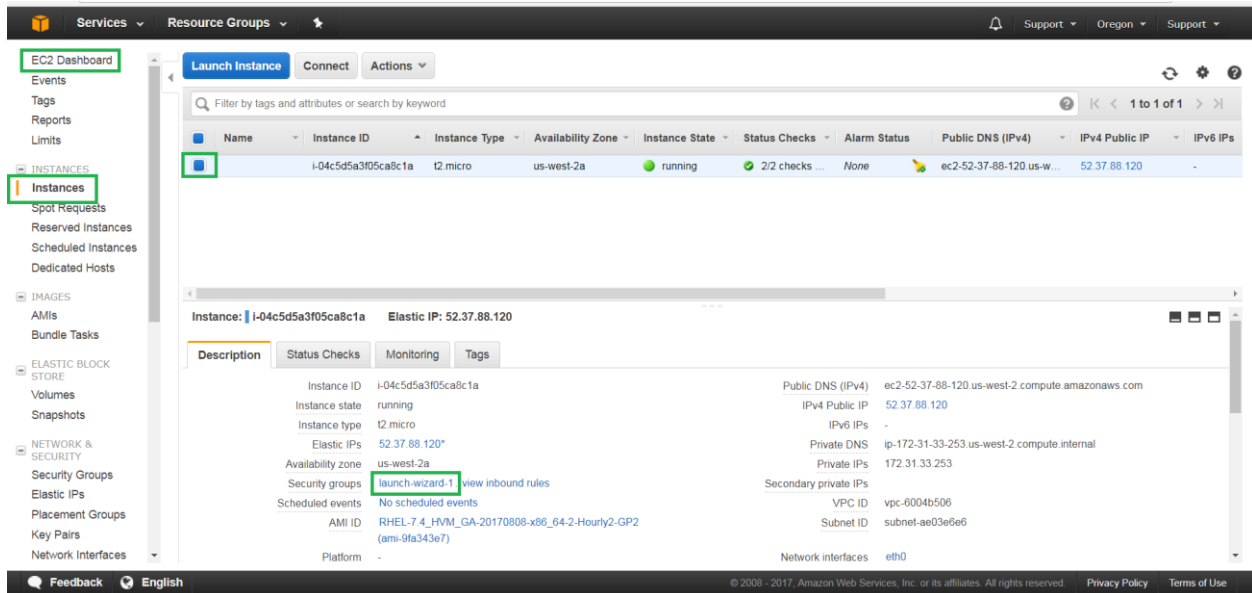
Actions => Associate Address

The screenshot shows the AWS console interface for associating an Elastic IP address. At the top, there are navigation tabs for 'Services' and 'Resource Groups'. The main heading is 'Associate address'. Below this, a message says 'Select the instance OR network interface to which you want to associate this Elastic IP address (52.37.88.120)'. The 'Resource type' is set to 'Instance' (highlighted with a green box). The 'Instance' dropdown menu shows 'i-04c5d5a3f05ca8c1a' (also highlighted with a green box). The 'Private IP' dropdown menu shows '172.31.33.253' (highlighted with a green box). There is a 'Reassociation' checkbox for 'Allow Elastic IP to be reassociated if already attached'. A yellow warning box contains the text: 'Warning: If you associate an Elastic IP address with your instance, your current public IP address is released. [Learn more.](#)'. At the bottom right, there are 'Cancel' and 'Associate' buttons, with the 'Associate' button highlighted by a green box. A footer bar contains 'Feedback', 'English', and copyright information.

4. Security Group

Ensure the Security Group associated with your instance allows udp 500, udp 4500 and ip protocol 50.

Note: By default, it is not allowed.



5. Openswan Configuration – Configure ipsec.conf

```
sudo vi /etc/ipsec.conf
```

- remove everything after virtual_private=
- virtual_private=

```
[ec2-user@ip-xxx-xx-xx-xxx ~]$ sudo cat /etc/ipsec.conf
# /etc/ipsec.conf - Libreswan IPsec configuration file

# Uncomment when using this configuration file with openswan
#version 2
#
# Manual: ipsec.conf.5

config setup
# which IPsec stack to use, "netkey" (the default), "klips" or "mast".
# For MacOSX use "bsd"
protostack=netkey
virtual_private=
#
# Normally, pluto logs via syslog. If you want to log to a file,
# specify below or to disable logging, eg for embedded systems, use
# the file name /dev/null
# Note: SELinux policies might prevent pluto writing to a log file at
# an unusual location.
#logfile=/var/log/pluto.log
#
# Do not enable debug options to debug configuration issues!
#
# plutodebug "all", "none" or a combination from below:
# "raw crypt parsing emitting control controlmore kernel pfkey
# natt x509 dpd dns oppo oppoinfo private".
```



```

# Note: "private" is not included with "all", as it can show confidential
# information. It must be specifically specified
# examples:
# plutodebug="control parsing"
# plutodebug="all crypt"
# Again: only enable plutodebug when asked by a developer
#plutodebug=none
#
# Enable core dumps (might require system changes, like ulimit -C)
# This is required for abrt to work properly
# Note: SELinux policies might prevent pluto writing the core at
# unusual locations
dumpdir=/var/run/pluto/
#
# NAT-TRAVERSAL support
# exclude networks used on server side by adding %v4:!a.b.c.0/24
# It seems that T-Mobile in the US and Rogers/Fido in Canada are
# using 25/8 as "private" address space on their wireless networks.
# This range has never been announced via BGP (at least up to 2015)

# For example connections, see your distribution's documentation directory,
# or https://libreswan.org/wiki/
#
# There is also a lot of information in the manual page, "man ipsec.conf"
#
# It is best to add your IPsec connections as separate files in /etc/ipsec.d/
include /etc/ipsec.d/*.conf
    
```

