# Vicon DataStream SDK 1.6.0 Developer's Manual

## Contents






Vicon DataStream SDK Developer's Manual May 2016
For use with Vicon DataStream SDK 1.6.0 and later.




Email: support@vicon.com. Web: www.vicon.com.

# About the Vicon DataStream Software Development Kit (SDK)

The Vicon DataStream Software Development Kit (SDK) allows easy programmable access to the information contained in the Vicon DataStream. The function calls within the SDK allow users to connect to and request data from the Vicon DataStream. The following combinations of platforms and technologies are supported:

| | Windows x86 (32-bit) | Windows x64 (64-bit) | Linux x86 (32-bit) | Linux x64 (64-bit) | Mac OSX (64&32-bit) |
|---|---|---|---|---|---|
| C++ | ✓ | ✓ | ✓ | ✓ | |
| .NET | ✓ | ✓ | | | |
| MATLAB | ✓ (can be run on Windows 64-bit OS) | ✓ (requires Microsoft Professional compiler) | | | |

## Important notes:

- Not all function calls contained within the SDK will return data when connected to certain Vicon Applications. For example, Vicon Blade does not support analog devices, and therefore will not output device information into the DataStream.
- The current DataStream format is supported by Vicon Nexus 1.4+, Vicon Blade 1.6+, and Tracker 1.0+. These applications may also output an additional stream in the legacy "Tarsus" format. This DataStream SDK only accesses the DataStream format.
- The current intention is that all future Vicon applications will support the DataStream format.
- Example files are supplied as unsupported examples only.
- The SDK only supports axis transformations into right handed co-ordinate systems.
- The SDK is designed to allow multiple instances of a Client within a single process which can connect to multiple DataStreams.

The SDK is supplied as shared libraries – DLLs on Windows and SOs on Linux. The shared libraries and supporting files are required to be copied alongside your client executable.

# Installing on Windows

There are separate installers for the 32-bit and 64-bit SDKs. The 64-bit installer will only work on a 64-bit version of Windows. The default install directories are:

64-bit Windows

- 32-bit SDK: *C:\Program Files (x86)\Vicon\DataStream SDK\Win32*
- 64-bit SDK: *C:\Program Files\Vicon\DataStream SDK\Win64*

32-bit Windows

- 32-bit SDK : *C:\Program Files\Vicon\DataStream SDK\Win32*

The SDK is supported on Windows 7.

# Installing on Linux

The SDK is provided as a compressed archive. Extract the archive into a convenient location on your system.

# Installing on Mac OSX

The dylibs should be placed in /usr/lib and marked as executable:

```
sudo cp libViconDataStreamSDK_CPP.dylib /usr/lib
sudo cp libDebugServices.dylib /usr/lib
sudo chmod 755 /usr/lib/libViconDataStreamSDK_CPP.dylib
sudo chmod 755 /usr/lib/libDebugServices.dylib
```

# Application linking and redistribution

## Windows – C++

The SDK was built using Visual Studio 2013.
Your application should

- #include "DataStreamClient.h"
- Link against "ViconDataStreamSDK_CPP.lib"
- Redistribute:
  - "ViconDataStreamSDK_CPP.dll"
  - “DebugServices.dll”
  - “boost_*-vc120-mt-1_58.dll”
  - "Microsoft.VC12.CRT"

## Windows – .NET

Your application should

- Link against the assembly "ViconDataStreamSDK_DotNET.dll".
- Redistribute:
  - "ViconDataStreamSDK_DotNET.dll"

- "ViconDataStreamSDK_CPP.dll"
- “DebugServices.dll”
- “boost_*-vc120-mt-1_58.dll”
- "Microsoft.VC12 CRT"

- Have the .NET Framework 4.5 or later installed.

The managed code in this assembly requires the unmanaged code in the C++ SDK

## Windows – MATLAB

Your application M file should be in the same directory as

- "Client.m"
- "DeviceType.m"
- "Direction.m"
- "Result.m"
- "StreamMode.m"
- "TimecodeStandard.m"
- "Unit.m"
- "ViconDataStreamSDK_MATLAB.dll"
- "ViconDataStreamSDK_MATLAB.h"
- “DebugServices.dll”
- “boost_*-vc120-mt-1_58.dll
- "Microsoft.VC12.CRT"

## Linux – C++

Your application should

- #include "Client.h"
- Link against "libViconDataStreamSDK_CPP.so"
- Redistribute
  - "libViconDataStreamSDK_CPP.so"
  - “libDebugServices.so”
  - “libboost_*-mt.so.1.xx.0”

The SDK was compiled with gcc version 4.8.2 (Red Hat 4.8.2-15 from devtools-2 for CentOS 5)

# What's new in version 1.6.0

- New function calls:
  - EnableMarkerRayData
  - DisableMarkerRayData
  - IsMarkerRayDataEnabled
  - GetMarkerRayContributionCount
  - GetMarkerRayContribution

# Requirements

- A compatible licensed version of Vicon Blade, Vicon Nexus, or Vicon Tracker must be present.
- LabVIEW will make use of the .NET dll, and has been found to function in versions 7.1 and 8.
- The MATLAB dll has been found to function in versions 7 and 8.
- The SDK has not been designed to allow access from Simulink.
- The Linux SDK has been specifically verified on CentOS 5.5, Ubuntu 8.04, Ubuntu 9.04, Fedora 9, and Fedora 11. It should also work on any platform supporting glibc 2.5 or later.

# Function *Result* return values

Every function returns a data structure containing elements specified in the Output section of each method reference. Most functions return a Result item, which indicates the success or cause of failure for the function and useful for debugging purposes.

When a function has returned false, the output arguments are set to an appropriate default value:

- Booleans will be set to false.
- Integers will be set to zero.
- Doubles will be set to zero.
- Strings will be set to zero length.
- When the output argument is an array, all elements are set in this manner.

# List of all SDK functions

## Construction and Destruction

*You can create many instances of the Vicon DataStream Client which can connect to multiple Vicon DataStream Servers.*

**C++**

C++ is object oriented, so use the class constructor.

```
{
  ViconDataStreamSDK::CPP::Client StackClient;
  Output_SomeFunction Output = StackClient.SomeFunction();
  …
} // Client is implicitly destroyed as it goes out of scope

ViconDataStreamSDK::CPP::Client * pHeapClient =
                      new ViconDataStreamSDK::CPP::Client();
Output_SomeFunction Output = pHeapClient->SomeFunction( Input );
delete pHeapClient;
```

**MATLAB**

The MATLAB SDK is object oriented, and needs to be explicitly loaded and unloaded.

```
Client.LoadViconDataStreamSDK();

pHeapClient = Client();
Output = pHeapClient.SomeFunction( Input );

Client.UnloadViconDataStreamSDK();
```

**.NET**

.NET is object oriented, so use the class constructor. Because objects are lazily garbage collected, your instance may outlive the last reference to it for some time. If the instance is pre-fetching frame data for you, then it can still use CPU and network bandwidth. Consider explicitly disconnecting prior to destruction.

```
C#
ViconDataStreamSDK.DotNET.Client pHeapClient =
                        new ViconDataStreamSDK.DotNET.Client();
Output_SomeFunction Output = pHeapClient.SomeFunction( InputParam );

// Signal to the garbage collector that it can clean up
pHeapClient.Disconnect();
pHeapClient = null;
```

## Result

| *The Result code indicates the success or failure of a function.* | | |
|---|---|---|
| | Unknown | The result is unknown. Treat it as a failure. |
| | NotImplemented | The function called has not been implemented in this version of the SDK. |
| | Success | The function call succeeded. |
| | InvalidHostName | The "HostName" parameter passed to Connect() is invalid. |
| | InvalidMulticastIP | The "MulticastIP" parameter was not in the range "224.0.0.0" – "239.255.255.255" |
| | ClientAlreadyConnected | Connect() was called whilst already connected to a DataStream. |
| | ClientConnectionFailed | Connect() could not establish a connection to the DataStream server. |
| | ServerAlreadyTransmittingMultcast | StartTransmittingMulticast() was called when the current DataStream server was already transmitting multicast on behalf of this client. |
| | ServerNotTransmittingMulticast | StopTransmittingMulticast() was called when the current DataStream server was not transmitting multicasr on behalf of this client. |
| | NotConnected | You have called a function which requires a connection to the DataStream server, but do not have a connection. |
| | NoFrame | You have called a function which requires a frame to be fetched from the DataStream server, but do not have a frame. |
| | InvalidIndex | An index you have passed to a function is out of range. |
| | InvalidCameraName | The Camera Name you passed to a function is invalid in this frame |
| | InvalidSubjectName | The Subject Name you passed to a function is invalid in this frame. |
| | InvalidSegmentName | The Segment Name you passed to a |

| | | |
|---|---|---|
| | | function is invalid in this frame. |
| | InvalidMarkerName | The Marker Name you passed to a function is invalid in this frame. |
| | InvalidDeviceName | The Device Name you passed to a function is invalid in this frame. |
| | InvalidDeviceOutputName | The Device Output Name you passed to a function is invalid in this frame. |
| | InvalidLatencySampleName | The Latency Sample Name you passed to a function is invalid in this frame. |
| | CoLinearAxes | The directions passed to SetAxisMapping() contain input which would cause two or more axis to lie along the same line, e.g. "Up" and "Down" are on the same line. |
| | LeftHandedAxes | The directions passed to SetAxisMapping() would result in a left handed co-ordinate system. This is not supported in the SDK. |
| | HapticAlreadySet | Haptic feedback is already set |
| C++ | <pre>namespace ViconDataStreamSDK<br>{<br>namespace CPP<br>{<br>namespace Result<br>{<br>  enum Enum<br>  {<br>  Unknown,<br>  NotImplemented,<br>  Success,<br>  InvalidHostName,<br>  InvalidMulticastIP,<br>  ClientAlreadyConnected,<br>  ClientConnectionFailed,<br>  ServerAlreadyTransmittingMulticast,<br>  ServerNotTransmittingMulticast,<br>  NotConnected,<br>  NoFrame,<br>  InvalidIndex,<br>  InvalidCameraName,<br>  InvalidSubjectName,<br>  InvalidSegmentName,<br>  InvalidMarkerName,<br>  InvalidDeviceName,<br>  InvalidDeviceOutputName,<br>  InvalidLatencySampleName,<br>  CoLinearAxes,<br>  LeftHandedAxes,<br>  HapticAlreadySet<br>  };<br>}</pre> | |

MATLAB

```
classdef Result
  properties (Constant = true)
    Unknown                              = 0;
    NotImplemented                       = 1;
    Success                              = 2;
    InvalidHostName                      = 3;
    InvalidMulticastIP                   = 4;
    ClientAlreadyConnected               = 6;
    ClientConnectionFailed               = 7;
    ServerAlreadyTransmittingMulticast = 8;
    ServerNotTransmittingMulticast     = 9;
    NotConnected                         = 10;
    NoFrame                              = 11;
    InvalidIndex                         = 12;
    InvalidCameraName          =13;
    InvalidSubjectName                   = 14;
    InvalidSegmentName                   = 15;
    InvalidMarkerName                    = 16;
    InvalidDeviceName                    = 17;
    InvalidDeviceOutputName              = 18;
    InvalidLatencySampleName             = 19;
    CoLinearAxes                         = 20;
    LeftHandedAxes                       = 21;
    HapticAlreadySet                    = 22;
  end

  properties
    Value
  end

  methods
    function obj = Result( value )
      obj.Value = value;
    end% Constructor
  end% methods

end% classdef
```

.NET

```
namespace ViconDataStreamSDK
{
namespace DotNET
{
public enum class Result
  {
    Unknown,
    NotImplemented,
    Success,
    InvalidHostName,
    InvalidMulticastIP,
    ClientAlreadyConnected,
    ClientConnectionFailed,
    ServerAlreadyTransmittingMulticast,
    ServerNotTransmittingMulticast,
    NotConnected,
    NoFrame,
    InvalidIndex,
    InvalidCameraName
    InvalidSubjectName,
    InvalidSegmentName,
    InvalidMarkerName,
    InvalidDeviceName,
    InvalidDeviceOutputName,
    InvalidLatencySampleName,
    CoLinearAxes,
    LeftHandedAxes
  };
} // End of namespace DotNET
} // End of namespace ViconDataStreamSDK
```

# GetVersion

| *Get the version of the Vicon DataStream SDK* | | | |
|---|---|---|---|
| Input | | | |
| Output | Major | unsigned int | The major version number. When this number increases we break backwards compatibility with previous major versions. |
| | Minor | unsigned int | The minor version number.<br>When this number increases we have probably added new functionality to the SDK without breaking backwards compatibility with previous versions. |
| | Point | unsigned int | The point version number.<br>When this number increases, we have introduced a bug fix or performance enhancement without breaking backwards compatibility. |

C++

```
// class Output_GetVersion
// {
// public:
//   unsigned int Major;
//   unsigned int Minor;
//   unsigned int Point;
// };
//
// Output_GetVersion GetVersion() const;

ViconDataStreamSDK::CPP::Client MyClient;
Output_GetVersion Output = MyClient.GetVersion();
```

MATLAB

```
% [Output] = GetVersion()

MyClient = Client();
Output = MyClient.GetVersion();
```

.NET

```
// class Output_GetVersion
// {
//   public uint Major;
//   public uint Minor;
//   public uint Point;
// };
//
// Output_GetVersion GetVersion();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
Output_GetVersion Output = MyClient.GetVersion();
```

# Connect

<table>
<tr><td colspan="4">Establish a dedicated connection to a Vicon DataStream Server<br>See Also : ConnectToMulticast, Disconnect, IsConnected</td></tr>
<tr><td>Input</td><td>Host Name</td><td>string</td><td>The DNS identifiable name, or IP address of the PC hosting the DataStream server. The function defaults to connecting on port 801. You can specify an alternate port number after a colon.<br><code>"localhost"</code><br><code>"MyViconPC:804"</code><br><code>"10.0.0.2"</code></td></tr>
<tr><td>Output</td><td>Result</td><td>Result</td><td><code>Result.Success</code><br><code>Result.InvalidHostName</code><br><code>Result.ClientAlreadyConnected</code><br><code>Result.ClientConnectionFailed</code></td></tr>
<tr><td>C++</td><td colspan="3"><pre>// class Output_Connect
// {
// public:
//   Result::Enum Result;
// };
//
// Output_Connect Connect( const String & HostName );

ViconDataStreamSDK::CPP::Client MyClient;
Output_Connect Output = MyClient.Connect( "localhost:801" );</pre></td></tr>
<tr><td>MATLAB</td><td colspan="3"><pre>% [Output] = Connect()

MyClient = Client();
Output = MyClient.Connect( 'locahost:801' );</pre></td></tr>
<tr><td>.NET</td><td colspan="3"><pre>// class Output_Connect
// {
//   public Result Result;
// };
//
// Output_Connect Connect( string HostName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
Output_Connect Output =  MyClient.Connect( "localhost:801" );</pre></td></tr>
</table>

# ConnectToMulticast

*Connect to a Vicon DataStream Server's Multicast stream. The stream content is managed by a client who calls StartTransmittingMulticast().*

*See Also : Connect, Disconnect, IsConnected, StartTransmittingMulticast, StopTransmittingMulticast*

| | | | |
|---|---|---|---|
| Input | LocalIP | string | The DNS identifiable name, or IP address of the local Ethernet interface on which you wish to receive multicast data. You should not specify a port (any port specified will be ignored). e.g.<br>`"localhost"`<br>`"10.0.0.2"` |
| | Multicast IP | string | The IP Address of the Multicast group on which data will be received. The address should be in the range "224.0.0.0" – "239.255.255.255" You may also specify a port by appending it to the end of the IP Address after a colon. e.g. 224.0.0.0:30001. If you do not specify a port it will default to 44801. |
| Output | Result | Result | `Result.Success`<br>`Result.InvalidHostName`<br>`Result.InvalidMulticastIP`<br>`Result.ClientAlreadyConnected`<br>`Result.ClientConnectionFailed` |

C++

```
// class Output_ConnectToMulticast
// {
// public:
//   Result::Enum Result;
// };
//
// Output_ConnectToMulticast
//  ConnectToMulticast ( const String & LocalIP,
//                       const String & MulticastIP  );

ViconDataStreamSDK::CPP::Client MyClient;
Output_ConnectToMulticast Output =
  MyClient.ConnectToMulticast( "localhost", "224.0.0.0" );
```

MATLAB

```
% [Output] = ConnectToMulticast()

MyClient = Client();
Output = MyClient.ConnectToMulticast( 'locahost', '224.0.0.0' );
```

.NET

```
// class Output_ConnectToMulticast
// {
//   public Result Result;
// };
//
// Output_ConnectToMulticast ConnectToMulticast ( string LocalIP,
//                                                string MulticastIP );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
Output_ConnectToMulticast Output =
  MyClient.ConnectToMulticast( "localhost", "224.0.0.0" );
```

# Disconnect

| | | | |
|---|---|---|---|
| *Disconnect from the Vicon DataStream Server.*<br>*See Also : Connect, IsConnected* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected` |

C++

```
// class Output_Disconnect
//   {
//   public:
//     Result::Enum Result;
//   };
//
// Output_Disconnect Disconnect();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_Disconnect Output = MyClient.Disconnect();
```

MATLAB

```
% [Output] = Connect()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.Disconnect();
```

.NET

```
// public class Output_Disconnect
// {
// public Result Result;
// };
//
// Output_Disconnect Disconnect()

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_Disconnect Output = MyClient.Disconnect();
```

# IsConnected

*Discover whether client is connected to the Vicon DataStream Server.*

*See Also : Connect, Disconnect*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Connected | boolean | True if you are connected to the stream, otherwise false. |

C++

```
// class Output_IsConnected
// {
// public:
//   bool Connected;
// };
//
// Output_IsConnected IsConnected() const;

ViconDataStreamSDK::CPP::CPP::Client MyClient;
Output_IsConnected Output = MyClient.IsConnected()
                          // Output.Connected == false
MyClient.Connect( "localhost" );
Output_IsConnected Output = MyClient.IsConnected()
                          // Output.Connected == true
                          // (assuming localhost is serving)
```

MATLAB

```
% [Output] = IsConnected()

MyClient = Client();
Output = MyClient.IsConnected()   // Output.Connected == false
MyClient.Connect( "localhost" );
Output = MyClient.IsConnected()   // Output.Connected == true
                                  // (assuming localhost is serving)
```

.NET

```
// public class Output_IsConnected
// {
//  public bool Connected;
// };
//
// Output_IsConnected IsConnected();

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
Output_IsConnected Output = MyClient.IsConnected()
                            // Output.Connected == false
MyClient.Connect( "localhost" );
Output_IsConnected Output = MyClient.IsConnected()
                            // Output.Connected == true
                            // (assuming localhost is serving)
```

