NaturalPoint®

NatNet API User's Guide

Version 2.2.0

April 26, 2010

NaturalPoint Corporation

33872 SE Eastgate Circle

Corvallis OR 97339

NaturalPoint Publication Number: P-OT-054

Printed in the US.

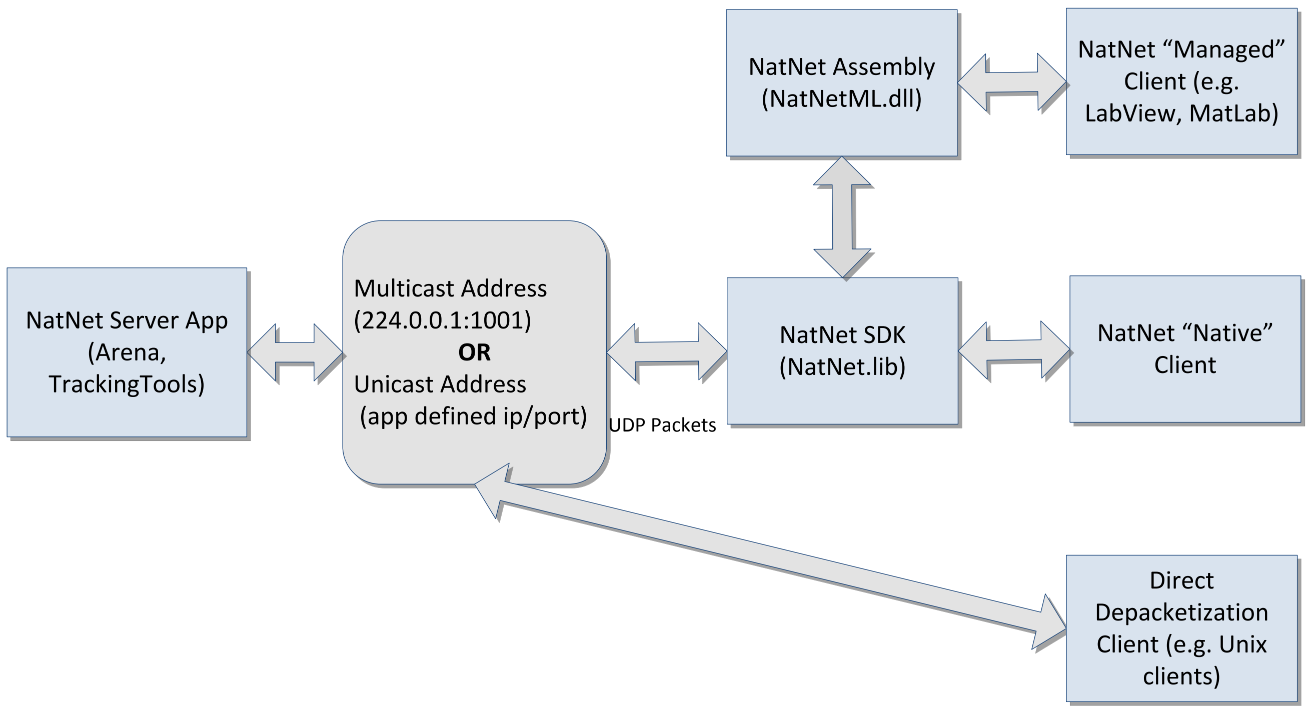
TABLE OF CONTENTS

NATNET OVERVIEW

The NatNet SDK is a Client/Server networking SDK for sending and receiving NaturalPoint data across networks. NatNet uses the UDP protocol in conjunction with either Point-To-Point Unicast or IP Multicasting for sending data.

The following diagram outlines the major component communication of a typical NetNet setup.

Figure 1 – NatNet Component Overview


NatNet Assembly (NatNetML.dll)
NatNet "Managed" Client (e.g. LabView, MatLab)
NatNet Server App (Arena, TrackingTools)
Multicast Address (224.0.0.1:1001) OR Unicast Address (app defined ip/port)
UDP Packets
NatNet SDK (NatNet.lib)
NatNet "Native" Client
Direct Depacketization Client (e.g. Unix clients)

A NatNet Server has 2 threads and 2 sockets, one for sending data, and one for receiving/sending commands. A NatNet Client has 2 threads and 2 sockets, one for receiving data, and one for receiving/sending commands.