6. Create your connection(s)

 **Note:** The naming is arbitrary.


```
sudo vi /etc/ipsec.d/neustar-sterling.conf
```


```

conn neustar-sterling
    type=tunnel
    authby=secret
    left=%defaultroute
    leftid=<Elastic IP>
    leftnexthop=%defaultroute
    leftsubnet=<Elastic IP CIDR>
    right=<remote peer>
    rightsubnet=<remote encryption domain CIDR>
    ike=
    phase2alg=
    pfs=yes
    auto=start
    
```

 **Note:** The tabbed declarations are important.



 **Note:** For more than one encryption domain change from <left|right>subnet to <left|right>subnets.


 **Note:** Separate encryption domain entries with a comma. **Ex.**
`rightsubnets=xxx.xxx.xx.x/xx,xxx.xxx.xx.x/xx`



```
[ec2-user@ip-xxx.xx.xx.xxx ~]$ sudo cat /etc/ipsec.d/neustar-sterling.conf
conn neustar-sterling
    type=tunnel
    authby=secret
    left=%defaultroute
    leftid=xx.xx.xx.xxx
    leftnexthop=%defaultroute
    leftsubnet=xx.xx.xx.xxx/xx
    right=xxx.xxx.xx.x
    rightsubnets=xxx.xxx.xx.x/xx,xxx.xxx.xx.x/xx
    ike=aes128-sha1;modp2048
    phase2alg=aes128-sha1;modp2048
    pfs=yes
    auto=start
```

dh-group—Diffie-Hellman group for key establishment.

- group1—768-bit Modular Exponential (MODP) algorithm.
- group2—1024-bit MODP algorithm
- group5—1536-bit MODP algorithm
- group14—2048-bit MODP group.
- group19—256-bit random Elliptic Curve Groups modulo a Prime (ECP groups) algorithm.
- group20—384-bit random ECP groups algorithm.

 **Note:** We recommend using group14, group19, or group20 instead of group1, group2, or group5.

7. Define your pre-shared key

Ensure the name of your .secrets file matches your connection file

 **Note:** Spacing matters!

```
sudo vi /etc/ipsec.d/neustar-sterling.secrets
```

```
<Elastic IP> <Neustar Peer IP>: PSK "Put a Preshared Key here!!"
```

```
[ec2-user@ip-xxx.xx.xx.xxx ~]$ sudo cat /etc/ipsec.d/neustar-sterling.secrets
xx.xx.xx.xxx x.xxx.xx.x : PSK "bogus12345!@#$$%"
```

8. Update sysctl.conf

On the fly



```
sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -w net.ipv4.all.accept_redirects=0
sudo sysctl -w net.ipv4.all.send_redirects=0
```

Permanently

```
sudo vi /etc/sysctl.conf
```

Add three advanced parameters

- net.ipv4.ip_forward = 1
- net.ipv4.conf.all.accept_redirects = 0
- net.ipv4.conf.all.send_redirects = 0

```
[ec2-user@ip-xxx-xx-xx-xxx ~]$ sudo cat /etc/sysctl.conf
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.ipv4.ip_forward = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
```

9. IP Tables

Install

```
sudo yum install iptables-services
```

Update Outbound

```
- sudo iptables -t nat -A POSTROUTING -s <private IP>/32 -d <destination_CIDR> -j SNAT --to-source <elastic IP>
```

```
- Example
```

```
sudo iptables -t nat -A POSTROUTING -s xxx.xx.xx.xxx/xx -d xxx.xxx.xx.x/xx -j SNAT --to-source xx.xx.xx.xxx
```

Update Inbound



- `sudo iptables -t nat -A PREROUTING -s <destination_CIDR> -d <elastic IP_CIDR> -j DNAT --to-destination <private IP>`

- Example

```
sudo iptables -t nat -A PREROUTING -s xxx.xxx.xx.x/xx -d xx.xx.xx.xxx/xx -j DNAT --to-destination 172.31.33.253
```

Verify iptables is setup correct

```
sudo iptables --table nat --list
```

```
[ec2-user@ip-xxx-xx-xx-xxx ~]$ sudo iptables --table nat --list
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
DNAT all -- xxx.xxx.xx.x/xx ec2-xx-xx-xx-xxx.us-west-2.compute.amazonaws.com to:xxx.xx.xx.xxx
DNAT all -- xxx.xxx.xx.x/xx ec2-xx-xx-xx-xxx.us-west-2.compute.amazonaws.com to:xxx.xx.xx.xxx

Chain INPUT (policy ACCEPT)

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
SNAT all -- ip-xxx-xx-xx-xxx.us-west-2.compute.internal xxx-xxx-xx-x/xx to:xx.xx.xx.xxx
SNAT all -- ip-xxx-xx-xx-xxx.us-west-2.compute.internal xxx-xxx-xx-x/xx to:xx.xx.xx.xxx
[ec2-user@ip-xxx-xx-xx-xxx ~]$
```

Save and enable iptables

```
sudo service iptables save
```

```
sudo systemctl enable iptables.service
```

Restart services

- o `sudo service network restart`
- o `sudo systemctl restart ipsec.service`

VPN Status

```
ipsec auto --status
```