# StartTransmittingMulticast

<table>
<tr><td colspan="4">Ask the DataStream Server to start transmitting the data you are receiving directly to a Multicast address as well. This allows multiple clients to connect to your stream (via ConnectToMulticast() ) whilst minimizing network bandwidth use and frame delivery latency.<br>See Also : Connect, ConnectToMulticast, Disconnect, StopTransmittingMulticast</td></tr>
<tr><td>Input</td><td>ServerIP</td><td>string</td><td>The IP Address of the server Ethernet interface from which the Multicast data will be sent. You should not specify a port number (any port number specified will be ignored)</td></tr>
<tr><td></td><td>MulticastIP</td><td>string</td><td>The IP Address of the Multicast group to which multicast data will be sent. The address should be in the range "224.0.0.0" – "239.255.255.255" You may also specify the port the data will be sent to by appending it to the IP Address after a colon e.g. 224.0.0.0:30001. If you do not specify a port it will default to 44801.</td></tr>
<tr><td>Output</td><td>Result</td><td>Result</td><td>Result.Success<br>Result.NotConnected<br>Result.InvalidMulticastIP<br>Result.ServerAlreadyTransmittingMulticast</td></tr>
<tr><td>C++</td><td colspan="3">

```
// class Output_StartTransmittingMulticast
// {
// public:
//   Result::Enum Result;
// };
//
// Output_StartTransmittingMulticast
//  StartTransmittingMulticast ( const String & ServerIP,
//                               const String & MulticastIP ) const;

ViconDataStreamSDK::CPP::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.StartTransmittingMulticast( "10.0.0.1", "224.0.0.0" );
```

</td></tr>
<tr><td>MATLAB</td><td colspan="3">

```
% [Output] = StartTransmittingMulticast ()

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.StartTransmittingMulticast( '10.0.0.1', '224.0.0.0' );
```

</td></tr>
<tr><td>.NET</td><td colspan="3">

```
// public class Output_StartTransmittingMulticast
// {
//  public Result Result;
// };
//
// Output_StartTransmittingMulticast
//  StartTransmittingMulticast( string ServerIP, string MulticastIP );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.StartTransmittingMulticast("10.0.0.1", "224.0.0.0" );
```

</td></tr>
</table>

# StopTransmittingMulticast

*Ask the DataStream Server to stop transmitting the data you are receiving directly to a Multicast address as well. You must previously have started a transmission via StartTransmittingMulticast.*

*See Also : Connect, ConnectToMulticast, Disconnect, StartTransmittingMulticast*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | Result.Success<br>Result.NotConnected<br>Result.ServerNotTransmittingMulticast |

C++

```
// class Output_StopTransmittingMulticast
// {
// public:
//   Result::Enum Result;
// };
//
// Output_StopTransmittingMulticast
//  StopTransmittingMulticast () const;

ViconDataStreamSDK::CPP::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.StartTransmittingMulticast( "10.0.0.1", "224.0.0.0" );
// Do some stuff
MyClient.StopTransmittingMulticast();
```

MATLAB

```
% [Output] = StopTransmittingMulticast ()

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.StartTransmittingMulticast( '10.0.0.1', '224.0.0.0' );
% Do some stuff
MyClient.StopTransmittingMulticast();
```

.NET

```
// public class Output_StopTransmittingMulticast
// {
//  public Result Result;
// };
//
// Output_StopTransmittingMulticast
//  StopTransmittingMulticast();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.StartTransmittingMulticast( "10.0.0.1", "224.0.0.0" );
// Do some stuff
MyClient.StopTransmittingMulticast();
```

# EnableSegmentData

*Enable kinematic segment data in the Vicon DataStream. You should call this function on startup, after connecting to the server, and before trying to read local or global segment data.*

*See Also : IsSegmetnDataEnabled, DisableSegmentData, EnableMarkerData, EnableUnlabelledMarkerData, EnableDeviceData, GetSegmentCount, GetSegmentName, GetSegmentGlobalTranslation, GetSegmentGlobalRotationXXX, GetSegmentLocalTranslation, GetSegmentLocalRotationXXX*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_EnableSegmentData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_EnableSegmentData EnableSegmentData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_EnableSegmentData Output = MyClient.EnableSegmentData();
```

MATLAB

```
% [Output] = EnableSegmentData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.EnableSegmentData();
```

.NET

```
// public class Output_EnableSegmentData
// {
//   public Result Result;
// };
//
// Output_EnableSegmentData EnableSegmentData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_EnableSegmentData Output = MyClient.EnableSegmentData();
```

# EnableMarkerData

*Enable labeled reconstructed marker data in the Vicon DataStream. Call this function on startup, after connecting to the server, and before trying to read labeled marker data.*

*See Also : IsMarkerDataEnabled, DisableMarkerData, EnableSegmentData, EnableUnlabelledMarkerData, EnableDeviceData, GetMarkerCount, GetMarkerName, GetMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_EnableMarkerData
//   {
//   public:
//     Result::Enum Result;
//   };
//
// Output_EnableMarkerData EnableMarkerData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_EnableMarkerData Output = MyClient.EnableMarkerData();
```

MATLAB

```
% [Output] = EnableMarkerData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.EnableMarkerData();
```

.NET

```
// public class Output_EnableMarkerData
// {
//   public Result Result;
// };
//
// Output_EnableMarkerData EnableMarkerData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_EnableMarkerData Output = MyClient.EnableMarkerData();
```

# EnableUnlabeledMarkerData

*Enable unlabeled reconstructed marker data in the Vicon DataStream. You should call this function on startup, after connecting to the server, and before trying to read global unlabeled marker data.*

*See Also : IsUnlabeledMarkerDataEnabled, DisableUnlabeledMarkerData, EnableSegmentData, EnableMarkerData, EnableDeviceData, GetUnlabeledMarkerCount, GetUnlabeledMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_EnableUnlabeledMarkerData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_EnableUnlabeledMarkerData EnableUnlabeledMarkerData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_EnableUnlabeledMarkerData Output =
                                    MyClient.EnableUnlabeledMarkerData();
```

MATLAB

```
% [Output] = EnableUnlabeledMarkerData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.EnableUnlabeledMarkerData();
```

.NET

```
// public class Output_EnableUnlabeledMarkerData
// {
//   public Result Result;
// };
//
// Output_EnableUnlabeledMarkerData EnableUnlabeledMarkerData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_EnableUnlabeledMarkerData Output =
                                    MyClient.EnableUnlabeledMarkerData();
```

# EnableMarkerRayData

*Enable information about the rays contributing toward each labelled marker in the Vicon DataStream. You should call this function on startup, after connecting to the server, and before trying to read global unlabeled marker data.*

*See Also : IsMarkerRayDataEnabled, DisableMarkerRayData, EnableSegmentData, EnableMarkerData, EnableDeviceData, GetUnlabeledMarkerCount, GetUnlabeledMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_EnableMarkerRayData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_EnableMarkerRayData EnableMarkerRayData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_EnableMarkerRayData Output =
                                    MyClient.EnableMarkerRayData();
```

MATLAB

```
% [Output] = EnableMarkerRayData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.EnableMarkerRayData();
```

.NET

```
// public class Output_EnableMarkerRayData
// {
//   public Result Result;
// };
//
// Output_EnableMarkerRayData EnableMarkerRayData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_EnableMarkerRayData Output =
                                    MyClient.EnableMarkerRayData();
```

# EnableDeviceData

| | | | |
|---|---|---|---|
| *Enable ForcePlate, EMG, and other device data in the Vicon DataStream. You should call this function on startup, after connecting to the server, and before trying to read device information.*<br>*See Also : IsDeviceDataEnabled, DisableDeviceData, EnableSegmentData, EnableMarkerData, EnableUnlabeledMarkerData, GetDeviceCount, GetDeviceName, GetDeviceOutputCount, GetDeviceOutputName,GetDeviceOutputValue* | | | |
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_EnableDeviceData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_EnableDeviceData EnableDeviceData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_EnableDeviceData Output = MyClient.EnableDeviceData();
```

MATLAB

```
% [Output] = EnableDeviceData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.EnableDeviceData();
```

.NET

```
// public class Output_EnableDeviceData
// {
//   public Result Result;
// };
//
// Output_EnableDeviceData EnableDeviceData();

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_EnableDeviceData Output = MyClient.EnableDeviceData();
```

# EnableCentroidData

| | | | |
|---|---|---|---|
| *Enable centroid data in the Vicon DataStream. You should call this function on startup, after connecting to the server, and before trying to read centroid information.*<br>*See Also : IsCentroidDataEnabled, DisableCentroidData* | | | |
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_EnableCentroidData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_EnableCentroidData EnableCentroidData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_EnableCentroidData Output = MyClient.EnableCentroidData();
```

MATLAB

```
% [Output] = EnableCentroidData ()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.EnableCentroidData ();
```

.NET

```
// public class Output_EnableCentroidData
// {
//   public Result Result;
// };
//
// Output_EnableCentroidData EnableCentroidData ();

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_EnableCentroidData Output = MyClient.EnableCentroidData ();
```

# DisableSegmentData

*Disable kinematic segment data in the Vicon DataStream.*

*See Also : IsSegmetnDataEnabled, EnableSegmentData, EnableMarkerData, EnableUnlabelledMarkerData, EnableDeviceData, GetSegmentCount, GetSegmentName, GetSegmentGlobalTranslation, GetSegmentGlobalRotationXXX, GetSegmentLocalTranslation, GetSegmentLocalRotationXXX*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_DisableSegmentData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_DisableSegmentData DisableSegmentData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_DisableSegmentData Output = MyClient.DisableSegmentData();
```

MATLAB

```
% [Output] = DisableSegmentData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.DisableSegmentData();
```

.NET

```
// public class Output_DisableSegmentData
// {
//   public Result Result;
// };
//
// Output_DisableSegmentData DisableSegmentData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_DisableSegmentData Output = MyClient.DisableSegmentData();
```

# DisableMarkerData

*Disable labeled reconstructed marker data in the Vicon DataStream.*

*See Also : IsMarkerDataEnabled, EnableMarkerData, EnableSegmentData, EnableUnlabelledMarkerData, EnableDeviceData, GetMarkerCount, GetMarkerName, GetMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_DisableMarkerData
//   {
//   public:
//     Result::Enum Result;
//   };
//
// Output_DisableMarkerData DisableMarkerData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_DisableMarkerData Output = MyClient.DisableMarkerData();
```

MATLAB

```
% [Output] = DisableMarkerData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.DisableMarkerData();
```

.NET

```
// public class Output_DisableMarkerData
// {
//   public Result Result;
// };
//
// Output_DisableMarkerData DisableMarkerData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                 new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_DisableMarkerData Output = MyClient.DisableMarkerData();
```

# DisableUnlabeledMarkerData

*Disable unlabeled reconstructed marker data in the Vicon DataStream.*

*See Also : IsUnlabeledMarkerDataEnabled, EnableUnlabeledMarkerData, EnableSegmentData, EnableMarkerData, EnableDeviceData, GetUnlabeledMarkerCount, GetUnlabeledMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_DisableUnlabeledMarkerData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_DisableUnlabeledMarkerData DisableUnlabeledMarkerData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_DisableUnlabeledMarkerData Output =

MyClient.DisableUnlabeledMarkerData();
```

MATLAB

```
% [Output] = DisableUnlabeledMarkerData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.DisableUnlabeledMarkerData();
```

.NET

```
// public class Output_DisableUnlabeledMarkerData
// {
//   public Result Result;
// };
//
// Output_DisableUnlabeledMarkerData DisableUnlabeledMarkerData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_DisableUnlabeledMarkerData Output =

MyClient.DisableUnlabeledMarkerData();
```

# DisableMarkerRayData

*Disable unlabeled reconstructed marker data in the Vicon DataStream.*

*See Also : IsMarkerRayDataEnabled, EnableMarkerRayData, EnableSegmentData, EnableMarkerData, EnableDeviceData, GetUnlabeledMarkerCount, GetUnlabeledMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_DisableMarkerRayData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_DisableMarkerRayData DisableMarkerRayData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_DisableMarkerRayData Output =
                                    MyClient.DisableMarkerRayData();
```

MATLAB

```
% [Output] = DisableMarkerRayData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.DisableMarkerRayData();
```

.NET

```
// public class Output_DisableMarkerRayData
// {
//   public Result Result;
// };
//
// Output_DisableMarkerRayData DisableMarkerRayData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_DisableMarkerRayData Output =
                                    MyClient.DisableMarkerRayData();
```

# DisableDeviceData

*Disable ForcePlate, EMG, and other device data in the Vicon DataStream.*

*See Also : IsDeviceDataEnabled, EnableDeviceData, EnableSegmentData, EnableMarkerData, EnableUnlabeledMarkerData, GetDeviceCount, GetDeviceName, GetDeviceOutputCount, GetDeviceOutputName,GetDeviceOutputValue*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |

C++

```
// class Output_DisableDeviceData
//  {
//  public:
//    Result::Enum Result;
//  };
//
// Output_DisableDeviceData DisableDeviceData();

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_DisableDeviceData Output = MyClient.DisableDeviceData();
```

MATLAB

```
% [Output] = DisableDeviceData()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.DisableDeviceData();
```

.NET

```
// public class Output_DisableDeviceData
// {
//   public Result Result;
// };
//
// Output_DisableDeviceData DisableDeviceData();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_DisableDeviceData Output = MyClient.DisableDeviceData();
```

# DisableCentroidData

| | | | |
|---|---|---|---|
| *Disable Centroid data in the Vicon DataStream.*<br>*See Also : IsCentroidDataEnabled, EnableCentroidData* | | | |
| Input | | | |
| Output | Result | Result | `Result.NotConnected`<br>`Result.Success` |
| C++ | `// class Output_DisableCentroidData`<br>`//  {`<br>`//  public:`<br>`//  Result::Enum Result;`<br>`//  };`<br>`//`<br>`// Output_DisableCentroidData DisableCentroidData();`<br><br>`ViconDataStreamSDK::CPP::Client MyClient;`<br>`MyClient.Connect( "localhost" );`<br>`Output_DisableCentroidData Output = MyClient.DisableCentroidData();` | | |
| MATLAB | `% [Output] = DisableCentroidData ()`<br><br>`MyClient = Client();`<br>`MyClient.Connect( "localhost" );`<br>`Output = MyClient.DisableCentroidData ();` | | |
| .NET | `// public class Output_DisableCentroidData`<br>`// {`<br>`//  public Result Result;`<br>`// };`<br>`//`<br>`// Output_DisableCentroidData DisableCentroidData ();`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient =`<br>`new ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`Output_DisableCentroidData Output = MyClient.DisableCentroidData ();` | | |

# IsSegmentDataEnabled

*Return whether kinematic segment data is enabled in the Vicon DataStream.*

*See Also : EnableSegmentData, DisableSegmentData, IsMarkerDataEnabled. IsUnlabeledMarkerDataEnabled, IsDeviceDataEnabled*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Enabled | boolean | Whether the data is enabled. |

C++

```
// class Output_IsSegmentDataEnabled
// {
// public:
//   bool Enabled;
// };
//
// Output_IsSegmentDataEnabled IsSegmentDataEnabled() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_IsSegmentDataEnabled Output = MyClient.IsSegmentDataEnabled();
                                     // Output.Enabled == false
MyClient.EnableSegmentData();
Output_IsSegmentDataEnabled Output = MyClient.IsSegmentDataEnabled();
                                     // Output.Enabled == true
```

MATLAB

```
% [Output] = IsSegmentDataEnabled()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.IsSegmentDataEnabled(); % Output.Enabled == false
MyClient.EnableSegmentData();
Output = MyClient.IsSegmentDataEnabled(); % Output.Enabled == true
```

.NET

```
// public class Output_IsSegmentDataEnabled
// {
//   public bool Enabled;
// };
// Output_IsSegmentDataEnabled IsSegmentDataEnabled();

ViconDataStreamSDK.DotNET.Client MyClient =             new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_IsSegmentDataEnabled Output = MyClient.IsSegmentDataEnabled();
                                     // Output.Enabled == false
MyClient.EnableSegmentData();
Output_IsSegmentDataEnabled Output = MyClient.IsSegmentDataEnabled();
                                     // Output.Enabled == true
```

# IsMarkerDataEnabled

*Return whether labeled reconstructed marker data is enabled in the DataStream.*

*See Also : EnableMarkerData, DisableMarkerData, IsSegmentDataEnabled. IsUnlabeledMarkerDataEnabled, IsDeviceDataEnabled*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Enabled | boolean | Whether the data is enabled. |

C++

```
// class Output_IsMarkerDataEnabled
// {
// public:
//   bool Enabled;
// };
//
// Output_IsMarkerDataEnabled IsMarkerDataEnabled() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_IsMarkerDataEnabled Output = MyClient.IsMarkerDataEnabled();
                                    // Output.Enabled == false
MyClient.EnableMarkerData();
Output_IsMarkerDataEnabled Output = MyClient.IsMarkerDataEnabled();
                                    // Output.Enabled == true
```

MATLAB

```
% [Output] = IsMarkerDataEnabled()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.IsMarkerDataEnabled(); % Output.Enabled == false
MyClient.EnableMarkerData();
Output = MyClient.IsMarkerDataEnabled(); % Output.Enabled == true
```

.NET

```
// public class Output_IsMarkerDataEnabled
// {
//   public bool Enabled;
// };
// Output_IsMarkerDataEnabled IsMarkerDataEnabled();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_IsMarkerDataEnabled Output = MyClient.IsMarkerDataEnabled();
                                    // Output.Enabled == false
MyClient.EnableMarkerData();
Output_IsMarkerDataEnabled Output = MyClient.IsMarkerDataEnabled();
                                    // Output.Enabled == true
```

# IsUnlabeledMarkerDataEnabled

*Return whether unlabeled marker data is enabled in the DataStream.*

*See Also : EnableUnlabeledMarkerData,DisableUnlabeledMarkerData, IsSegmentDataEnabled. IsMarkerDataEnabled, IsDeviceDataEnabled*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Enabled | boolean | Whether the data is enabled. |

C++

```
// class Output_IsUnlabeledMarkerDataEnabled
// {
// public:
//   bool Enabled;
// };
//
// Output_IsUnlabeledMarkerDataEnabled
//                 IsUnlabeledMarkerDataEnabled() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_IsUnlabeledMarkerDataEnabled Output =
      MyClient.IsUnlabeledMarkerDataEnabled(); // Output.Enabled ==
false
MyClient.EnableUnlabeledMarkerData();
Output_IsUnlabeledMarkerDataEnabled Output =
      MyClient.IsUnlabeledMarkerDataEnabled(); // Output.Enabled ==
true
```

MATLAB

```
% [Output] = IsUnlabeledMarkerDataEnabled()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.IsUnlabeledMarkerDataEnabled(); % Output.Enabled ==
false
MyClient.EnableUnlabeledMarkerData();
Output = MyClient.IsUnlabeledMarkerDataEnabled(); % Output.Enabled ==
true
```