NatNet servers and clients can exist on the same or separate machines. Additionally, multiple NatNet clients can connect to a single NatNet server. When a NatNet server is configured to use IP Multicast, the data is only sent once, to the Multicast group.

SDK CONTENTS

The NatNet SDK consists of:

• **NatNet Library**	Native C++ networking library (headers, static library (.lib) and dynamic import library (.lib/.dll))
• **NatNet Assembly**	Managed .NET assembly (NatNetML.dll) for use in .Net compatible clients.
• **NatNet Samples**	Sample projects and executables designed to be quickly integrated into your own code.

FOLDER CONTENTS

Figure 2 - NatNet SDK Folder contents

Folder	Contents
\include	NatNet SDK header files. Client applications should include these.
\lib	Static and dynamic library files for the NatNet SDK.
\lib\x64	64-bit versions of the library files.
\Samples	VisualStudio 2005 samples. Use the solution file here to open all sample projects.
\Samples\bin	Precompiled samples with sample data files.
\Samples\SampleClient	Sample NatNet console app that connects to a NatNet server, receives a data stream, and writes that data stream to an ascii file
\Samples\SampleClient3D	Sample NatNet console app that connects to a NatNet server, receives a data stream, and displays that data in an OpenGL 3D window.
\Samples\SimpleServer	Sample NatNet console app that creates and starts a NatNet server, creates simple Marker, RigidBody, and Skeleton data, and streams that data onto the network.
\Samples\PacketClient	Simple example showing how to connect to a NatNet multicast stream and decode NatNet packets directly without using the NatNet SDK.
\Samples\WinFormsSample	Simple ***C#*** .NET sample showing how to use the NatNet managed assembly (NatNETML.dll).

RUNNING THE SAMPLES

Pre-compiled versions of the NatNet samples have been provided in the \Samples\bin folder. These versions can be used to quickly test your application. Please refer to the instructions in this section for information on running specific samples.

Note! The Visual C++ runtime libraries are required to run the samples. If you encounter an error message when attempting to run the samples, especially on machines without Visual C++ installed, please install the VC runtime redistributable package located in Samples\VCRedist. If the problem peresists, pleas try rebuildingthe samples using VisualC++, or contact support.

RUNNING THE SIMPLE CLIENT-SERVER SAMPLE

1. Start the server:

    SimpleServer.exe

2. Start the client:

    SampleClient.exe [IPAddress] [OutputFilename.txt]

3. Start streaming by pressing 's' in the SimpleServer console window.

You should begin to see data streaming in the client window or to text file.

__Note__

- [parameters] are optional.
- If no IP address is specified, the client will assume the server is on the same machine (local machine).

RUNNING THE RIGID BODY SAMPLE (SAMPLECLIENT3D)

With Client/Server on same machine:

1. [**Arena**] Load a dataset with ridid body or skeleton definitions (pt2 and skl files)
2. [**Arena**] Enable network streaming (Other -> Stream Frames)
3. [**Arena**] Enable streaming rigid body data (check Other-> Rigid Body Data)
4. [**Sample3D**] File -> Connect

With Client/Server on separate machines:

1. [**Arena**] Load a dataset with ridid body or skeleton definitions (pt2 and skl files)
2. [**Arena**] Set IP address to stream from (Other -> IP address edit box)
3. [**Arena**] Enable network streaming (Other -> Stream Frames)
4. [**Arena**] Enable streaming rigid body data (check Other-> Rigid Body Data)
5. [**Sample3D**] Set Client and Server IP addresses
6. [**Sample3D**] File -> Connect