.NET

```
// public class Output_IsUnlabeledMarkerDataEnabled
// {
//   public bool Enabled;
// };
//
// Output_IsUnlabeledMarkerDataEnabled IsUnlabeledMarkerDataEnabled();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_IsUnlabeledMarkerDataEnabled Output =
              MyClient.IsMarkerDataEnabled(); // Output.Enabled ==
false
MyClient.EnableUnlabeledMarkerData();
Output_IsUnlabeledMarkerDataEnabled Output =
     MyClient.IsUnlabeledMarkerDataEnabled(); // Output.Enabled == true
```

# IsMarkerRayDataEnabled

*Return whether unlabeled marker data is enabled in the DataStream.*

*See Also : EnableMarkerRayData,DisableMarkerRayData, IsSegmentDataEnabled. IsMarkerDataEnabled, IsDeviceDataEnabled*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Enabled | boolean | Whether the data is enabled. |

C++

```
// class Output_IsMarkerRayDataEnabled
// {
// public:
//   bool Enabled;
// };
//
// Output_IsMarkerRayDataEnabled
//                IsMarkerRayDataEnabled() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_IsMarkerRayDataEnabled Output =
      MyClient.IsMarkerRayDataEnabled(); // Output.Enabled == false
MyClient.EnableMarkerRayData();
Output_IsMarkerRayDataEnabled Output =
      MyClient.IsMarkerRayDataEnabled(); // Output.Enabled == true
```

MATLAB

```
% [Output] = IsMarkerRayDataEnabled()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.IsMarkerRayDataEnabled(); % Output.Enabled == false
MyClient.EnableMarkerRayData();
Output = MyClient.IsMarkerRayDataEnabled(); % Output.Enabled == true
```

.NET

```
// public class Output_IsMarkerRayDataEnabled
// {
//   public bool Enabled;
// };
//
// Output_IsMarkerRayDataEnabled IsMarkerRayDataEnabled();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_IsMarkerRayDataEnabled Output =
              MyClient.IsMarkerDataEnabled(); // Output.Enabled ==
false
MyClient.EnableMarkerRayData();
Output_IsMarkerRayDataEnabled Output =
     MyClient.IsMarkerRayDataEnabled(); // Output.Enabled == true
```

# IsDeviceDataEnabled

*Return whether ForcePlate, EMG, and other device data is enabled in the data stream.*

*See Also : EnableDeviceData, DisableDeviceData, IsSegmentDataEnabled. IsMarkerDataEnabled, IsUnlabeledMarkerDataEnabled*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Enabled | boolean | Whether the data is enabled. |

C++

```
// class Output_IsDeviceDataEnabled
// {
// public:
//   bool Enabled;
// };
//
// Output_IsDeviceDataEnabled IsDeviceDataEnabled() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_IsDeviceDataEnabled Output = MyClient.IsDeviceDataEnabled();
                                    // Output.Enabled == false
MyClient.EnableDeviceData();
Output_IsDeviceDataEnabled Output = MyClient.IsDeviceDataEnabled();
                                    // Output.Enabled == true
```

MATLAB

```
% [Output] = IsDeviceDataEnabled()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.IsDeviceDataEnabled(); % Output.Enabled == false
MyClient.EnableDeviceData();
Output = MyClient.IsDeviceDataEnabled(); % Output.Enabled == true
```

.NET

```
// public class Output_IsDeviceDataEnabled
// {
//   public bool Enabled;
// };
// Output_IsDeviceDataEnabled IsDeviceDataEnabled();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_IsDeviceDataEnabled Output = MyClient.IsDeviceDataEnabled();
                                    // Output.Enabled == false
MyClient.EnableDeviceData();
Output_IsDeviceDataEnabled Output = MyClient.IsDeviceDataEnabled();
                                    // Output.Enabled == true
```

# IsCentroidDataEnabled

*Return whether Centroid data is enabled in the data stream.*
*See Also : EnableCentroidData, DisableCentroidData*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Enabled | boolean | Whether the data is enabled. |

C++

```
// class Output_IsCentroidDataEnabled
// {
// public:
//   bool Enabled;
// };
//
// Output_IsCentroidDataEnabled IsCentroidDataEnabled () const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_IsCentroidDataEnabled Output = MyClient.IsCentroidDataEnabled
();
                                    // Output.Enabled == false
MyClient.EnableCentroidData();
Output_IsCentroidDataEnabled Output = MyClient.IsCentroidDataEnabled
();
                                    // Output.Enabled == true
```

MATLAB

```
% [Output] = IsCentroidDataEnabled ()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.IsCentroidDataEnabled (); % Output.Enabled == false
MyClient.EnableCentroidData();
Output = MyClient.IsCentroidDataEnabled (); % Output.Enabled == true
```

.NET

```
// public class Output_IsCentroidDataEnabled
// {
//   public bool Enabled;
// };
//
// Output_IsCentroidDataEnabled IsCentroidDataEnabled ();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_IsCentroidDataEnabled Output = MyClient.IsCentroidDataEnabled
();
                                    // Output.Enabled == false
MyClient.EnableCentroidData();
Output_IsCentroidDataEnabled Output = MyClient.IsCentroidDataEnabled
();
                                    // Output.Enabled == true
```

# SetBufferSize

| | | | |
|---|---|---|---|
| *Set the number of frames the client should buffer.*<br>*The default value is 1, which always supplies the latest frame.*<br>*Choose a higher number to reduce the risk of missing frames between calls to GetFrame. Higher numbers may introduce latency when frames are late.*<br>*See: GetFrame* | | | |
| Input | BufferSize | Integer | The maximum number of frames to buffer. |
| C++ | `ViconDataStreamSDK::CPP::Client MyClient;`<br>`MyClient.Connect( "localhost" );`<br>`MyClient.SetBufferSize( 10 );` | | |

# SetStreamMode

*There are three modes that the SDK can operate in. Each mode has a different impact on the Client, Server, and network resources used.*

- *In "ServerPush" mode, the Server pushes every new frame of data over the network to the Client. The Server will try not to drop any frames. This results in the lowest latency we can achieve. If the Client is unable to read data at the rate it is being sent, then it is buffered, firstly in the Client, then on the TCP/IP connection, and then at the Server. Once all buffers have filled up then frames may be dropped at the Server and the performance of the Server may be affected. The GetFrame() method will return the most recently received frame if available, or block the calling thread if the most recently received frame has already been processed.*
- *In "ClientPull" mode, the Client waits for a call to GetFrame(), and then request the latest frame of data from the Server. This increases latency, because we need to send a request over the network to the Server, the Server has to prepare the frame of data for the Client, and then we need to send the data back over the network. Network bandwidth is kept to a minimum, because the Server only sends what you need. We are very unlikely to fill up our buffers, and Server performance is unlikely to be affected. The GetFrame() method blocks the calling thread until the frame has been received.*
- *"ClientPullPreFetch" is an enhancement to "ClientPull" mode. A thread in the SDK continuously and preemptively does a "ClientPull" on your behalf, storing the latest requested frame in memory. When you next call GetFrame(), the SDK returns the last requested frame which we had cached in memory. GetFrame() does not need to block the calling thread. As with normal "ClientPull", buffers are unlikely to fill up, Server performance is unlikely to be affected. Latency is slightly reduced, but network traffic may increase if we request frames on behalf of the Client which are never used.*

*The stream defaults to "ClientPull" mode as this is considered the safest option. If performance is a problem, then try "ClientPullPreFetch" followed by "ServerPush".*

*See Also : GetFrame, GetLatencyTotal*

| | | | |
|---|---|---|---|
| Input | Mode | StreamMode | `StreamMode.ServerPush`<br>`StreamMode.ClientPull`<br>`StreamMode.ClientPullPreFetch` |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected` |

C++

```
// class Output_SetStreamMode
// {
// public:
//   Result::Enum Result;
// };
// Output_SetStreamMode SetStreamMode( const StreamMode::Enum Mode );

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.SetStreamMode( ViconDataStreamSDK::CPP::StreamMode::ServerPush
);
MyClient.SetStreamMode( ViconDataStreamSDK::CPP::StreamMode::ClientPull
);
MyClient.SetStreamMode(
ViconDataStreamSDK::CPP::StreamMode::ClientPullPreFetch );
```

MATLAB

```
% [Output] = SetStreamMode( Mode );

MyClient = Client();
MyClient.Connect( 'localhost' );
MyClient.SetStreamMode( StreamMode.ServerPush );
MyClient.SetStreamMode( StreamMode.ClientPull );
MyClient.SetStreamMode( StreamMode.ClientPullPreFetch );
```

| | |
|---|---|
| .NET | `// class Output_SetStreamMode`<br>`// {`<br>`// public Result Result;`<br>`// };`<br>`//`<br>`// Output_SetStreamMode SetStreamMode( StreamMode Mode );`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient =`<br>`new ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`MyClient.SetStreamMode( ViconDataStreamSDK.DotNET.StreamMode.ServerPush );`<br>`MyClient.SetStreamMode( ViconDataStreamSDK.DotNET.StreamMode.ClientPull );`<br>`MyClient.SetStreamMode( ViconDataStreamSDK.DotNET.StreamMode.ClientPullPreFetch);` |

# SetAxisMapping

*Remaps the 3D axis.*

*Vicon Data uses a right handed co-ordinate system, with +X forward, +Y left, and +Z up. Other systems use different co-ordinate systems. The SDK can transform its data into any valid right-handed co-ordinate system by re-mapping each axis.*

*Specify the direction of your X, Y, and Z axis relative to yourself as the observer. Valid directions are "Up", "Down", "Left", "Right", "Forward", and "Backward". Note that "Forward" means moving away from you, and "Backward" is moving towards you.*

*Common usages are*

- *Z-up : SetAxisMapping( Forward, Left, Up )*
- *Y-up : SetAxisMapping( Forward, Up, Right )*

*See Also : GetAxisMapping*

| | | | |
|---|---|---|---|
| Input | XAxis | Direction | |
| | YAxis | Direction | |
| | ZAxis | Direction | |
| Output | Result | Result | `Result.Success`<br>`Result.CoLinearAxes`<br>`Result.LeftHandedAxes` |

C++

```
// class Output_SetAxisMapping
// {
// public:
//   Result::Enum Result;
// };
//
// Output_SetAxisMapping SetAxisMapping( const Direction::Enum XAxis,
//                                       const Direction::Enum YAxis,
//                                       const Direction::Enum ZAxis )

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.SetAxisMapping( ViconDataStreamSDK::CPP::Direction::Forward,
                         ViconDataStreamSDK::CPP::Direction::Left,
                         ViconDataStreamSDK::CPP::Direction::Up );
```

MATLAB

```
% [Output] = SetAxisMapping( XAxis,
%                             YAxis,
%                             ZAxis )

MyClient = Client();
MyClient.SetAxisMapping( Direction.Forward, ...
                         Direction.Left,    ...
                         Direction.Up );
```

.NET

```
// public class Output_SetAxisMapping
// {
// public Result Result;
// };
//
// Output_SetAxisMapping SetAxisMapping( Direction XAxis,
//                                       Direction YAxis,
//                                       Direction ZAxis );

ViconDataStreamSDK.DotNET.Client MyClient =
```

| | |
|---|---|
| | new ViconDataStreamSDK.DotNET.Client();<br>MyClient.SetAxisMapping( ViconDataStreamSDK.DotNET.Direction.Forward,<br>ViconDataStreamSDK.DotNET.Direction.Left,<br>ViconDataStreamSDK.DotNET.Direction.Up ); |

# GetAxisMapping

*Get the current Axis mapping.*
*See Also : SetAxisMapping*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | XAxis | Direction | |
| | YAxis | Direction | |
| | ZAxis | Direction | |

C++

```
// class Output_GetAxisMapping
// {
// public:
//   Direction::Enum XAxis;
//   Direction::Enum YAxis;
//   Direction::Enum ZAxis;
// };
// Output_GetAxisMapping GetAxisMapping() const;

ViconDataStreamSDK::CPP::Client MyClient;
Output_GetAxisMapping Output = MyClient.GetAxisMapping();
  // Output.XAxis == ViconDataStreamSDK::CPP::Direction::Forward
  // Output.YAxis == ViconDataStreamSDK::CPP::Direction::Left
  // Output.ZAxis == ViconDataStreamSDK::CPP::Direction::Up
```

MATLAB

```
% [Output] = GetAxisMapping()

MyClient = Client();
Output = MyClient.GetAxisMapping();
  % Output.XAxis == Direction.Forward
  % Output.YAxis == Direction.Left
  % Output.ZAxis == Direction.Up
```

.NET

```
// public class Output_GetAxisMapping
// {
//   public Direction XAxis;
//   public Direction YAxis;
//   public Direction ZAxis;
// };
// Output_GetAxisMapping GetAxisMapping();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
Output_GetAxisMapping Output = MyClient.GetAxisMapping();
  // Output.XAxis == ViconDataStreamSDK.DotNET.Direction.Forward
  // Output.YAxis == ViconDataStreamSDK.DotNET.Direction.Left
  // Output.ZAxis == ViconDataStreamSDK.DotNET.Direction.Up
```

# GetFrame

*Request a new frame to be fetched from the Vicon DataStream Server.*
*See Also : SetStreamMode*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected` |

C++

```
// class Output_GetFrame
// {
// public:
//   Result::Enum Result;
// };
//
// Output_GetFrame GetFrame();

ViconDataStreamSDK::CPP::Client MyClient;
Output_GetFrame Output;
Output = MyClient.GetFrame(); // Output.Result == NotConnected
MyClient.Connect( "localhost" );
Output = MyClient.GetFrame(); // Output.Result == Success
```

MATLAB

```
% [Output] = GetFrame()

MyClient = Client();
Output = MyClient.GetFrame(); // Output.Result == NotConnected
MyClient.Connect( "localhost" );
Output = MyClient.GetFrame(); // Output.Result == Success
```

.NET

```
// public class Output_GetFrame
// {
// public Result Result;
// };
//
// Output_GetFrame GetFrame();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
Output_GetFrame Output;
Output = MyClient.GetFrame(); // Output.Result == NotConnected
MyClient.Connect( "localhost" );
Output = MyClient.GetFrame(); // Output.Result == Success
```

# GetFrameNumber

*Return the number of the last frame retrieved from the DataStream.*
*See Also : GetFrame, GetTimecode*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Frame Number | unsigned integer | The frame number |

C++

```
// class Output_GetFrameNumber
// {
//  public:
//    Result::Enum Result;
//    unsigned int FrameNumber;
//  };
//
// Output_GetFrameNumber GetFrameNumber() const;
ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_GetFrameNumber Output;
Output = MyClient.GetFrameNumber(); // Output.Result == NoFrame
                                    // Output.FrameNumber == 0
MyClient.GetFrame();
Output = MyClient.GetFrameNumber(); // Output.Result == Success
                                    // Output.FrameNumber >= 1
```

MATLAB

```
% [Output] = GetFrameNumber()

MyClient = Client();
MyClient.Connect( "localhost" );
Output = MyClient.GetFrameNumber(); % Output.Result == NoFrame
                                    % Output.FrameNumber == 0
MyClient.GetFrame();
Output = MyClient.GetFrameNumber(); % Output.Result == Success
                                    % Output.FrameNumber >= 1
```

.NET

```
// class Output_GetFrameNumber
// {
//    public Result Result;
//    public uint   FrameNumber;
//  };
//
// Output_GetFrameNumber GetFrameNumber();

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_GetFrameNumber Output;
Output = MyClient.GetFrameNumber(); // Output.Result == NoFrame
                                    // Output.FrameNumber == 0
MyClient.GetFrame();
Output = MyClient.GetFrameNumber(); // Output.Result == Success
                                    // Output.FrameNumber >= 1
```

# GetLatencyTotal

| | | | |
|---|---|---|---|
| *Return the total latency in seconds introduced at various stages of the real-time pipeline. If no latency information is available then all latencies will be reported as 0.0.*<br>*See Also : GetFrame, GetTimecode, GetLatencySampleCount, GetLatencySampleName, GetLatencySampleValue* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Total | double | The total latency in seconds made from summing the other latencies. |

C++

```
// class Output_GetLatencyTotal
// {
//  public:
//    Result::Enum Result;
//    double       Total;
//  };
//
// Output_GetLatencyTotal GetLatencyTotal() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencyTotal Output = MyClient.GetLatencyTotal();
```

MATLAB

```
% [Output] = GetLatencyTotal()

MyClient = Client();
MyClient.Connect( 'localhost' );
MyClient.GetFrame();
Output = MyClient.GetLatencyTotal();
```

.NET

```
// class Output_GetLatencyTotal
// {
//    public Result Result;
//    public double Total;
//  };
//
// Output_GetLatencyTotal GetLatencyTotal();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencyTotal Output = MyClient.GetLatencyTotal();
```

# GetLatencySampleCount

| | | | |
|---|---|---|---|
| *Return the number of latency measurements that were taken at various stages of the real-time pipeline.This value can be passed into GetLatencySampleName().*<br>*See Also : GetFrame, GetTimecode, GetLatencyTotal, GetLatencySampleName, GetLatencySampleValue* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Count | unsigned int | The number of samples taken. |

C++

```
// class Output_GetLatencySampleCount
// {
//  public:
//    Result::Enum Result;
//    unsigned int Count;
//  };
//
// Output_GetLatencySampleCount GetLatencySampleCount() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencySampleCount Output = MyClient.GetLatencySampleCount();
```

MATLAB

```
% [Output] = GetLatencySampleCount()

MyClient = Client();
MyClient.Connect( 'localhost' );
MyClient.GetFrame();
Output = MyClient.GetLatencySampleCount();
```

.NET

```
// class Output_GetLatencySampleCount
// {
//    public Result Result;
//    public uint   Count;
//  };
//
// Output_GetLatencySampleCount GetLatencySampleCount();

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencySampleCount Output = MyClient.GetLatencySampleCount();
```

# GetLatencySampleName

*Return the name of a latency sample. This value can be passed into GetLatencySampleValue().*

*See Also : GetFrame, GetTimecode, GetLatencyTotal, GetLatencySampleCount, GetLatencySampleValue*

| | | | |
|---|---|---|---|
| Input | LatencySampleIndex | Unsigned int | The index of the name. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | Name | string | The name of the latency sample. |

C++

```
A valid Latency Sample Index is between 0 and GetLatencySampleCount()-1

// class Output_GetLatencySampleName
// {
//  public:
//    Result::Enum Result;
//    String       Name;
// };
//
// Output_GetLatencySampleName
//   GetLatencySampleName( const unsigned int LatencySampleIndex )
const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencySampleName Output = MyClient.GetLatencySampleName( 0 );
// Output.Name == "Data Collected"
```

MATLAB

```
A valid Latency Sample Index is between 1 and GetLatencySampleCount()

% [Output] = GetLatencySampleName()

MyClient = Client();
MyClient.Connect( 'localhost' );
MyClient.GetFrame();
Output = MyClient.GetLatencySampleName( 1 );
% Output.Name == 'Data Collected'
```

.NET

```
A valid Latency Sample Index is between 0 and GetLatencySampleCount()-1

// class Output_GetLatencySampleName
// {
//    public Result Result;
//    public string Name;
// };
// Output_GetLatencySampleName
//  GetLatencySampleName( uint LatencySampleIndex );

ViconDataStreamSDK.DotNET.Client MyClient =
                                 new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencySampleName Output = MyClient.GetLatencySampleName( 0 );
// Output.Name == "Data Collected"
```

# GetLatencySampleValue

*Return the duration of a named latency sample in seconds. This value can be passed into GetLatencySampleValue().*

*See Also : GetFrame, GetTimecode, GetLatencyTotal, GetLatencySampleCount, GetLatencySampleValue*

| | | | |
|---|---|---|---|
| Input | LatencySampleName | string | The name of the latency sample. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidLatencySampleName` |
| | Value | double | The duration of the latency in seconds. |

C++

```
// class Output_GetLatencySampleValue
// {
//  public:
//    Result::Enum Result;
//    double       Value;
//  };
// Output_GetLatencySampleValue
//   GetLatencySampleValue( const String & LatencySampleName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencySampleValue Output =
  MyClient.GetLatencySampleValue( "Data Collected" );
// Output.Value == 0.1
```

MATLAB

```
% [Output] = GetLatencySampleValue()

MyClient = Client();
MyClient.Connect( 'localhost' );
MyClient.GetFrame();
Output = MyClient.GetLatencySampleValue( 'Data Collected' );
% Output.Value == 0.1
```

.NET

```
// class Output_GetLatencySampleValue
// {
//    public Result Result;
//    public double Value;
//  };
// Output_GetLatencySampleValue
//  GetLatencySampleValue( string LatencySampleName );
ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetLatencySampleName Output =
  MyClient.GetLatencySampleValue( "Data Collected" );
// Output.Value == 0.1
```

# GetTimecode

| | | | |
|---|---|---|---|
| *Return the timecode information for the last frame retrieved from the DataStream. If the stream is valid but timecode is not available the Output will be Result.Success and the Standard will be None.*<br>*See Also : GetFrame, GetFrameNumber* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Hours | Unsigned integer | |
| | Minutes | Unsigned integer | |
| | Seconds | Unsigned integer | |
| | Frames | Unsigned integer | |
| | SubFrame | Unsigned integer | |
| | FieldFlag | Boolean | |
| | Standard | TimecodeStandard | None<br>PAL<br>NTSC<br>NTSCDrop<br>Film |
| | SubFramesPerFrame | Unsigned integer | |
| | UserBits | Unsigned integer | |

C++

```
// class Output_GetTimecode
// {
// public:
//   Result::Enum           Result;
//   unsigned int           Hours;
//   unsigned int           Minutes;
//   unsigned int           Seconds;
//   unsigned int           Frames;
//   unsigned int           SubFrame;
//   bool                   FieldFlag;
//   TimecodeStandard::Enum Standard;
//   unsigned int           SubFramesPerFrame;
//   unsigned int           UserBits;
// };
//
// Output_GetTimecode GetTimecode() const;
```

| | |
|---|---|
| | ```<br>ViconDataStreamSDK::CPP::Client MyClient;<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br>Output_GetTimecode Output = MyClient.GetTimecode();<br>``` |
| MATLAB | ```<br>% [Output] = GetTimecode()<br><br>MyClient = Client();<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br>Output = MyClient.GetTimecode();<br>``` |
| .NET | ```<br>// class Output_GetTimecode<br>// {<br>//   public Result           Result;<br>//   public uint             Hours;<br>//   public uint             Minutes;<br>//   public uint             Seconds;<br>//   public uint             Frames;<br>//   public uint             SubFrame;<br>//   public bool             FieldFlag;<br>//   public TimecodeStandard Standard;<br>//   public uint             SubFramesPerFrame;<br>//   public uint             UserBits;<br>// };<br>//<br>// Output_GetTimecode GetTimecode();<br><br>ViconDataStreamSDK.DotNET.Client MyClient =<br>                            new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br>Output_GetTimecode Output = MyClient.GetTimecode();<br>``` |

# GetFrameRate

*Return the Vicon camera system frame rate (in Hz) at the time of the last frame retrieved from the DataStream.*

*See Also : GetFrame, GetFrameNumber, GetTimecode*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | FrameRateHz | double | |

C++

```
// class Output_GetFrameRate
// {
// public:
//   Result::Enum           Result;
//   double           FrameRateHz;
// };
//
// Output_GetFrameRate GetFrameRate() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetFrameRate Output = MyClient.GetFrameRate ();
```

MATLAB

```
% [Output] = GetFrameRate()

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output = MyClient.GetFrameRate ();
```

.NET

```
// class Output_GetTimecode
// {
//   public Result           Result;
//   public double FrameRateHz;
// };
//
// Output_GetFrameRate GetFrameRate ();

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
Output_GetFrameRate Output = MyClient.GetFrameRate ();
```

# GetSubjectCount

| | | | |
|---|---|---|---|
| *Return the number of subjects in the DataStream. This information can be used in conjunction with GetSubjectName*<br>*See Also : GetSubjectName* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Subject Count | unsigned integer | The number of subjects |

C++

```
// class Output_GetSubjectCount
// {
// public:
//   Result::Enum Result;
//   unsigned int SubjectCount;
// };
//
// Output_GetSubjectCount GetSubjectCount() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
Output_GetSubjectCount Output;
Output = MyClient.GetSubjectCount(); // Output.Result == NoFrame
                                     // Ooutput.SubjectCount == 0
MyClient.GetFrame();
Output = MyClient.GetSubjectCount(); // Output.Result == Success
                                     // Output.SubjectCount >= 0
```

MATLAB

```
% [Output] = GetSubjectCount()

MyClient = Client();
MyClient.Connect( 'localhost' );
Output = MyClient.GetSubjectCount(); % Output.Result == NoFrame
                                     % Ooutput.SubjectCount == 0
MyClient.GetFrame();
Output = MyClient.GetSubjectCount(); % Output.Result == Success
                                     % Output.SubjectCount >= 0
```

.NET

```
// class Output_GetSubjectCount
// {
//   public Result Result;
//   public uint   SubjectCount;
// };
//
// Output_GetSubjectCount GetSubjectCount();
//
// Output_GetSubjectCount GetSubjectCount()

ViconDataStreamSDK.DotNET.Client MyClient =
                                 new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
Output_GetSubjectCount Output;
Output = MyClient.GetSubjectCount(); // Output.Result == NoFrame
                                     // Ooutput.SubjectCount == 0
MyClient.GetFrame();
Output = MyClient.GetSubjectCount(); // Output.Result == Success
                                     // Output.SubjectCount >= 0
```

# GetSubjectName

*Return the name of a subject. This can be passed into segment and marker functions.*
*See Also : GetSubjectCount*

| | | | |
|---|---|---|---|
| Input | Subject Index | unsigned integer | The index of the subject. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | Subject Name | string | The name of the subject |

**C++**

A valid Subject Index is between 0 and GetSubjectCount()-1

```
// class Output_GetSubjectName
// {
// public:
//   Result::Enum Result;
//   String       SubjectName;
// };
//
// Output_GetSubjectName GetSubjectName(
//                         const unsigned int SubjectIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSubjectCount OutputGSC;
OutputGSC = MyClient.GetSubjectCount(); // OutputGSC.Result == Success
                                        // OutputGSC.SubjectCount == 2

Output_GetSubjectName OutputGSN;
OutputGSN = MyClient.GetSubjectName(0);// OutputGSN.Result == Success
                                       // OutputGSN.SubjectName == "Al"
OutputGSN = MyClient.GetSubjectName(1);// OutputGSN.Result == Success
                                       // OutputGSN .SubjectName ==
"Bob"
OutputGSN = MyClient.GetSubjectName(2);// OutputGSN.Result ==
InvalidIndex
                                       // OutputGSN.SubjectName == ""
```

**MATLAB**

A valid Subject Index is between 1 and GetSubjectCount()

```
% [Output] = GetSubjectName( SubjectIndex )

MyClient = Client;
MyClient.Connect( 'localhost' );
MyClient.GetFrame();

OutputGSC = MyClient.GetSubjectCount(); % OutputGSC.Result == Success
                                        % OutputGSC.SubjectCount == 2

OutputGSN = MyClient.GetSubjectName(1); % OutputGSN.Result == Success
                                        % OutputGSN.SubjectName == 'Al'
OutputGSN = MyClient.GetSubjectName(2); % OutputGSN.Result == Success
                                        % OutputGSN .SubjectName ==
'Bob'
OutputGSN = MyClient.GetSubjectName(3); % OutputGSN.Result ==
InvalidIndex
                                       // OutputGSN.SubjectName == ''
```

| | |
|---|---|
| .NET | A valid Subject Index is between 0 and GetSubjectCount()-1<br>`// public class Output_GetSubjectName`<br>`// {`<br>`//   public Result Result;`<br>`//   public string SubjectName;`<br>`// };`<br>`//`<br>`// Output_GetSubjectName GetSubjectName( uint SubjectIndex );`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient =`<br>`  new ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`MyClient.GetFrame();`<br><br>`Output_GetSubjectCount OutputGSC;`<br>`OutputGSC = MyClient.GetSubjectCount(); // OutputGSC.Result == Success`<br>`  // OutputGSC.SubjectCount == 2`<br><br>`Output_GetSubjectName OutputGSN;`<br>`OutputGSN = MyClient.GetSubjectName(0);// OutputGSN.Result == Success`<br>`  // OutputGSN.SubjectName == "Al"`<br>`OutputGSN = MyClient.GetSubjectName(1);// OutputGSN.Result == Success`<br>`  // OutputGSN .SubjectName ==`<br>`"Bob"`<br>`OutputGSN = MyClient.GetSubjectName(2);// OutputGSN.Result ==`<br>`InvalidIndex`<br>`  // OutputGSN.SubjectName == ""` |

# GetSubjectRootSegmentName

*Return the name of the root segment for a specified subject. This can be passed into segment functions.The root segment is the ancestor of all other segments in the subject.*

*See Also : GetSegmentCount, GetSegmentParentName, GetSegmentChildCount, GetSegmentChildName*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName` |
| | Segment Name | string | The name of the root segment |

C++

```
// class Output_GetSubjectRootSegmentName
// {
// public:
//   Result::Enum Result;
//   String       SegmentName;
// };
//
// Output_GetSubjectRootSegmentName GetSubjectRootSegmentName(
//                       const String & SubjectName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSubjectRootSegmentName Output;
Output = MyClient.GetSubjectRootSegmentName( "Bob" );
                            // Output.Result == Success
                            // Output.SegmentName == "Pelvis"
```

MATLAB

```
% [Output] = GetSubjectRootSegmentName( SubjectName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output = MyClient.GetSubjectRootSegmentName( "Bob" );
                            % Output.Result == Success
                            % Output.SegmentName == "Pelvis"
```

.NET

```
// public class Output_GetSubjectRootSegmentName
// {
//   public Result Result;
//   public string SegmentName;
// };
//
// Output_GetSubjectRootSegmentName GetSubjectRootSegmentName(
//                                         string SubjectName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();
```

| | |
|---|---|
| | `Output_GetSubjectRootSegmentName Output;`<br>`Output = MyClient.GetSubjectRootSegmentName( "Bob" );`<br>`// Output.Result == Success`<br>`// Output.SegmentName == "Pelvis"` |

# GetSegmentCount

*Return the number of segments for a specified subject in the DataStream. This information can be used in conjunction with GetSegmentName*

*See Also : GetSubjectName, GetSegmentName*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName` |
| | Segment Count | unsigned integer | The number of segments |

C++

```
// class Output_GetSegmentCount
// {
// public:
//   Result::Enum Result;
//   unsigned int SegmentCount;
// };
//
// Output_GetSegmentCount GetSegmentCount(
//                      const String  & SubjectName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableSegmentData();
MyClient.Connect( "localhost" );

Output_GetSegmentCount Output;
Output = MyClient.GetSegmentCount( "Bob" ); // Output.Result == NoFrame
                                            // Output.SegmentCount == 0
MyClient.GetFrame();

Output = MyClient.GetSegmentCount( "Al" );  // Output.Result ==
                                            //
InvalidSubjectName
                                            // Output.SegmentCount == 0

Output = MyClient.GetSegmentCount( "Bob" ); // Output.Result == Success
                                            // Output.SegmentCount >= 0
```

MATLAB

```
% [Output] = GetSegmentCount( SubjectName )

MyClient = Client();
MyClient.EnableSegmentData();
MyClient.Connect( "localhost" );

Output = MyClient.GetSegmentCount( "Bob" ); % Output.Result == NoFrame
                                            % Output.SegmentCount == 0
MyClient.GetFrame();

Output = MyClient.GetSegmentCount( "Al" );  % Output.Result ==
                                            %
InvalidSubjectName
                                            % Output.SegmentCount == 0

Output = MyClient.GetSegmentCount( "Bob" ); % Output.Result == Success
                                            % Output.SegmentCount >= 0
```

.NET

```
// public class Output_GetSegmentCount
// {
//   public Result Result;
```

```
//   public uint   SegmentCount;
// };
//
// Output_GetSegmentCount GetSegmentCount( string SubjectName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.EnableSegmentData();
MyClient.Connect( "localhost" );

Output_GetSegmentCount Output;
Output = MyClient.GetSegmentCount( "Bob" ); // Output.Result == NoFrame
                                            // Output.SegmentCount == 0
MyClient.GetFrame();

Output = MyClient.GetSegmentCount( "Al" );  // Output.Result ==
                                            //
InvalidSubjectName
                                            // Output.SegmentCount == 0

Output = MyClient.GetSegmentCount( "Bob" ); // Output.Result == Success
                                            // Output.SegmentCount >= 0
```

# GetSegmentName

*Return the name of a subject segment specified by index.*

*See Also : GetSegmentCount, GetSegmentChildCount, GetSegmentChildName, GetSubjectRootSegmentName*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Index | unsigned int | The index of the segment |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidIndex` |
| | Segment Name | string | The name of the parent segment or an empty string if it is the root segment. |

C++

```
// class Output_GetSegmentName
// {
// public:
//   Result::Enum Result;
//   String       SegmentName;
// };
//
// Output_GetSegmentName GetSegmentName(
//                       const String & SubjectName,
//                       const unsigned int SegmentIndex ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentName Output;
// SegmentIndex should be between 0 and GetSegmentCount() - 1
Output = MyClient.GetSegmentName( "Bob", 0 );
```

MATLAB

```
% [Output] = GetSegmentName( SubjectName, SegmentIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

% SegmentIndex should be between 1 and GetSegmentCount()
Output = MyClient.GetSegmentName( "Bob", 1 );
```

.NET

```
// public class Output_GetSegmentName
// {
//   public Result Result;
//   public unsiged int SegmentIndex;
// };
//
// Output_GetSegmentName GetSegmentName(
//                                       string SubjectName,
//                                       unsigned int SegmentIndex );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
```

| | |
|---|---|
| | ```<br>MyClient.Connect( "localhost" );<br>MyClient.EnableSegmentData();<br>MyClient.GetFrame();<br><br>Output_GetSegmentParentName Output;<br><br>// SegmentIndex should be between 0 and GetSegmentCount() - 1<br>Output = MyClient.GetSegmentName( "Bob", 0 );<br>``` |

# GetSegmentChildCount

*Return the name of a child segment for a specified subject segment.This can be passed into segment functions.*

*See Also : GetSegmentCount*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the parent segment. |
| | Segment Index | unsigned integer | The index of the child segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName`<br>`Result.InvalidIndex` |
| | Segment Name | string | The name of the child segment |

C++

A valid Segment Index is between 0 and GetSegmentChildCount()-1

```
// class Output_GetSegmentChildName
// {
// public:
//   Result::Enum Result;
//   String       SegmentName;
// };
//
// Output_GetSegmentChildName GetSegmentName(
//                       const String      & SubjectName,
//                       const String      & SegmentName,
//                       const unsigned int  SegmentIndex ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentChildCount OutputGSCC;
OutputGSCC = MyClient.GetSegmentChildCount( "Bob", "Pelvis" );
                            // OutputGSCC.Result == Success
                            // OutputGSCC.SegmentCount == 2

Output_GetSegmentChildName  OutputGSCN;
OutputGSCN = MyClient.GetSegmentName( "Alice", 0 );
                            // OutputGSN.Result == InvalidSubjectName
                            // OutputGSN.SegmentName == ""

OutputGSCN = MyClient.GetSegmentName( "Bob", "Pelvis", 0 );
                            // OutputGSCN.Result == Success
                            // OutputGSCN.SegmentName == "LFemur"
OutputGSCN = MyClient.GetSegmentName( "Bob", "Pelvis", 1 );
                            // OutputGSCN.Result == Success
                            // OutputGSCN.SegmentName == "RFemur"
OutputGSCN = MyClient.GetSegmentName( "Bob", "Pelvis", 2 );
                            // OutputGSCN.Result == InvalidIndex
                            // OutputGSCN.SegmentName == ""
                            // (no third segment)
```

MATLAB

A valid Segment Index is between 1 and GetSegmentChildCount()

```
% [Output] = GetSegmentChildName( SubjectName, SegmentName, SegmentIndex
)

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

OutputGSCC = MyClient.GetSegmentChildCount( "Bob", "Pelvis" );
                            % OutputGSCC.Result == Success
                            % OutputGSCC.SegmentCount == 2

OutputGSCN = MyClient.GetSegmentChildName( "Alice", "Pelvis", 1 );
                            % OutputGSCN.Result == InvalidSubjectName
                            % OutputGSCN.SegmentName == ""

OutputGSCN = MyClient.GetSegmentChildName( "Bob", "Pelvis", 1 );
                            % OutputGSCN.Result == Success
                            % OutputGSCN.SegmentName == "LFemur"
OutputGSCN = MyClient.GetSegmentChildName( "Bob", "Pelvis", 2 );
                            % OutputGSCN.Result == Success
                            % OutputGSCN.SegmentName == "RFemur"
OutputGSCN = MyClient.GetSegmentChildName( "Bob", "Pelvis", 3 );
                            % OutputGSCN.Result == InvalidIndex
                            % OutputGSCN.SegmentName == ""
                            % (no third segment)
```