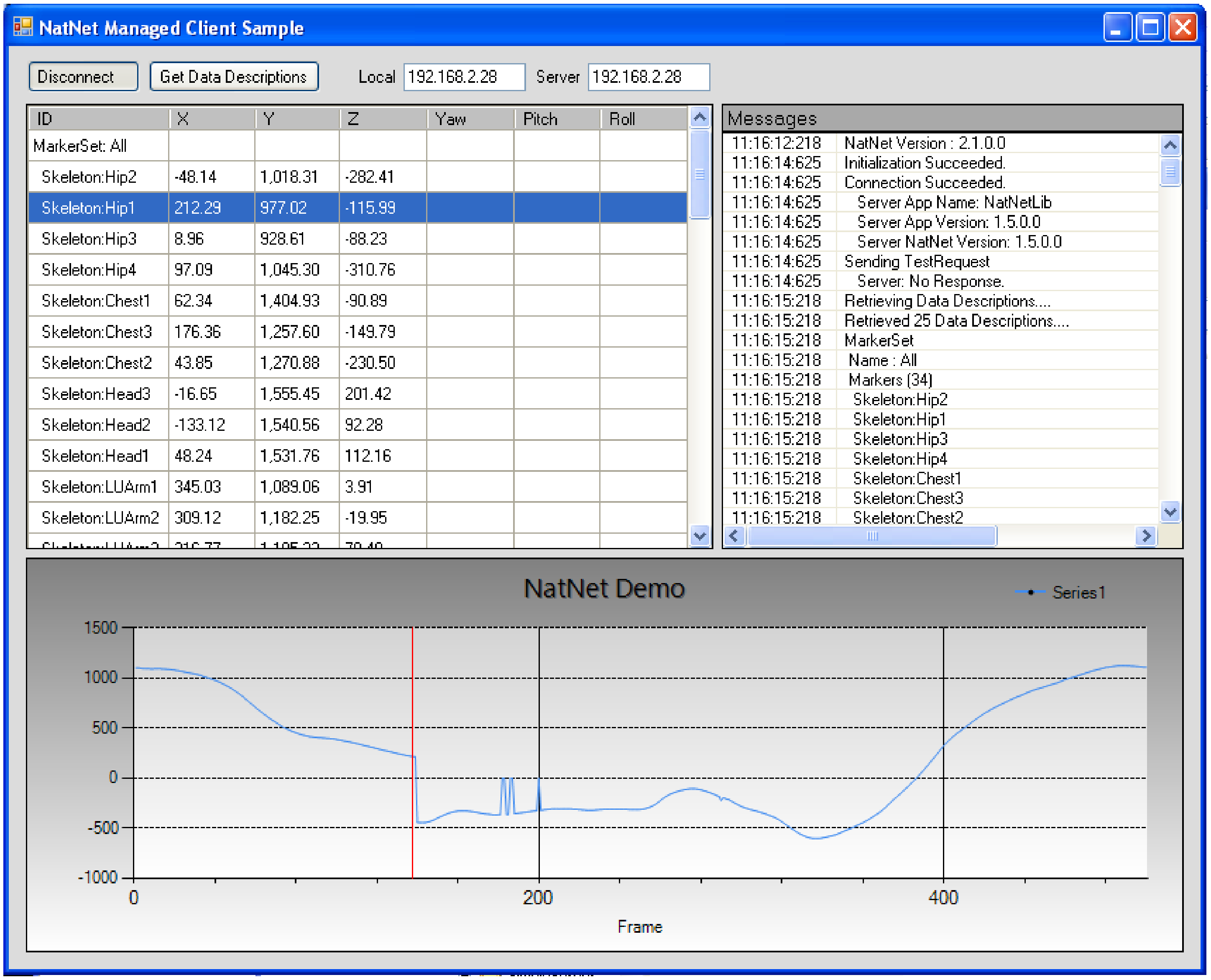
__Note__

• **IP Address**	IP Address of client NIC card you wish to use.
• **Server IP Address**	IP Address of server entered in step 2 above.