.NET

A valid Segment Index is between 0 and GetSegmentChildCount()-1

```
// public class Output_GetSegmentChildName
// {
//   public Result Result;
//   public string SegmentName;
// };
// Output_GetSegmentChildName GetSegmentChildName( string SubjectName,
//                                                 string SegmentName,
//                                                 uint   SegmentIndex
);

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentChildCount OutputGSCC;
OutputGSCC = MyClient.GetSegmentChildCount( "Bob", "Pelvis" );
                            // OutputGSCC.Result == Success
                            // OutputGSCC.SegmentCount == 2

Output_GetSegmentChildName  OutputGSCN;
OutputGSCN = MyClient.GetSegmentChildName( "Alice", "Pelvis", 0 );
                            // OutputGSCN.Result == InvalidSubjectName
                            // OutputGSCN.SegmentName == ""

OutputGSCN = MyClient.GetSegmentChildName( "Bob", "Pelvis", 0 );
                            // OutputGSCN.Result == Success
                            // OutputGSCN.SegmentName == "LFemur"
OutputGSCN = MyClient.GetSegmentChildName( "Bob", "Pelvis", 1 );
                            // OutputGSCN.Result == Success
                            // OutputGSCN.SegmentName == "RFemur"
OutputGSCN = MyClient.GetSegmentChildName( "Bob", "Pelvis", 2 );
                            // OutputGSCN.Result == InvalidIndex
                            // OutputGSCN.SegmentName == ""
                            // (no third segment)
```

# GetSegmentChildName

| | | | |
|---|---|---|---|
| *Return the name of the child segment for a specified subject segment and index.*<br>*See Also : GetSegmentCount, GetSegmentChildCount, GetSegmentChildName, GetSubjectRootSegmentName* | | | |
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment |
| | Segment Index | unsigned int | The index of the child segment |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName`<br>`Result.InvalidIndex` |
| | Segment Name | string | The name of the child segment |

C++

```
// class Output_GetSegmentChildName
// {
// public:
//   Result::Enum Result;
//   String       SegmentName;
// };
//
// Output_GetSegmentChildName GetSegmentParentName(
//                       const String & SubjectName,
//                       const String & SegmentName,
//                       const unsigned int SegmentIndex ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentChildName Output;
// Segment index should be between 0 and GetSegmentChildCount() - 1
Output = MyClient.GetSegmentChildName( "Bob", "Pelvis", 0 );
```

MATLAB

```
% [Output] = GetSegmentChildName( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

// Segment index should be between 1 and GetSegmentChildCount()
Output = MyClient.GetSegmentChildName( "Bob", "Pelvis", 1 );
```

.NET

```
// public class Output_GetSegmentChildName
// {
//   public Result Result;
//   public string SegmentName;
// };
//
// Output_GetSegmentChildName GetSegmentChildName(
//                                       string SubjectName,
```

```
//                                          string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                 new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentChildName Output;
// Segment index should be between 0 and GetSegmentChildCount() - 1
Output = MyClient.GetSegmentChildName( "Bob", "Pelvis", 0 );
```

# GetSegmentParentName

| | | | |
|---|---|---|---|
| *Return the name of the parent segment for a specified subject segment. If the specified segment is the root segment of the subject then it will return an empty string.*<br>*See Also : GetSegmentCount, GetSegmentChildCount, GetSegmentChildName, GetSubjectRootSegmentName* | | | |
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Segment Name | string | The name of the parent segment or an empty string if it is the root segment. |

C++

```
// class Output_GetSegmentParentName
// {
// public:
//   Result::Enum Result;
//   String       SegmentName;
// };
//
// Output_GetSegmentParentName GetSegmentParentName(
//                       const String & SubjectName,
//                       const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentParentName Output;
Output = MyClient.GetSegmentParentName( "Bob", "Pelvis" );
                            // Output.Result == Success
                            // Output.SegmentName == ""
                            // This is the root segment
Output = MyClient.GetSegmentParentName( "Bob", "LFemur" );
                            // Output.Result == Success
                            // Output.SegmentName == "Pelvis"
```

MATLAB

```
% [Output] = GetSegmentParentName( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output = MyClient.GetSegmentParentName( "Bob", "Pelvis" );
                            % Output.Result == Success
                            % Output.SegmentCount == ""
                            % This is the root segment
Output = MyClient.GetSegmentParentName( "Bob", "LFemur" );
                            % Output.Result == Success
                            % Output.SegmentCount == "Pelvis"
```

.NET

```
// public class Output_GetSegmentParentName
// {
```

```
//   public Result Result;
//   public string SegmentName;
// };
//
// Output_GetSegmentParentName GetSegmentParentName(
//                                         string SubjectName,
//                                         string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentParentName Output;
Output = MyClient.GetSegmentParentName( "Bob", "Pelvis" );
                              // Output.Result == Success
                              // Output.SegmentName == ""
                              // This is the root segment
Output = MyClient.GetSegmentParentName( "Bob", "LFemur" );
                              // Output.Result == Success
                              // Output.SegmentName == "Pelvis"
```

# GetSegmentStaticTranslation

*Return thestatic pose translation of a subject segment.*

*See Also : GetSegmentStaticRotationHelical, GetSegmentStaticRotationMatrix, GetSegmentStaticRotationQuaternion, GetSegmentStaticRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Translation | double[3] | The translation of the segment |

C++

```
// class Output_GetSegmentStaticTranslation
// {
// public:
//   Result::Enum Result;
//   double       Translation[ 3 ];
// };
//
// Output_GetSegmentStaticTranslation GetSegmentStaticTranslation(
//        const String & SubjectName,
//        const String & SegmentName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentStaticTranslation Output =
 MyClient.GetSegmentStaticTranslation( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentStaticTranslation( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output = MyClient.GetSegmentStaticTranslation( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentStaticTranslation
// {
//   public Result   Result;
//   public double[] Translation;
// };
//
// Output_GetSegmentStaticTranslation GetSegmentStaticTranslation(
//        string  SubjectName,
//        string  SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
```

| | |
|---|---|
| | MyClient.EnableSegmentData();<br>MyClient.GetFrame();<br><br>Output_GetSegmentStaticTranslation Output =<br>MyClient.GetSegmentStaticTranslations( "Alice", "Pelvis" ); |

# GetSegmentStaticRotationHelical

*Return thestatic pose rotation of a subject segment in helical co-ordinates.*

*The helical co-ordinates represent a vector whose length is the amount of rotation in radians, and the direction is the axis about which to rotate.*

*See Also : GetSegmentStaticTranslation, GetSegmentStaticRotationMatrix, GetSegmentStaticRotationQuaternion, GetSegmentStaticRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[3] | The rotation of the segment |

C++

```
// class Output_GetSegmentStaticRotationHelical
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 3 ];
// };
//
// Output_GetSegmentStaticRotationHelical
//   GetSegmentStaticRotationHelical(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentStaticRotationHelical Output =
  MyClient.GetSegmentStaticRotationHelical( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentStaticRotationHelical( SubjectName, SegmentName
)

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentStaticRotationHelical( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentStaticRotationHelical
// {
//   public Result   Result;
//   public double[] Rotation;
// };
//
// Output_GetSegmentStaticRotationHelical
//   GetSegmentStaticRotationHelical( string SubjectName,
//                                    string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
```

| | |
|---|---|
| | new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br><br>Output_GetSegmentStaticRotationHelical Output =<br>MyClient.GetSegmentStaticRotationHelical( "Alice", "Pelvis" ); |

# GetSegmentStaticRotationMatrix

*Return thestatic pose rotation of a subject segment as a 3x3 row-major matrix.*

*See Also : GetSegmentStaticTranslation, GetSegmentStaticRotationHelical, GetSegmentStaticRotationQuaternion, GetSegmentStaticRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Success | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[9] | The rotation of the segment |

C++

```
// class Output_GetSegmentStaticRotationMatrix
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 9 ];
// };
//
// Output_GetSegmentStaticRotationMatrix
//   GetSegmentStaticRotationMatrix(
//        const String & SubjectName,
//        const String & SegmentName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentStaticRotationMatrix Output =
  MyClient.GetSegmentStaticRotationMatrix( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentStaticRotationMatrix( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentStaticRotationMatrix( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentStaticRotationMatrix
// {
//   public Result   Result;
//   public double[] Rotation;
// };
//
// Output_GetSegmentStaticRotationMatrix
//   GetSegmentStaticRotationMatrix( string SubjectName,
//                                   string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();
```

| | |
|---|---|
| | `Output_GetSegmentStaticRotationMatrix Output =`<br>`  MyClient.GetSegmentStaticRotationMatrix( "Alice", "Pelvis" );` |

# GetSegmentStaticRotationQuaternion

*Return the static pose rotation of a subject segment in quaternion co-ordinates.*

*The quaternion is of the form (x, y, z, w) where w is the real component and x, y & z are the imaginary components.* ***N.B. This is different from that used in many other applications, which use (w, x, y, z).***

*See Also : GetSegmentStaticTranslation, GetSegmentStaticRotationHelical, GetSegmentStaticRotationMatrix, GetSegmentStaticRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[4] | The rotation of the segment |

C++

```
// class Output_GetSegmentStaticRotationQuaternion
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 4 ];
// };
//
// Output_GetSegmentStaticRotationQuaternion
//    GetSegmentStaticRotationQuaternion(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentStaticRotationQuaternion Output =
   MyClient.GetSegmentStaticRotationQuaternion( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentStaticRotationQuaternion( SubjectName,
SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentStaticRotationQuaternion( "Alice", "Pelvis"
);
```

.NET

```
// public class Output_GetSegmentStaticRotationQuaternion
// {
//   public Result   Result;
//   public double[] Rotation;
// };
//
// Output_GetSegmentStaticRotationQuaternion
//   GetSegmentStaticRotationQuaternion( string SubjectName,
//                                       string SegmentName );
```

| | |
|---|---|
| | ```<br>ViconDataStreamSDK.DotNET.Client MyClient =<br>new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br><br>Output_GetSegmentStaticRotationQuaternion Output =<br>MyClient.GetSegmentStaticRotationQuaternion( "Alice", "Pelvis" );<br>``` |

# GetSegmentStaticRotationEulerXYZ

*Return the static pose rotation of a subject segment in EulerXYZ co-ordinates.*

*See Also : GetSegmentStaticTranslation, GetSegmentStaticRotationHelical, GetSegmentStaticRotationMatrix, GetSegmentStaticRotationQuaternion, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[3] | The rotation of the segment |

C++

```
// class Output_GetSegmentStaticRotationEulerXYZ
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 3 ];
// };
//
// Output_GetSegmentStaticRotationEulerXYZ
//   GetSegmentStaticRotationEulerXYZ(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentStaticRotationEulerXYZ Output =
  MyClient.GetSegmentStaticRotationEulerXYZ( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentStaticRotationEulerXYZ( SubjectName, SegmentName
)

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentStaticRotationEulerXYZ( "Alice", "Pelvis"
);
```

.NET

```
// public class Output_GetSegmentStaticRotationEulerXYZ
// {
//   public Result   Result;
//   public double[] Rotation;
// };
//
// Output_GetSegmentStaticRotationEulerXYZ
//   GetSegmentStaticRotationEulerXYZ( string SubjectName,
//                                     string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
```

| | |
|---|---|
| | `MyClient.GetFrame();`<br><br>`Output_GetSegmentStaticRotationEulerXYZ Output =`<br>`  MyClient.GetSegmentStaticRotationEulerXYZ( "Alice", "Pelvis" );` |

# GetSegmentGlobalTranslation

*Return the translation of a subject segment in global co-ordinates.*

*The Translation is of the form ( x, y, z ) where x, y & z are in Millimeters with respect to the global origin.*

*See Also : GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Translation | double[3] | The translation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Translation will be [0,0,0] |

C++

```
// class Output_GetSegmentGlobalTranslation
// {
// public:
//   Result::Enum Result;
//   double       Translation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentGlobalTranslation GetSegmentGlobalTranslation(
//        const String & SubjectName,
//        const String & SegmentName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentGlobalTranslation Output =
 MyClient.GetSegmentGlobalTranslation( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentGlobalTranslation( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output = MyClient.GetSegmentGlobalTranslation( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentGlobalTranslation
// {
```

```
//   public Result   Result;
//   public double[] Translation;
//   public bool     Occluded;
// };
//
// Output_GetSegmentGlobalTranslation GetSegmentGlobalTranslation(
//        string  SubjectName,
//        string  SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentGlobalTranslation Output =
 MyClient.GetSegmentGlobalTranslations( "Alice", "Pelvis" );
```

# GetSegmentGlobalRotationHelical

*Return the rotation of a subject segment in global helical co-ordinates.*

*See Also : GetSegmentGlobalTranslation, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[3] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Rotation will be [0,0,0] |

C++

```
// class Output_GetSegmentGlobalRotationHelical
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentGlobalRotationHelical
//   GetSegmentGlobalRotationHelical(
//         const String & SubjectName,
//         const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationHelical Output =
  MyClient.GetSegmentGlobalRotationHelical( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentGlobalRotationHelical( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentGlobalRotationHelical( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentGlobalRotationHelical
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
// Output_GetSegmentGlobalRotationHelical
```

```
//   GetSegmentGlobalRotationHelical( string SubjectName,
//                                    string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                               new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationHelical Output =
  MyClient.GetSegmentGlobalRotationHelical( "Alice", "Pelvis" );
```

# GetSegmentGlobalRotationMatrix

*Return the rotation of a subject segment as a 3x3 row-major matrix in global co-ordinates.*

*See Also : GetSegmentGlobalTranslation, GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Success | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[9] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. |

C++

```
// class Output_GetSegmentGlobalRotationMatrix
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 9 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentGlobalRotationMatrix
//   GetSegmentGlobalRotationMatrix(
//         const String & SubjectName,
//         const String & SegmentName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationMatrix Output =
  MyClient.GetSegmentGlobalRotationMatrix( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentGlobalRotationMatrix( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentGlobalRotationMatrix( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentGlobalRotationMatrix
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
// Output_GetSegmentGlobalRotationMatrix
//   GetSegmentGlobalRotationMatrix( string SubjectName,
//                                   string SegmentName );
```

| | |
|---|---|
| | ```<br>ViconDataStreamSDK.DotNET.Client MyClient =<br>                              new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br><br>Output_GetSegmentGlobalRotationMatrix Output =<br>  MyClient.GetSegmentGlobalRotationMatrix( "Alice", "Pelvis" );<br>``` |

# GetSegmentGlobalRotationQuaternion

*Return the rotation of a subject segment in global quaternion co-ordinates.*

*The quaternion is of the form (x, y, z, w) where w is the real component and x, y & z are the imaginary components.* ***N.B. This is different from that used in many other applications, which use (w, x, y, z).***

*See Also : GetSegmentGlobalTranslation, GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationEulerXYZ, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[4] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Rotation will be [0,0,0,0] |

C++

```
// class Output_GetSegmentGlobalRotationQuaternion
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 4 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentGlobalRotationQuaternion
//    GetSegmentGlobalRotationQuaternion(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationQuaternion Output =
   MyClient.GetSegmentGlobalRotationQuaternion( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentGlobalRotationQuaternion( SubjectName,
SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentGlobalRotationQuaternion( "Alice", "Pelvis"
);
```

.NET

```
// public class Output_GetSegmentGlobalRotationQuaternion
// {
```

```
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
// Output_GetSegmentGlobalRotationQuaternion
//   GetSegmentGlobalRotationQuaternion( string SubjectName,
//                                       string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationQuaternion Output =
   MyClient.GetSegmentGlobalRotationQuaternion( "Alice", "Pelvis" );
```

# GetSegmentGlobalRotationEulerXYZ

*Return the rotation of a subject segment in global EulerXYZ co-ordinates.*

*See Also : GetSegmentGlobalTranslation, GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationQuaternion, GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[3] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Rotation will be [0,0,0] |

C++

```
// class Output_GetSegmentGlobalRotationEulerXYZ
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentGlobalRotationEulerXYZ
//   GetSegmentGlobalRotationEulerXYZ(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationEulerXYZ Output =
  MyClient.GetSegmentGlobalRotationEulerXYZ( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentGlobalRotationEulerXYZ( SubjectName, SegmentName
)

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentGlobalRotationEulerXYZ( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentGlobalRotationEulerXYZ
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
```

```
// Output_GetSegmentGlobalRotationEulerXYZ
//   GetSegmentGlobalRotationEulerXYZ( string SubjectName,
//                                     string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentGlobalRotationEulerXYZ Output =
  MyClient.GetSegmentGlobalRotationEulerXYZ( "Alice", "Pelvis" );
```

# GetSegmentLocalTranslation

*Return the translation of a subject segment in local co-ordinates relative to its parent segment.*

*See Also : GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ, GetSegmentGlobalTranslation,GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Translation | double[3] | The translation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Translation will be [0,0,0] |

C++

```
// class Output_GetSegmentLocalTranslation
// {
// public:
//   Result::Enum Result;
//   double       Translation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentLocalTranslation GetSegmentLocalTranslation(
//        const String & SubjectName,
//        const String & SegmentName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentLocalTranslation Output =
 MyClient.GetSegmentLocalTranslation( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentLocalTranslation( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output = MyClient.GetSegmentLocalTranslation( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentLocalTranslation
// {
//   public Result   Result;
//   public double[] Translation;
//   public bool     Occluded;
// };
```

```
//
// Output_GetSegmentLocalTranslation GetSegmentLocalTranslation(
//        string  SubjectName,
//        string  SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableSegmentData();
MyClient.GetFrame();

Output_GetSegmentLocalTranslation Output =
 MyClient.GetSegmentLocalTranslations( "Alice", "Pelvis" );
```

# GetSegmentLocalRotationHelical

*Return the rotation of a subject segment in local helical co-ordinates relative to its parent segment.*

*See Also : GetSegmentLocalTranslation, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ, GetSegmentGlobalTranslation,GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[3] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Rotation will be [0,0,0] |

C++

```
// class Output_GetSegmentLocalRotationHelical
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentLocalRotationHelical
//   GetSegmentLocalRotationHelical(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationHelical Output =
  MyClient.GetSegmentLocalRotationHelical( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentLocalRotationHelical( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentLocalRotationHelical( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentLocalRotationHelical
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
```

```
// Output_GetSegmentLocalRotationHelical
//   GetSegmentLocalRotationHelical( string SubjectName,
//                                   string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationHelical Output =
  MyClient.GetSegmentLocalRotationHelical( "Alice", "Pelvis" );
```

# GetSegmentLocalRotationMatrix

*Return the rotation row-major matrix of a subject segment in local co-ordinates relative to its parent segment.*

*See Also : GetSegmentLocalTranslation, GetSegmentLocalRotationQuaternion, GetSegmentLocalRotationEulerXYZ, GetSegmentGlobalTranslation,GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix , GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[9] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. |

C++

```
// class Output_GetSegmentLocalRotationMatrix
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 9 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentLocalRotationMatrix
//   GetSegmentLocalRotationMatrix(
//        const String & SubjectName,
//        const String & SegmentName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationMatrix Output =
  MyClient.GetSegmentLocalRotationMatrix( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentLocalRotationMatrix( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentLocalRotationMatrix( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentLocalRotationMatrix
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
```

```
// Output_GetSegmentLocalRotationMatrix
//   GetSegmentLocalRotationMatrix( string SubjectName,
//                                  string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                               new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationMatrix Output =
  MyClient.GetSegmentLocalRotationMatrix( "Alice", "Pelvis" );
```

# GetSegmentLocalRotationQuaternion

*Return the rotation of a subject segment in local quaternion co-ordinates relative to its parent segment.*

*The quaternion is of the form (x, y, z, w) where w is the real component and x, y & z are the imaginary components.* ***N.B. This is different from that used in many other applications, which use (w, x, y, z).***

*See Also : GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix , GetSegmentLocalRotationEulerXYZ, GetSegmentGlobalTranslation,GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix, GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[4] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Rotation will be [0,0,0,0] |

C++

```
// class Output_GetSegmentLocalRotationQuaternion
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 4 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentLocalRotationQuaternion
//    GetSegmentLocalRotationQuaternion(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationQuaternion Output =
   MyClient.GetSegmentLocalRotationQuaternion( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentLocalRotationQuaternion( SubjectName, SegmentName
)

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentLocalRotationQuaternion( "Alice", "Pelvis"
);
```

.NET

```
// public class Output_GetSegmentLocalRotationQuaternion
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
//
// Output_GetSegmentLocalRotationQuaternion
//   GetSegmentLocalRotationQuaternion( string SubjectName,
//                                      string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationQuaternion Output =
   MyClient.GetSegmentLocalRotationQuaternion( "Alice", "Pelvis" );
```

# GetSegmentLocalRotationEulerXYZ

*Return the rotation of a subject segment in local EulerXYZ co-ordinates relative to its parent segment.*

*See Also : GetSegmentLocalTranslation, GetSegmentLocalRotationHelical, GetSegmentLocalRotationMatrix, GetSegmentLocalRotationQuaternion, GetSegmentGlobalTranslation,GetSegmentGlobalRotationHelical, GetSegmentGlobalRotationMatrix , GetSegmentGlobalRotationQuaternion, GetSegmentGlobalRotationEulerXYZ*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Segment Name | string | The name of the segment. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidSegmentName` |
| | Rotation | double[3] | The rotation of the segment |
| | Occluded | boolean | True if the segment was absent at this frame. In this case the Rotation will be [0,0,0] |

C++

```
// class Output_GetSegmentLocalRotationEulerXYZ
// {
// public:
//   Result::Enum Result;
//   double       Rotation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetSegmentLocalRotationEulerXYZ
//   GetSegmentLocalRotationEulerXYZ(
//        const String & SubjectName,
//        const String & SegmentName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationEulerXYZ Output =
  MyClient.GetSegmentLocalRotationEulerXYZ( "Alice", "Pelvis" );
```

MATLAB

```
% [Output] = GetSegmentLocalRotationEulerXYZ( SubjectName, SegmentName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetSegmentLocalRotationEulerXYZ( "Alice", "Pelvis" );
```

.NET

```
// public class Output_GetSegmentLocalRotationEulerXYZ
// {
//   public Result   Result;
//   public double[] Rotation;
//   public bool     Occluded;
// };
```

```
//
// Output_GetSegmentLocalRotationEulerXYZ
//   GetSegmentLocalRotationEulerXYZ( string SubjectName,
//                                    string SegmentName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetSegmentLocalRotationEulerXYZ Output =
  MyClient.GetSegmentLocalRotationEulerXYZ( "Alice", "Pelvis" );
```

# GetMarkerCount

*Return the number of markers for a specified subject in the DataStream. This information can be used in conjunction with GetMarkerName*

*See Also : GetSubjectName, GetMarkerName*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName` |
| | Marker Count | unsigned integer | The number of markers |

C++

```
// class Output_GetMarkerCount
// {
// public:
//   Result::Enum Result;
//   unsigned int MarkerCount;
// };
//
// Output_GetMarkerCount GetMarkerCount(
//     const String & SubjectName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableMarkerData();
MyClient.Connect( "localhost" );

Output_GetMarkerCount Output;
Output = MyClient.GetMarkerCount( "Bob" ); // Output.Result == NoFrame
                                           // Output.MarkerCount == 0
MyClient.GetFrame();

Output = MyClient.GetMarkerCount( "Alice" );
                                    // Output.Result ==
InvalidSubjectName
                                    // Output.MarkerCount == 0
                                    // (no "Alice")

Output = MyClient.GetMarkerCount( "Bob" ); // Output.Result == Success
                                           // Output.MarkerCount >= 0
```

MATLAB

```
% [Output] = GetMarkerCount( SubjectName )

MyClient = Client();
MyClient.EnableMarkerData();
MyClient.Connect( "localhost" );

Output = MyClient.GetMarkerCount( "Bob" ); % Output.Result == NoFrame
                                           % Output.MarkerCount == 0
MyClient.GetFrame();

Output = MyClient.GetMarkerCount( "Alice" );
                                    % Output.Result ==
InvalidSubjectName
                                    % Output.MarkerCount == 0
                                    % (no "Alice")

Output = MyClient.GetMarkerCount( "Bob" ); % Output.Result == Success
                                           % Output.MarkerCount >= 0
```

.NET

```
// public class Output_GetMarkerCount
// {
//   public Result Result;
//   public uint   MarkerCount;
// };
//
// Output_GetMarkerCount GetMarkerCount( string SubjectName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                  new ViconDataStreamSDK.DotNET.Client();
MyClient.EnableMarkerData();
MyClient.Connect( "localhost" );

Output_GetMarkerCount Output;
Output = MyClient.GetMarkerCount( "Bob" ); // Output.Result == NoFrame
                                           // Output.MarkerCount == 0
MyClient.GetFrame();

Output = MyClient.GetMarkerCount( "Alice" );
                                    // Output.Result ==
InvalidSubjectName
                                    // Output.MarkerCount == 0
                                    // (no "Alice")

Output = MyClient.GetMarkerCount( "Bob" ); // Output.Result == Success
                                           // Output.MarkerCount >= 0
```

# GetMarkerName

*Return the name of a marker for a specified subject. This can be passed into GetMarkerGlobalTranslation.*

*See Also : GetMarkerCount, GetMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Marker Index | unsigned integer | The index of the marker. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidIndex` |
| | Marker Name | string | The name of the marker |

C++

A valid Marker Index is between 0 and GetMarkerCount()-1

```
// class Output_GetMarkerName
// {
// public:
//   Result::Enum Result;
//   String       MarkerName;
// };
//
// Output_GetMarkerName GetMarkerName(
//     const String      & SubjectName,
//     const unsigned int  MarkerIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output_GetMarkerCount OutputGMC;
OutputGMC = MyClient.GetMarkerCount( "Bob" );
                                        // OutputGMC.Result == Success
                                        // OutputGMC.MarkerCount == 2

Output_GetMarkerName OutputGMN;
OutputGMN = MyClient.GetMarkerName( "Alice", 0 );
                                    // OutputGMN.Result ==
InvalidSubjectName
                                    // OutputGMN.MarkerName == ""
                                    // (no "Alice")

OutputGMN = MyClient.GetMarkerName( "Bob", 0 );
                                      // OutputGMN.Result == Success
                                      // OutputGMN.MarkerName == "LASI"
OutputGMN = MyClient.GetMarkerName( "Bob", 1 );
                                      // OutputGMN.Result == Success
                                      // OutputGMN.MarkerName == "RASI"
OutputGMN = MyClient.GetMarkerName( "Bob", 2 );
                                      // OutputGMN.Result == InvalidIndex
                                      // OutputGMN.MarkerName == ""
                                      // (no third marker)
```

MATLAB

A valid Marker Index is between 1 and GetMarkerCount()

```
% [Output] = GetMarkerName( SubjectName, MarkerIndex )
```

```
MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

OutputGMC = MyClient.GetMarkerCount( "Bob" );
                                        // OutputGMC.Result == Success
                                        // OutputGMC.MarkerCount == 2

OutputGMN = MyClient.GetMarkerName( "Alice", 1 );
                                    // OutputGMN.Result ==
InvalidSubjectName
                                    // OutputGMN.MarkerName == ""
                                    // (no "Alice")

OutputGMN = MyClient.GetMarkerName( "Bob", 1 );
                                      // OutputGMN.Result == Success
                                      // OutputGMN.MarkerName == "LASI"
OutputGMN = MyClient.GetMarkerName( "Bob", 2 );
                                      // OutputGMN.Result == Success
                                      // OutputGMN.MarkerName == "RASI"
OutputGMN = MyClient.GetMarkerName( "Bob", 3 );
                                      // OutputGMN.Result == InvalidIndex
                                      // OutputGMN.MarkerName == ""
                                      // (no third marker)
```

.NET

A valid Marker Index is between 0 and GetMarkerCount()-1

```
// public class Output_GetMarkerName
// {
//   public Result Result;
//   public string MarkerName;
// };
//
// Output_GetMarkerName GetMarkerName( string SubjectName,
//                                     uint   MarkerIndex );

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output_GetMarkerCount OutputGMC;
OutputGMC = MyClient.GetMarkerCount( "Bob" );
                                        // OutputGMC.Result == Success
                                        // OutputGMC.MarkerCount == 2

Output_GetMarkerName OutputGMN;
OutputGMN = MyClient.GetMarkerName( "Alice", 0 );
                                    // OutputGMN.Result ==
InvalidSubjectName
                                    // OutputGMN.MarkerName == ""
                                    // (no "Alice")

OutputGMN = MyClient.GetMarkerName( "Bob", 0 );
                                      // OutputGMN.Result == Success
                                      // OutputGMN.MarkerName == "LASI"
OutputGMN = MyClient.GetMarkerName( "Bob", 1 );
                                      // OutputGMN.Result == Success
                                      // OutputGMN.MarkerName == "RASI"
OutputGMN = MyClient.GetMarkerName( "Bob", 2 );
                                      // OutputGMN.Result == InvalidIndex
                                      // OutputGMN.MarkerName == ""
                                      // (no third marker)
```

# GetMarkerParentName

*Return the name of the segment which is the parent of this marker.*
*See Also : GetMarkerCount, GetMarkerName, GetMarkerGlobalTranslation*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Marker Name | string | The name of the marker. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidMarkerName` |
| | Segment Name | string | The name of the parent segment. |

C++

```
// class Output_GetMarkerParentName
// {
// public:
//   Result::Enum Result;
//   String       SegmentName;
// };
//
// Output_GetMarkerParentName GetMarkerParentName(
//     const String      & SubjectName,
//     const String      & MarkerName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output_GetMarkerParentName Output;
Output = MyClient.GetMarkerParentName( "Bob", "LFHD" );
                                       // Output.Result == Success
                                       // Output.SegmentName == "Head"
```

MATLAB

```
% [Output] = GetMarkerParentName( SubjectName, MarkerName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output = MyClient.GetMarkerParentName( "Bob", "LFHD" );
                                       // Output.Result == Success
                                       // Output.SegmentName == "Head"
```

.NET

```
// public class Output_GetMarkerParentName
// {
//   public Result Result;
//   public string SegmentName;
// };
//
// Output_GetMarkerParentName GetMarkerParentName( string SubjectName,
//                                                 string MarkerName );

ViconDataStreamSDK.DotNET.Client MyClient =
                                 new ViconDataStreamSDK.DotNET.Client();
```

| | |
|---|---|
| | ```<br>MyClient.Connect( "localhost" );<br>MyClient.EnableMarkerData();<br>MyClient.GetFrame();<br><br>Output_GetMarkerParentName Output;<br>Output = MyClient.GetMarkerParentName( "Bob", "LFHD" );<br>                                        // Output.Result == Success<br>                                        // Output.SegmentName == "Head"<br>``` |

# GetMarkerGlobalTranslation

*Return the translation of a subject marker in global co-ordinates.*

*The Translation is of the form ( x, y, z ) where x, y & z are in Millimeters with respect to the global origin.*

*See Also : GetMarkerName*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Marker Name | string | The name of the marker. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidMarkerName` |
| | Translation | double[3] | The translation of the marker |
| | Occluded | boolean | True if the marker was absent at this frame. In this case the Translation will be [0,0,0] |

C++

```
// class Output_GetMarkerGlobalTranslation
// {
// public:
//   Result::Enum Result;
//   double       Translation[ 3 ];
//   bool         Occluded;
// };
//
// Output_GetMarkerGlobalTranslation GetMarkerGlobalTranslation(
//        const String & SubjectName,
//        const String & MarkerName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output_GetMarkerGlobalTranslation Output =
  MyClient.GetMarkerGlobalTranslation( "Alice", "LASI" );
```

MATLAB

```
% [Output] = GetMarkerGlobalTranslation( SubjectName, MarkerName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output = MyClient.GetMarkerGlobalTranslation( "Alice", "LASI" );
```

.NET

```
// public class Output_GetMarkerGlobalTranslation
// {
//   public Result   Result;
//   public double[] Translation[];
//   public bool     Occluded;
// };
//
// Output_GetMarkerGlobalTranslation
```

```
//   GetMarkerGlobalTranslation( string SubjectName,
//                               string MarkerName );

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerData();
MyClient.GetFrame();

Output_GetMarkerGlobalTranslation Output =
  MyClient.GetMarkerGlobalTranslation( "Alice", "LASI" );
```

# GetMarkerRayContributionCount

*Return the number of rays that are contributing to a labelled marker in the data stream. This information can be used in conjunction with GetMarkerRayContribution*

*See Also : GetMarkerRayContribution, EnableMarkerRayData, DisableMarkerRayData, IsMarkerRayDataEnabled*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Marker Name | string | The name of the marker. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidMarkerName` |
| | RayContributionCount | unsigned integer | The number of rays |

C++

```
// class Output_GetMarkerRayContributionCount
// {
// public:
//   Result::Enum Result;
//   unsigned int RayContributionCount;
// };
//
// Output_GetMarkerRayContributionCount GetMarker RayContributionCount (
//        const String & SubjectName,
//        const String & MarkerName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableMarkerRayData();
MyClient.GetFrame();

Output_GetMarkerRayContributionCount Output =
  MyClient.GetMarkerRayContributionCount ( "Alice", "LASI" );
```

MATLAB

```
% [Output] = GetMarkerRayContributionCount ( SubjectName, MarkerName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerRayData();
MyClient.GetFrame();

Output = MyClient.GetMarkerRayContributionCount ( "Alice", "LASI" );
```

.NET

```
// public class Output_GetMarkerRayContributionCount
// {
//   public Result   Result;
//   unsigned int    RayContributionCount;
// };
//
// Output_GetMarkerRayContributionCount
//   GetMarkerRayContributionCount( string SubjectName,
//                                  string MarkerName );

ViconDataStreamSDK.DotNET.Client MyClient =
                            new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerRayData();
MyClient.GetFrame();
```

| | |
|---|---|
| | `Output_GetMarkerRayContributionCount Output = MyClient.GetMarkerRayContributionCount( "Alice", "LASI" );` |

# GetMarkerRayContribution

*Return the camera id for an indexed ray that is contributing to a labelled marker in the data stream. This information can be used in conjunction with GetMarkerRayContributionCount*

*See Also : GetMarkerRayContributionCount, EnableMarkerRayData, DisableMarkerRayData, IsMarkerRayDataEnabled*

| | | | |
|---|---|---|---|
| Input | Subject Name | string | The name of the subject |
| | Marker Name | string | The name of the marker. |
| | MarkerRayContributionIndex | unsigned int | The index of the ray required |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidSubjectName`<br>`Result.InvalidMarkerName`<br>`Result.InvalidIndex` |
| | CameraID | unsigned integer | The Camera ID of the camera producing the ray |
| | CentroidIndex | unsigned integer | The index of the centroid resulting from the ray |

**C++**

A valid Ray Index is between 0 and GetMarkerRayContributionCount()-1

```
// class Output_GetMarkerRayContribution
// {
// public:
//   Result::Enum Result;
//   unsigned int CameraID;
//   unsigned int CentroidIndex;
// };
//
// Output_GetMarkerRayContribution GetMarkerRayContribution (
//        const String & SubjectName,
//        const String & MarkerName,
//        unsigned int MarkerRayContributionIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableMarkerRayData();
MyClient.GetFrame();

Output_GetMarkerRayContribution Output =
  MyClient.GetMarkerRayContribution( "Alice", "LASI", 0 );
```

**MATLAB**

A valid Ray Index is between 1 and GetMarkerRayContributionCount()

```
% [Output] = GetMarkerRayContributionCount ( SubjectName, MarkerName,
MarkerRayContributionIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableMarkerRayData();
MyClient.GetFrame();

Output = MyClient.GetMarkerRayContribution ( "Alice", "LASI", 1 );
```

| | |
|---|---|
| .NET | A valid Ray Index is between 0 and GetMarkerRayContributionCount()-1<br>`// public class Output_GetMarkerRayContribution`<br>`// {`<br>`//   public Result   Result;`<br>`//   unsigned int    CameraID;`<br>`//   unsigned int    CentroidIndex`<br>`// };`<br>`//`<br><br>`// Output_GetMarkerRayContribution`<br>`//   GetMarkerRayContributionCount(`<br>`//                              string SubjectName,`<br>`//                              string MarkerName,`<br>`//                              unsigned int MarkerRayContributionIndex );`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient =`<br>`                            new ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`MyClient.EnableMarkerRayData();`<br>`MyClient.GetFrame();`<br><br>`Output_GetMarkerRayContribution Output =`<br>`  MyClient.GetMarkerRayContribution( "Alice", "LASI", 0 );` |
| | |

# GetUnlabeledMarkerCount

*Return the number of unlabeled markers in the data stream. This information can be used in conjunction with GetGlobalUnlabeledMarkerTranslation*

*See Also : GetGlobalUnlabeledMarkerTranslation*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | MarkerCount | unsigned integer | The number of markers |

C++

```
// class Output_GetUnlabeledMarkerCount
// {
// public:
//   Result::Enum Result;
//   unsigned int MarkerCount;
// };
// Output_GetUnlabeledMarkerCount GetUnlabeledMarkerCount() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableUnlabeledMarkerData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetUnlabeledMarkerCount Output =
  MyClient.GetUnlabeledMarkerCount(); // Output.Result == Success
                                      // Output.MarkerCount >= 0
```

MATLAB

```
% [Output] = GetUnlabeledMarkerCount();

MyClient = Client();
MyClient.EnableUnlabeledMarkerData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetUnlabeledMarkerCount(); // Output.Result ==
Success
                                             // Output.MarkerCount >= 0
```