RUNNING THE .NET SAMPLE

1. Start a NatNet server application (e.g. Arena or TrackingTools).
2. Enable NatNet streaming from the Server application.
3. Start the WinForms sample application from the NatNet Samples folder.
4. Update the “Local” and “Server” IP Addresses as necessary.
5. Press the “Connect” button to connect to the server.
6. Press the “GetDataDesc” button to request and display a detailed description of the Server’s currently streamed objects.
7. Select a Row in the DataGrid to display that value in the graph.

Figure 3 – Receiving NatNet data in a .NET Environment



ID	X	Y	Z	Yaw	Pitch	Roll
MarkerSet: All						
Skeleton:Hip2	-48.14	1,018.31	-282.41			
Skeleton:Hip1	212.29	977.02	-115.99			
Skeleton:Hip3	8.96	928.61	-88.23			
Skeleton:Hip4	97.09	1,045.30	-310.76			
Skeleton:Chest1	62.34	1,404.93	-90.89			
Skeleton:Chest3	176.36	1,257.60	-149.79			
Skeleton:Chest2	43.85	1,270.88	-230.50			
Skeleton:Head3	-16.65	1,555.45	201.42			
Skeleton:Head2	-133.12	1,540.56	92.28			
Skeleton:Head1	48.24	1,531.76	112.16			
Skeleton:LUArm1	345.03	1,089.06	3.91			
Skeleton:LUArm2	309.12	1,182.25	-19.95			

```
Messages
11:16:12:218   NatNet Version : 2.1.0.0
11:16:14:625   Initialization Succeeded.
11:16:14:625   Connection Succeeded.
11:16:14:625      Server App Name: NatNetLib
11:16:14:625      Server App Version: 1.5.0.0
11:16:14:625      Server NatNet Version: 1.5.0.0
11:16:14:625   Sending TestRequest
11:16:14:625      Server: No Response.
11:16:15:218   Retrieving Data Descriptions....
11:16:15:218   Retrieved 25 Data Descriptions....
11:16:15:218   MarkerSet
11:16:15:218    Name : All
11:16:15:218    Markers (34)
11:16:15:218     Skeleton:Hip2
11:16:15:218     Skeleton:Hip1
11:16:15:218     Skeleton:Hip3
11:16:15:218     Skeleton:Hip4
11:16:15:218     Skeleton:Chest1
11:16:15:218     Skeleton:Chest3
11:16:15:218     Skeleton:Chest2
```

USING THE NATNET SDK

The code samples are the quickest path towards getting NatNet data into your application. We typically recommend you:

1. Identify your application's development/interface requirements (managed, native, etc).
2. Adapt the NatNet sample code from the corresponding NatNet sample application in the samples folder into your application.
3. Use the API reference for additional information.

The Visual Studio solution file \Samples\NatNetSamples.sln will open and build all of the NatNet sample projects.

If you are creating an application from scratch, please refer to the following sections for application specific requirements.

BUILDING A NATIVE CLIENT TO RECEIVE NATNET DATA

Steps for building a NatNet client application/library to receive data from a NatNet server application such as Arena or TrackingTools:

1. Adapt the SampleClient sample (SampleClient.cpp) to your application's code.
2. Include NatNetClient.h, NatNetHelper.h, and NatNetTypes.h
3. Link to NatNetLib.lib (dynamic) **OR** NatNetLibStatic.lib (static)
4. [OPTIONAL] If linking dynamically, define NATNETLIB_IMPORTS and distribute NatNetLib.dll with your application

Note : Be sure to link to ws2_32.lib if linking to NatLetLib statically.

BUILDING A NATIVE SERVER TO SEND NATNET DATA

Steps for building a NatNet server application/library to send/forward NatNet formatted data to a NatNet client application:

1. Adapt SimpleServer (SampleServer.cpp) to your application's code.
2. Include NatNetServer.h, NatNetHelper.h, and NatNetTypes.h
3. Link to NatNetLib.lib (dynamic) **OR** NatNetLibStatic.lib (static)
4. [OPTIONAL] if linking dynamically, define NATNETLIB_IMPORTS and distribute NatNetLib.dll with your application

Note : Be sure to link to ws2_32.lib if linking to NatLetLib statically.