.NET

```
// public class Output_GetUnlabeledMarkerCount
// {
//   public Result Result;
//   public uint   MarkerCount;
// };
// Output_GetUnlabeledMarkerCount GetUnlabeledMarkerCount();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.EnableUnlabeledMarkerData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetUnlabeledMarkerCount Output =
  MyClient.GetUnlabeledMarkerCount(); // Output.Result == Success
                                      // Output.MarkerCount >= 0
```

# GetUnlabeledMarkerGlobalTranslation

*Return the translation of an unlabeled marker in global co-ordinates.*

*The Translation is of the form ( x, y, z ) where x, y & z are in Millimeters with respect to the global origin.*

*See Also : GetUnlabelledMarkerCount*

| | | | |
|---|---|---|---|
| Input | Marker Index | unsigned integer | The index of the marker. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | Translation | double[3] | The translation of the marker |

**C++**

A valid Marker Index is between 0 and GetUnlabeledMarkerCount()-1

```
// class Output_GetUnlabeledMarkerGlobalTranslation
// {
// public:
//   Result::Enum Result;
//   double       Translation[ 3 ];
// };
//
// Output_GetUnlabeledMarkerGlobalTranslation
//   GetUnlabeledMarkerGlobalTranslation(
//       const unsigned int MarkerIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableUnlabeledMarkerData();
MyClient.GetFrame();

Output_GetUnlabeledMarkerGlobalTranslation Output =
  MyClient.GetUnlabeledMarkerGlobalTranslation( 0 );
```

**MATLAB**

A valid Marker Index is between 1 and GetUnlabeledMarkerCount()

```
% [Output] = GetUnlabeledMarkerGlobalTranslation( MarkerIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableUnlabeledMarkerData();
MyClient.GetFrame();

Output = MyClient.GetUnlabeledMarkerGlobalTranslation( 1 );
```

**.NET**

A valid Marker Index is between 0 and GetUnlabeledMarkerCount()-1

```
// public class Output_GetUnlabeledMarkerGlobalTranslation
// {
//   public Result   Result;
//   public double[] Translation;
// };
//
// Output_GetUnlabeledMarkerGlobalTranslation
//   GetUnlabeledMarkerGlobalTranslation( uint MarkerIndex ) const;

ViconDataStreamSDK.DotNET.Client MyClient =
                              new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableUnlabeledMarkerData();
```

| | |
|---|---|
| | `MyClient.GetFrame();`<br><br>`Output_GetUnlabeledMarkerGlobalTranslation Output =`<br>`  MyClient.GetUnlabeledMarkerGlobalTranslation( 0 );` |

# GetDeviceCount

| | | | |
|---|---|---|---|
| *Return the number of ForcePlates, EMGs, and other devices in the DataStream. This information can be used in conjunction with GetDeviceName*<br>*See Also : GetDeviceName* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Device Count | unsigned integer | The number of devices |

C++

```
// class Output_GetDeviceCount
// {
// public:
//   Result::Enum Result;
//   unsigned int DeviceCount;
// };
// Output_GetDeviceCount GetDeviceCount() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetDeviceCount Output = MyClient.GetDeviceCount();
                                                  // Output.Result == Success
                                                  // Output.DeviceCount >= 0
```

MATLAB

```
% [Output] = GetDeviceCount()

MyClient = Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetDeviceCount();  // Output.Result == Success
                                     // Output.DeviceCount >= 0
```

.NET

```
// public class Output_GetDeviceCount
// {
//   public Result Result;
//   public uint   DeviceCount;
// };
// Output_GetDeviceCount GetDeviceCount();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetDeviceCount Output = MyClient.GetDeviceCount();
                                        // Output.Result == Success
                                        // Output.DeviceCount >= 0
```

# GetDeviceName

*Return the name and type of a device. This name can be passed into device functions.*
*See Also : GetDeviceCount, GetDeviceOutputCount, GetDeviceOutputValue*

| | | | |
|---|---|---|---|
| Input | Device Index | unsigned integer | The index of the device. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | Device Name | string | The name of the device |
| | Device Type | DeviceType | Unknown<br>ForcePlate |

**C++**

A valid Device Index is between 0 and GetDeviceCount()-1

```
// class Output_GetDeviceName
// {
// public:
//   Result::Enum     Result;
//   String           DeviceName;
//   DeviceType::Enum DeviceType;
// };
//
// Output_GetDeviceName
//   GetDeviceName( const unsigned int DeviceIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output_GetDeviceCount OutputGDC;
OutputGDC = MyClient.GetDeviceCount( DeviceCount );
                                    // OutputGDC.Result == Success
                                    // OutputGDC.DeviceCount == 2

Output_GetDeviceName OutputGDN;
OutputGDN = MyClient.GetDeviceName( 0 );
                                    // OutputGDN.Result == Success
                                    // OutputGDN.DeviceName ==
"ZeroWire"
                                    // OutputGDN.DeviceType == Unknown

OutputGDN = MyClient.GetDeviceName( 1 );
                                    // OutputGDN.Result == Success
                                    // OutputGDN.DeviceName == "AMTI #1"
                                    // OutputGDN.DeviceType ==
ForcePlate

OutputGDN = MyClient.GetDeviceName( 2 );
                                    // OutputGDN.Result == InvalidIndex
                                    // OutputGDN.DeviceName == ""
                                    // OutputGDN.DeviceType == Unknown
```

**MATLAB**

A valid Device Index is between 1 and GetDeviceCount()

```
% [Output] = GetDeviceName( DeviceIndex )
```

| | |
|---|---|
| | MyClient = Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableDeviceData();<br>MyClient.GetFrame();<br><br>OutputGDC = MyClient.GetDeviceCount( DeviceCount );<br>% OutputGDC.Result == Success<br>% OutputGDC.DeviceCount == 2<br><br>OutputGDN = MyClient.GetDeviceName( 1 );<br>% OutputGDN.Result == Success<br>% OutputGDN.DeviceName == "ZeroWire"<br>% OutputGDN.DeviceType == Unknown<br><br>OutputGDN = MyClient.GetDeviceName( 2 );<br>% OutputGDN.Result == Success<br>% OutputGDN.DeviceName == "AMTI #1"<br>% OutputGDN.DeviceType == ForcePlate<br><br>OutputGDN = MyClient.GetDeviceName( 3 );<br>% OutputGDN.Result == InvalidIndex<br>% OutputGDN.DeviceName == ""<br>% OutputGDN.DeviceType == Unknown |
| .NET | A valid Device Index is between 0 and GetDeviceCount()-1<br>// public class Output_GetDeviceName<br>// {<br>// public Result Result;<br>// public string DeviceName;<br>// public DeviceType DeviceType;<br>// };<br>//<br>// Output_GetDeviceName<br>// GetDeviceName( uint DeviceIndex );<br><br>ViconDataStreamSDK.DotNET.Client MyClient =<br>new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableDeviceData();<br>MyClient.GetFrame();<br><br>Output_GetDeviceCount OutputGDC;<br>OutputGDC = MyClient.GetDeviceCount( DeviceCount );<br>// OutputGDC.Result == Success<br>// OutputGDC.DeviceCount == 2<br><br>Output_GetDeviceName OutputGDN;<br>OutputGDN = MyClient.GetDeviceName( 0 );<br>// OutputGDN.Result == Success<br>// OutputGDN.DeviceName == "ZeroWire"<br>// OutputGDN.DeviceType == Unknown<br><br>OutputGDN = MyClient.GetDeviceName( 1 );<br>// OutputGDN.Result == Success<br>// OutputGDN.DeviceName == "AMTI #1"<br>// OutputGDN.DeviceType == ForcePlate<br><br>OutputGDN = MyClient.GetDeviceName( 2 );<br>// OutputGDN.Result == InvalidIndex<br>// OutputGDN.DeviceName == ""<br>// OutputGDN.DeviceType == Unknown |

# GetDeviceOutputCount

| | | | |
|---|---|---|---|
| *Return the number of outputs for a device in the data stream. This information can be used in conjunction with GetDeviceOutputName*<br>*See Also : GetDeviceName, GetDeviceOutputName* | | | |
| Input | Device Name | string | The device name |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidDeviceName` |
| | Device Output Count | unsigned integer | The number of device outputs |

C++

```
// class Output_GetDeviceOutputCount
// {
// public:
//   Result::Enum Result;
//   unsigned int DeviceOutputCount;
// };
//
// Output_GetDeviceOutputCount GetDeviceOutputCount(
//              const String & DeviceName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output_GetDeviceOutputCount Output;
Output = MyClient.GetDeviceOutputCount( "DataGlove" );
                                    // Output.Result ==
InvalidDeviceName
                                    // Output.DeviceOutputCount == 0
                                    // (no "DataGlove" device)

Output = MyClient.GetDeviceOutputCount( "ZeroWire" );
                                    // Output.Result == Success
                                    // Output.DeviceOutputCount == 6
```

MATLAB

```
% [Output] = GetDeviceOutputCount( DeviceName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output = MyClient.GetDeviceOutputCount( "DataGlove" );
                                    // Output.Result ==
InvalidDeviceName
                                    // Output.DeviceOutputCount == 0
                                    // (no "DataGlove" device)

Output = MyClient.GetDeviceOutputCount( "ZeroWire" );
                                    // Output.Result == Success
                                    // Output.DeviceOutputCount == 6
```

.NET

```
// public class Output_GetDeviceOutputCount
// {
//   public Result Result;
//   public uint   DeviceOutputCount;
```

```
// };
//
// Output_GetDeviceOutputCount GetDeviceOutputCount( string DeviceName
);

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output_GetDeviceOutputCount Output;
Output = MyClient.GetDeviceOutputCount( "DataGlove" );
                                      // Output.Result ==
InvalidDeviceName
                                      // Output.DeviceOutputCount == 0
                                      // (no "DataGlove" device)

Output = MyClient.GetDeviceOutputCount( "ZeroWire" );
                                      // Output.Result == Success
                                      // Output.DeviceOutputCount == 6
```

# GetDeviceOutputName

| | | | |
|---|---|---|---|
| *Return the name and SI unit of a device output. This name can be passed into GetDeviceOutputValue.*<br>*See Also : GetDeviceCount, GetDeviceOutputCount, GetDeviceOutputValue* | | | |
| Input | Device Name | string | The device name |
| | Device Output Index | integer | The index of the device output. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidDeviceName`<br>`Result.InvalidIndex` |
| | Device Output Name | string | The name of the device output, e.g.<br>"Fx" - Force X<br>"Fy" - Force Y<br>"Fz" - Force Z<br>"Mx" - Moment X<br>"My" - Moment Y<br>"Mz" - Moment Z<br>"Cx" - Centre Of Pressure X<br>"Cy" - Centre Of Pressure Y<br>"Cz" - Centre Of Pressure Z<br>"Pin1" - Analog Input 1<br>"Pin2" - Analog Input 2 |
| | Device Output Unit | Unit | The unit of the device output.<br>Unit.Unknown<br>Unit.Volt<br>Unit.Newton<br>Unit.NewtonMeter<br>Unit.Meter<br>Unit.Kilogram<br>Unit.Second<br>Unit.Ampere<br>Unit.Kelvin<br>Unit.Mole<br>Unit.Candela<br>Unit.Radian<br>Unit.Steradian<br>Unit.MeterSquared<br>Unit.MeterCubed |

| | | | |
|---|---|---|---|
| | | | Unit.MeterPerSecond<br>Unit.MeterPerSecondSquared<br>Unit.RadianPerSecond<br>Unit.RadianPerSecondSquared<br>Unit.Hertz<br>Unit.Joule<br>Unit.Watt<br>Unit.Pascal<br>Unit.Lumen<br>Unit.Lux<br>Unit.Coulomb<br>Unit.Ohm<br>Unit.Farad<br>Unit.Weber<br>Unit.Tesla<br>Unit.Henry<br>Unit.Siemens<br>Unit.Becquerel<br>Unit.Gray<br>Unit.Sievert<br>Unit.Katal |
| C++ | A valid Device Output Index is between 0 and GetDeviceOutputCount()-1<br>`// class Output_GetDeviceOutputName`<br>`// {`<br>`// public:`<br>`//  Result::Enum Result;`<br>`//  String  DeviceOutputName;`<br>`//  Unit::Enum  DeviceOutputUnit;`<br>`// };`<br>`//`<br>`// Output_GetDeviceOutputName GetDeviceOutputName(`<br>`//  const String  & DeviceName,`<br>`//  const unsigned int  DeviceOutputIndex ) const;`<br><br>`ViconDataStreamSDK::CPP::Client MyClient;`<br>`MyClient.Connect( "localhost" );`<br>`MyClient.EnableDeviceData();`<br>`MyClient.GetFrame();`<br><br>`Output_GetDeviceOutputName Output =`<br>`  MyClient.GetDeviceOutputName( "AMTI", 0 );`<br>`  // Output.Result == Success`<br>`  // Output.DeviceOutputName == "Fx"`<br>`  // Output.DeviceOutputUnit == Newton` | | |
| MATLAB | A valid Device Output Index is between 1 and GetDeviceOutputCount()<br>`% [Output] = GetDeviceOutputName( DeviceName, DeviceOutputIndex )`<br><br>`MyClient = Client();` | | |

| | |
|---|---|
| | MATLAB:<br>`MyClient.Connect( "localhost" );`<br>`MyClient.EnableDeviceData();`<br>`MyClient.GetFrame();`<br><br>`Output = MyClient.GetDeviceOutputName( "AMTI", 0 );`<br>`% Output.Result == Success`<br>`% Output.DeviceOutputName == "Fx"`<br>`% Output.DeviceOutputUnit == Newton` |
| .NET | A valid Device Output Index is between 0 and GetDeviceOutputCount()-1<br>`// public class Output_GetDeviceOutputName`<br>`// {`<br>`//  public Result Result;`<br>`//  public string DeviceOutputName;`<br>`//  public Unit  DeviceOutputUnit;`<br>`// };`<br>`//`<br>`// Output_GetDeviceOutputName GetDeviceOutputName(`<br>`//  string DeviceName,`<br>`//  uint  DeviceOutputIndex );`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient =`<br>`new ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`MyClient.EnableDeviceData();`<br>`MyClient.GetFrame();`<br><br>`Output_GetDeviceOutputName Output =`<br>`MyClient.GetDeviceOutputName( "AMTI", 0 );`<br>`// Output.Result == Success`<br>`// Output.DeviceOutputName == "Fx"`<br>`// Output.DeviceOutputUnit == Newton` |

# GetDeviceOutputValue

*Return the value of a device output. If there are multiple samples for a frame, then the first sample is returned.*

*The force plate data provided in the individual device channels is in a coordinate system local to the plate aligned Z upwards, Y towards the front of the plate. This coordinate system is located at the center of the top surface of the plate. Any plate origin offset has been accounted for in the moment data. These are forces not reactions.*

*See Also : GetDeviceCount, GetDeviceOutputCount, GetDeviceOutputName*

| | | | |
|---|---|---|---|
| Input | Device Name | string | The device name |
| | Device Output Name | string | The name of the device output. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidDeviceName`<br>`Result.InvalidDeviceOutputName` |
| | Value | double | The value of the device output |
| | Occluded | boolean | True if the value was absent at this frame. In this case the Value will be 0. |

C++

```
// class Output_GetDeviceOutputValue
// {
// public:
//   Result::Enum Result;
//   double       Value;
//   bool         Occluded;
// };
//
// Output_GetDeviceOutputValue
//    GetDeviceOutputValue(
//       const String & DeviceName,
//       const String & DeviceOutputName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output_GetDeviceOutputValue Output =
  MyClient.GetDeviceOutputValue( "AMTI", "Fx" );
                                      // Output.Result == Success
                                      // Output.Value == ?
                                      // Output.Occluded = ?
```

MATLAB

```
// [Output] =  GetDeviceOutputValue( DeviceName, DeviceOutputName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output = MyClient.GetDeviceOutputValue( "AMTI", "Fx" );
                                      // Output.Result == Success
```

| | |
|---|---|
| | // Output.Value == ?<br>// Output.Occluded = ? |
| .NET | // public class Output_GetDeviceOutputValue<br>// {<br>//  public Result Result;<br>//  public double Value;<br>//  public bool  Occluded;<br>// };<br>//<br>// Output_GetDeviceOutputValue<br>//  GetDeviceOutputValue( string DeviceName,<br>//  string DeviceOutputName );<br><br>ViconDataStreamSDK.DotNET.Client MyClient =<br>  new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableDeviceData();<br>MyClient.GetFrame();<br><br>Output_GetDeviceOutputValue Output =<br>  MyClient.GetDeviceOutputValue( "AMTI", "Fx" );<br>// Output.Result == Success<br>// Output.Value == ?<br>// Output.Occluded = ? |

# GetDeviceOutputSubsamples

*Return the number of samples available the specified device for the current frame. If an analogue device is sampling at 1000 Hz and the system is running at 100 Hz then this function will return 10.*

*The samples can accessed by supplying the subsample index to GetDeviceOutputValue. See below.*

*See Also :GetDeviceOutputCount, GetDeviceOutputValue*

| | | | |
|---|---|---|---|
| Input | Device Name | string | The device name |
| | Device Output Name | string | The name of the device output. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex`<br>`Result.InvalidDeviceName`<br>`Result.InvalidDeviceOutputName` |
| | DeviceOutputSubsamples | Uint | The number of subsamples for this device output. |
| | Occluded | boolean | True if the value was absent at this frame. In this case the Value will be 0. |

C++

```
//  class Output_GetDeviceOutputSubsamples
//  {
//  public:
//    Result::Enum Result;
//    unsigned int DeviceOutputSubsamples;
//    bool         Occluded;
//  };
//
//    Output_GetDeviceOutputSubsamples GetDeviceOutputSubsamples( const
String & DeviceName,
//                                                                const
String & DeviceOutputName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output_GetDeviceOutputSubsamples Output =
  MyClient. GetDeviceOutputSubsamples ( "AMTI", "Fx" );
```

MATLAB

```
// [Output] =  GetDeviceOutputSubsamples( DeviceName, DeviceOutputName
)

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output = MyClient. GetDeviceOutputSubsamples ( "AMTI", "Fx" );
                                        // Output.Result == Success
                                        // Output. DeviceOutputSubsamples
== ?
```

| | |
|---|---|
| | // Output.Occluded = ? |
| .NET | // public class Output_GetDeviceOutputSubsamples<br>// {<br>//  public Result Result;<br>//  unsigned int DeviceOutputSubsamples;<br>//  public bool  Occluded;<br>// };<br>//<br>// Output_GetDeviceOutputSubsamples^ GetDeviceOutputSubsamples( String^ DeviceName,<br>//<br>String^  DeviceOutputName )<br><br>ViconDataStreamSDK.DotNET.Client MyClient =<br>new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableDeviceData();<br>MyClient.GetFrame();<br><br>Output_GetDeviceOutputSubsamples Output =<br>MyClient.GetDeviceOutputSubsamples( "AMTI", "Fx" );<br>// Output.Result == Success<br>// Output.DeviceOutputSubsamples == ?<br>// Output.Occluded = ? |