BUILDING A MANAGED .NET CLIENT TO RECEIVE NATNET DATA

Steps for building a managed NatNet client application.

1. Add the NatNetML.dll .NET assembly as a reference to your VB.NET/C# project.
2. The NatNetML namespace is now available to your code, in addition to intellisense library comments.

Note : When distributing your .NET application, be sure to distribute the NatNetML.dll as well.

API REFERENCE

The NatNET API consist of the following objects:

• **NatNetClient**	The class for communicating with a NatNet Server such as Arena or Tracking Tools.
• **NatNetServer**	The class for implementing a NatNet server and sending NatNet formatted data packets.
• **NatNet Data Types**	Structures encapsulating data encoded in NatNet packets.
• **NatNet Assembly**	A managed (.NET) class library that can be called by .NET components. The NatNet assembly wraps the underlying native NatNet library, exposing the NatNetClient and NatNet Data Types for use in .NET compatible environments (e.g. VB.NET, C#, LabView, MatLab).

NATNET DATA TYPES

NatNet server applications stream three (3) fundamental types of motion capture data.

Figure 4 – NatNet Data Types

Data Type	Description
MarkerSet Data	A named collection of identified markers and the marker positions (X,Y,Z)
RigidBody Data	A named segment with a unique ID, position, and orientation data, and the collection of identified markers used to define it.
Skeleton Data	A named, hierarchical collection of RigidBodies.

NatNet clients can discover what data objects a server application is currently streaming using the **DataSetDescriptions** structure.

NatNet clients receive actual data from a server using the **FrameOfMocapData** structure.

Dataset Descriptions This packet contains a description of the motion capture data sets (MarkerSets, Skeletons, RigidBody) for which a frame of motion capture data will be generated.

Frame of Mocap Data This packet contains a single frame of motion capture data for all the data sets described in the Dataset Descriptions.

The SampleClient sample illustrates how to retrieve data descriptions and data and interpret this data.

Please refer to the ***NatNetTypes.h*** header file or the NatNetML.dll assembly for the most up to date descriptions of the types.

NATNETCLIENT CLASS

DESCRIPTION

NatNetClient is a complete C++ class for connecting to NatNet server applications, such as NaturalPoint Arena and NaturalPoint TrackingTools.

CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

NatNetClient::NatNetClient ()

Creates a new (multicast) instance of a NatNet Client.

NatNetClient::NatNetClient (int iConnectionType)

Creates a new instance of a NatNet Client using the specified connection protocol.

Parameters:

iConnectionType *Type of connection (0 = Multicast, 1 = Unicast).*

NatNetClient::~NatNetClient ()

Destructor.

NatNetClient::Uninitialize()

Disconnects from server.

MEMBER FUNCTION DOCUMENTATION

int NatNetClient::GetDataDescriptions (sDataDescriptions ** *pDataDescriptions*)

Requests a description of the current streamed data objects from the server app. This call blocks until request is responded to or times out.

Parameters:

pDataDescriptions *Array of Data Descriptions.*

Returns:

On success, number of data objects. 0 otherwise.

sFrameOfMocapData * NatNetClient::GetLastFrameOfData ()

Retrieves the most recently received frame of mocap data.

Returns:

Frame of Mocap Data

int NatNetClient::GetServerDescription (sServerDescription * *pServerDescription*)

Requests a description of the current NatNet server the client is connected to. This call blocks until request is responded to or times out.

Parameters:

pServerDescription *Description of the NatNet server.*

Returns:

On success, number of data objects. 0 otherwise.

int NatNetClient::Initialize (char * szLocalAddress, char * szServerAddress)
int NatNetClient::Initialize (char * szLocalAddress, char * szServerAddress, int HostCommandPort)
int NatNetClient::Initialize (char * szLocalAddress, char * szServerAddress, int HostCommandPort, int HostDataPort)

Initializes client socket and attempts to connect to a NatNet server at the specified address.

Parameters:

szLocalAddress *IP address of client*
szServerAddress *IP address of server*
HostCommandPort *server command port (default = 1510)*
HostDataPort *server data port (default = 1511)*

Returns:

0 if successful, error code otherwise

void NatNetClient::SetMulticastAddress (char * *szMulticast*)

Sets the NatNet server multicast group/address to connect to. SetMulticastAddress() must be called before calling Initialize(...).

Parameters:

szCommand *application defined Message string*

void NatNetClient::NatNetVersion (unsigned char *Version*[4])

Retrieves the version of the NatNet library the client is using.

Parameters:

Version *version array (form: major.minor.build.revision)*

void NatNetClient::SendMessage (char * *szCommand*)

Sends a message to the server and returns. Response will be delivered in-band.

Parameters:

szCommand *application defined Message string*

int NatNetClient::SendMessageAndWait (char * *szCommand*, int *tries*, int *timeout*, void ** *Response*, int * *pnBytes*)

Sends an application-defined message to the NatNet server and waits for a response.

Parameters:

szCommand *Application defined message.*
tries *Number of times to try and send the message*
timeout *time to wait for response (in milliseconds) before timing out*
Response *Application defined response.*
pnBytes *Number of bytes in response*

Returns:

0 if succssful, error code otherwise.

int NatNetClient::SendMessageAndWait (char * *szCommand*, void ** *Response*, int * *pnBytes*)

Sends an application-defined message to the NatNet server and waits for a response.

Parameters:

szCommand *Application defined message.*
Response *Application defined response.*
pnBytes *Number of bytes in response.*

Returns:

0 if successful, error code otherwise.

int NatNetClient:: SetDataCallback(void (*CallbackFunction)(sFrameOfMocapData *FrameOfData, void* pUserData), void* pUserData /*=NULL*/)

Sets the data callback function for NatNet frame delivery. This function will be called whenever NatNet receives an in-band data (e.g. frame of data).

Parameters:

CallbackFunction *Callback Function*
pUserData *User-Definable data*

Returns:

0 if successful, error code otherwise.

void NatNetClient::SetVerbosityLevel (int *iLevel*)

Sets the message reporting level for internal NatNet messages.

Parameters:

iLevel *Verbosity level (see Verbosity level in **NatNetTypes.h)***

int NatNetClient::Uninitialize ()

Disconnects from the current NatNet Server.

Returns:

0 if successful, error code otherwise.

APPENDIX A : BITSTREAM SYNTAX

In order to provide the most current bitstream syntax, the NatNet SDK includes a testable working depacketization sample that decodes NatNet Packets directly without using the NatNet client library.

Note: Decoding packets directly is not recommended. The bitstream packet syntax is subject to change, requiring an application to rebuild against the latest NatNet library. NatNet packets should only be decoded directly where use of the NatNet library is not possible.

Using the NatNet client library protects client applications from future bistream syntax changes.

BUILDING A DIRECT DEPACKETIZATION CLIENT (WITHOUT NATNET)

For situtions where you would like to receive a NatNet data stream but it is not possible to use the NatNet client library (e.g. on an unsupported platform such as Unix), you can use the PacketClient sample as a template for depacketizing NatNet packets directly.

1. Adapt the PacketClient sample (PacketClient.cpp) to your application's code.
2. Regularly update your code with each revision to the NatNet bitstream syntax.

TECHNICAL SUPPORT

NaturalPoint is committed to providing best-in-class technical support.

In order to provide you with the most up to date information as quickly as possible, we recommend the following procedure:

1. Update to the latest software. For the latest versions of OptiTrack software, drivers, and SDK samples, please visit our downloads section:

   http://www.naturalpoint.com/optitrack/support/downloads.html

2. Check out the OptiTrack FAQs:

   http://www.naturalpoint.com/optitrack/support/opti-faq.html

3. Check the forums. Very often a similar issue has been reported and solved in the forums:

   http://forum.naturalpoint.com/

4. Contact technical support:

   Phone: 541-753-6645

   Fax: 541-753-6689

   Email Form: http://www.naturalpoint.com/optitrack/support/contact/

   Mail: NaturalPoint Corporation
   P.O. Box 2317
   Corvallis, OR 97339