# GetDeviceOutputValue2

*Return the value of a device output. This override allows access to the individual subsamples for the current frame of data. See GetDeviceOutputValue for information about the meaning of the force plate channels.*

*See Also : GetDeviceOutputSubsamples, GetDeviceOutputValue*

| | | | |
|---|---|---|---|
| Input | Device Name | string | The device name |
| | Device Output Name | string | The name of the device output. |
| | Subsample | unsigned int | The subsample to access. |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex`<br>`Result.InvalidDeviceName`<br>`Result.InvalidDeviceOutputName` |
| | Value | double | The value of the device output |
| | Occluded | boolean | True if the value was absent at this frame. In this case the Value will be 0. |

C++

```
// class Output_GetDeviceOutputValue
// {
// public:
//   Result::Enum Result;
//   double       Value;
//   bool         Occluded;
// };
//
// Output_GetDeviceOutputValue
//    GetDeviceOutputValue(
//       const String & DeviceName,
//       const String & DeviceOutputName ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output_GetDeviceOutputValue Output =
  MyClient.GetDeviceOutputValue( "AMTI", "Fx", 6 );
                                      // Output.Result == Success
                                      // Output.Value == ?
                                      // Output.Occluded = ?
```

MATLAB

```
// [Output] =  GetDeviceOutputValue( DeviceName, DeviceOutputName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient.EnableDeviceData();
MyClient.GetFrame();

Output = MyClient.GetDeviceOutputValue( "AMTI", "Fx", 6 );
                                      // Output.Result == Success
```

| | |
|---|---|
| | // Output.Value == ?<br>// Output.Occluded = ? |
| .NET | // public class Output_GetDeviceOutputValue<br>// {<br>//  public Result Result;<br>//  public double Value;<br>//  public bool  Occluded;<br>// };<br>//<br>// Output_GetDeviceOutputValue<br>//  GetDeviceOutputValue( string DeviceName,<br>//  string DeviceOutputName );<br><br>ViconDataStreamSDK.DotNET.Client MyClient =<br>  new ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableDeviceData();<br>MyClient.GetFrame();<br><br>Output_GetDeviceOutputValue Output =<br>  MyClient.GetDeviceOutputValue( "AMTI", "Fx", 6 );<br>// Output.Result == Success<br>// Output.Value == ?<br>// Output.Occluded = ? |

# GetForcePlateCount

*Return the number of ForcePlates available in the DataStream.*
*See Also : GetGlobalForceVector, GetGlobalMomentVector, GetGlobalCentreOfPressure*

| | | | |
|---|---|---|---|
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Force Plate Count | unsigned integer | The number of force plates |

C++

```
// class Output_GetForcePlateCount
// {
// public:
//   Result::Enum Result;
//   unsigned int ForcePlateCount;
// };
//
// Output_GetForcePlateCount GetForcePlateCount() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetForcePlateCount Output = MyClient. GetForcePlateCount ();
                                                // Output.Result == Success
                                                // Output. ForcePlateCount
>= 0
```

MATLAB

```
% [Output] = GetForcePlateCount()

MyClient = Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetForcePlateCount();  // Output.Result == Success
                                         // Output.ForcePlateCount >= 0
```

.NET

```
// public class Output_GetForcePlateCount
// {
//   public Result Result;
//   public uint   ForcePlateCount;
// };
//
// Output_GetForcePlateCount GetForcePlateCount();

ViconDataStreamSDK.DotNET.Client MyClient =
                                new ViconDataStreamSDK.DotNET.Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetForcePlateCount Output = MyClient.GetForcePlateCount();
                                                // Output.Result == Success
                                                // Output.ForcePlateCount
>= 0
```

# GetGlobalForceVector

*Return the force vector for the plate in global co-ordinates.*

*The vector is in Newtons and is with respect to the global coordinate system regardless of the orientation of the plate. The vector represents the force exerted upon the plate, not the reaction force.*

*If multiple sub-samples are available this function returns the first subsample. See the alternate version of this function to access all of the analogue data.*

*See Also : GetGlobalMomentVector, GetGlobalCentreOfPressure*

| | | | |
|---|---|---|---|
| Input | Force Plate Index | unsigned integer | The index of the plate |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | ForceVector | double[3] | The force on the plate |

**C++**

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

```
// class Output_GetGlobalForceVector
// {
// public:
//   Result::Enum Result;
//   double       ForceVector[ 3 ];
// };
//
// Output_GetGlobalForceVector
//   GetGlobalForceVector (
//       const unsigned int ForcePlateIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetGlobalForceVector Output = MyClient.GetGlobalForceVector( 0
);
```

**MATLAB**

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

```
% [Output] = GetGlobalForceVector( ForcePlateIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output = MyClient. GetGlobalForceVector( 1 );
```

**.NET**

A valid ForcePlateIndex is between 0 and GetForcePlateCount() - 1

```
//  public ref class Output_GetGlobalForceVector
//  {
//  public:
//    Result           Result;
//    array< double >^ ForceVector;
//  };
//
// Output_GetGlobalForceVector
//   GetGlobalForceVector( uint ForcePlateIndex ) const;
```

| | |
|---|---|
| | ```<br>ViconDataStreamSDK.DotNET.Client MyClient = new<br>ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableUnlabeledMarkerData();<br>MyClient.GetFrame();<br><br>Output_ GetGlobalForceVector Output =  MyClient. GetGlobalForceVector(<br>0 );<br>``` |

# GetGlobalMomentVector

*Return the moment vector for the plate in global co-ordinates.*

*The vector is in Newton-Meters and is with respect to the global coordinate system regardless of the orientation of the plate.*

*The vector represents the moment exerted upon the plate, not the reaction moment. Any force plate origin offset is accounted for in the moments so they are acting about the exact centre of the top surface of the plate.*

*If multiple sub-samples are available this function returns the first subsample. See the alternate version of this function to access all of the analogue data.*

*See Also : GetGlobalForceVector, GetGlobalCentreOfPressure*

| | | | |
|---|---|---|---|
| Input | Plate Index | unsigned integer | The index of the force plate |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | MomentVector | double[3] | The moment exterted on the plate |

C++

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

```
// class Output_GetGlobalMomentVector
// {
// public:
//   Result::Enum Result;
//   double       MomentVector[ 3 ];
// };
//
// Output_GetGlobalMomentVector GetGlobalMomentVector (
//       const unsigned int ForcePlateIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetGlobalMomentVector Output = MyClient.GetGlobalMomentVector( 0
);
```

MATLAB

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

```
% [Output] = GetGlobalMomentVector( ForcePlateIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output = MyClient. GetGlobalMomentVector( 1 );
```

.NET

A valid ForcePlateIndex is between 0 and GetForcePlateCount() - 1

```
//  public ref class Output_GetGlobalMomentVector
//  {
//  public:
//    Result           Result;
//    array< double >^ MomentVector;
//  };
```

| | |
|---|---|
| | `// Output_GetGlobalMomentVector`<br>`//  GetGlobalMomentVector( uint ForcePlateIndex ) const;`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient = new`<br>`ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`MyClient. EnableDeviceData ();`<br>`MyClient.GetFrame();`<br><br>`Output_ GetGlobalMomentVector Output = MyClient.GetGlobalMomentVector(`<br>`0 );` |

# GetGlobalCentreOfPressure

*Return the centre of pressure for the plate in global co-ordinates.*

*The position is in millimeters and is with respect to the global coordinate system.*

*If multiple sub-samples are available this function returns the first subsample. See the alternate version of this function to access all of the analogue data.*

*See Also : GetGlobalForceVector, GetGlobalMomentVector*

| | | | |
|---|---|---|---|
| Input | Plate Index | unsigned integer | The index of the force plate |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | CentreOfPressure | double[3] | The CoP. |

C++

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

```
// class Output_GetGlobalCentreOfPressure
// {
// public:
//   Result::Enum Result;
//   double       CentreOfPressure[ 3 ];
// };
//
// Output_GetGlobalCentreOfPressure
//   GetGlobalCentreOfPressure (
//       const unsigned int ForcePlateIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetGlobalCentreOfPressure Output =
MyClient.GetGlobalCentreOfPressure( 0 );
```

MATLAB

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

```
% [Output] = GetGlobalCentreOfPressure( ForcePlateIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output = MyClient. GetGlobalCentreOfPressure( 1 );
```

.NET

A valid ForcePlateIndex is between 0 and GetForcePlateCount() - 1

```
// public class Output_ GetGlobalCentreOfPressure
// {
//  public:
//    Result           Result;
//    array< double >^ CentreOfPressure;
// };
//
// Output_GetGlobalCentreOfPressure
//   GetGlobalCentreOfPressure( uint ForcePlateIndex ) const;

ViconDataStreamSDK.DotNET.Client MyClient =  new
ViconDataStreamSDK.DotNET.Client();
```

| | |
|---|---|
| | MyClient.Connect( "localhost" );<br>MyClient. EnableDeviceData ();<br>MyClient.GetFrame();<br><br>Output_ GetGlobalCentreOfPressure Output =<br>MyClient.GetGlobalCentreOfPressure( 0 ); |

# GetForcePlateSubsamples

*Return the number of subsamples available for a specified plate in the current frame. Additional versions of GetGlobalForceVector, GetGlobalMomentVector GetGlobalCentreOfPressure take the subsample index to allow access of all the force plate data.*

*See Also : GetGlobalForceVector, GetGlobalMomentVector, GetGlobalCentreOfPressure*

| | | | |
|---|---|---|---|
| Input | Plate Index | unsigned integer | The index of the force plate |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | ForcePlateSubsamples | unsigned integer | The number of subsamples. |

**C++**

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

```
// class Output_GetForcePlateSubsamples
// {
// public:
//   Result::Enum Result;
//   unsigned int ForcePlateSubsamples;
// };
//
// Output_GetForcePlateSubsamples
//    GetForcePlateSubsamples( const unsigned int ForcePlateIndex )
const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetForcePlateSubsamples Output =
MyClient.GetForcePlateSubsamples ( 0 );
                                              // Output.Result == Success
                                              //
Output.ForcePlateSubsamples >= 0
```

**MATLAB**

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

```
% [Output] = GetForcePlateSubsamples()

MyClient = Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient. GetForcePlateSubsamples( 1 );
                                          // Output.Result == Success
                                          // Output.
ForcePlateSubsamples >= 0
```

**.NET**

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

```
// public class Output_GetForcePlateSubsamples
// {
//   public Result Result;
//   public uint   ForcePlateSubsamples;
// };
//
// Output_GetForcePlateCount GetForcePlateSubsamples( unsigned int
ForcePlateIndex );
```

| | |
|---|---|
| | ```<br>ViconDataStreamSDK.DotNET.Client MyClient =<br>new ViconDataStreamSDK.DotNET.Client();<br>MyClient.EnableDeviceData();<br>MyClient.Connect( "localhost" );<br>MyClient.GetFrame();<br><br>Output_GetForcePlateSubsamples Output = MyClient.<br>GetForcePlateSubsamples ( 0 );<br>// Output.Result == Success<br>// Output.<br>ForcePlateSubsamples >= 0<br>``` |

# GetGlobalForceVector2

*Return the force vector for the plate in global co-ordinates. This version takes a subsample index that allows access to all of the force information.*

*The vector is in Newtons and is with respect to the global coordinate system regardless of the orientation of the plate. The vector represents the force exerted upon the plate, not the reaction force.*

*See Also : GetGlobalMomentVector, GetGlobalCentreOfPressure*

| | | | |
|---|---|---|---|
| Input | Force Plate Index | unsigned integer | The index of the plate |
| | Subsample | unsigned integer | The subsample to access |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | ForceVector | double[3] | The force on the plate |

C++

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

A valid Subsample is between 0 and GetForcePlateSubsamples()-1

```
// class Output_GetGlobalForceVector
// {
// public:
//   Result::Enum Result;
//   double       ForceVector[ 3 ];
// };
//
// Output_GetGlobalForceVector
//   GetGlobalForceVector (
//       const unsigned int ForcePlateIndex, const unsigned int
Subsample ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

const unsigned int Index(0);
const unsigned int Samples = MyClient.GetForcePlateSubsamples( index
).ForcePlateSubsamples;
for( unsigned int Sample = 0; Sample < Samples; ++ Sample)
{
    Output_GetGlobalForceVector Output = MyClient.GetGlobalForceVector(
Index, Sample );
}
```

MATLAB

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

A valid Subsample is between 1 and GetForcePlateSubsamples()

```
% [Output] = GetGlobalForceVector( ForcePlateIndex, Subsample )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();
Index = 0;
Output_GetForcePlateSubsamples = MyClient.GetForcePlateSubsamples(
```

| | |
|---|---|
| | Index );<br>for Sample = 1:Output_GetForcePlateSubsamples.ForcePlateSubsamples<br>  Output = MyClient. GetGlobalForceVector( Index, Sample );<br>end |
| .NET | A valid ForcePlateIndex is between 0 and GetForcePlateCount() – 1<br>A valid Subsample is between 0 and GetForcePlateSubsamples()-1<br>// public ref class Output_GetGlobalForceVector<br>// {<br>// public:<br>//  Result  Result;<br>//  array< double >^ ForceVector;<br>// };<br>//<br>// Output_GetGlobalForceVector<br>//  GetGlobalForceVector( uint ForcePlateIndex, uint Subsample )<br>const;<br><br>ViconDataStreamSDK.DotNET.Client MyClient = new<br>ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient.EnableUnlabeledMarkerData();<br>MyClient.GetFrame();<br><br>uint Index = 0;<br>uint Samples =<br>MyClient.GetForcePlateSubsamples(ForcePlateIndex).ForcePlateSubsamples;<br>for (uint Sample = 0; Sample < Samples; ++ Sample)<br>{<br>  Output_GetGlobalForceVector Output = MyClient.GetGlobalForceVector(<br>Index, Sample );<br>} |

# GetGlobalMomentVector2

*Return the moment vector for the plate in global co-ordinates. This version takes a subsample index that allows access to all of the force information.*

*The vector is in Newton-Meters and is with respect to the global coordinate system regardless of the orientation of the plate.*

*The vector represents the moment exerted upon the plate, not the reaction moment. Any force plate origin offset is accounted for in the moments so they are acting about the exact centre of the top surface of the plate.*

*See Also : GetGlobalForceVector, GetGlobalCentreOfPressure*

| | | | |
|---|---|---|---|
| Input | Plate Index | unsigned integer | The index of the force plate |
| | Subsample | unsigned integer | The subsample to access |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | MomentVector | double[3] | The moment exterted on the plate |

**C++**

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

A valid Subsample is between 0 and GetForcePlateSubsamples()-1

```
// class Output_GetGlobalMomentVector
// {
// public:
//   Result::Enum Result;
//   double       MomentVector[ 3 ];
// };
//
// Output_GetGlobalMomentVector GetGlobalMomentVector (
//       const unsigned int ForcePlateIndex, const unsigned int
Subsample ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

const unsigned int Index(0);
const unsigned int Samples = MyClient.GetForcePlateSubsamples( index
).ForcePlateSubsamples;
for( unsigned int Sample = 0; Sample < Samples; ++ Sample)
{
    Output_GetGlobalMomentVector  Output =
MyClient.GetGlobalMomentVector( Index, Sample );
}
```

**MATLAB**

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

A valid Subsample is between 1 and GetForcePlateSubsamples()

```
% [Output] = GetGlobalMomentVector( ForcePlateIndex, Subsample )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
```

| | |
|---|---|
| | MyClient.GetFrame();<br><br>Index = 0;<br>Output_GetForcePlateSubsamples = MyClient.GetForcePlateSubsamples(<br>Index );<br>for Sample = 1:Output_GetForcePlateSubsamples.ForcePlateSubsamples<br>  Output = MyClient. GetGlobalMomentVector ( Index, Sample );<br>end |
| .NET | A valid ForcePlateIndex is between 0 and GetForcePlateCount() - 1<br>A valid Subsample is between 0 and GetForcePlateSubsamples()-1<br>//  public ref class Output_GetGlobalMomentVector<br>//  {<br>//  public:<br>//  Result  Result;<br>//  array< double >^ MomentVector;<br>//  };<br>// Output_GetGlobalMomentVector<br>//  GetGlobalMomentVector( uint ForcePlateIndex, uint Subsample )<br>const;<br><br>ViconDataStreamSDK.DotNET.Client MyClient = new<br>ViconDataStreamSDK.DotNET.Client();<br>MyClient.Connect( "localhost" );<br>MyClient. EnableDeviceData ();<br>MyClient.GetFrame();<br><br>uint Index = 0;<br>uint Samples =<br>MyClient.GetForcePlateSubsamples(ForcePlateIndex).ForcePlateSubsamples;<br>for (uint Sample = 0; Sample < Samples; ++ Sample)<br>{<br>  Output_GetGlobalMomentVector Output =<br>MyClient.GetGlobalMomentVector( Index, Sample );<br>} |

# GetGlobalCentreOfPressure$_2$

| | | | |
|---|---|---|---|
| *Return the centre of pressure for the plate in global co-ordinates. This version takes a subsample index that allows access to all of the force information.*<br>*The position is in millimeters and is with respect to the global coordinate system.*<br>*See Also : GetGlobalForceVector, GetGlobalMomentVector* | | | |
| Input | Plate Index | unsigned integer | The index of the force plate |
| | Subsample | unsigned integer | The subsample to access |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | CentreOfPressure | double[3] | The CoP. |

C++

A valid ForcePlateIndex is between 0 and GetForcePlateCount()-1

A valid Subsample is between 0 and GetForcePlateSubsamples()-1

```
// class Output_GetGlobalCentreOfPressure
// {
// public:
//   Result::Enum Result;
//   double       CentreOfPressure[ 3 ];
// };
//
// Output_GetGlobalCentreOfPressure
//   GetGlobalCentreOfPressure (
//       const unsigned int ForcePlateIndex, const unsigned int
Subsample ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

const unsigned int Index(0);
const unsigned int Samples = MyClient.GetForcePlateSubsamples( index
).ForcePlateSubsamples;
for( unsigned int Sample = 0; Sample < Samples; ++ Sample)
{
  Output_GetGlobalCentreOfPressure Output =
MyClient.GetGlobalCentreOfPressure(Index,Sample);
}
```

MATLAB

A valid ForcePlateIndex is between 1 and GetForcePlateCount()

A valid Subsample is between 1 and GetForcePlateSubsamples()

```
% [Output] = GetGlobalCentreOfPressure( ForcePlateIndex, Subsample )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Index = 0;
Output_GetForcePlateSubsamples = MyClient.GetForcePlateSubsamples(
Index );
for Sample = 1:Output_GetForcePlateSubsamples.ForcePlateSubsamples
```

<table>
<tr><td></td><td>

```
    Output = MyClient.GetGlobalCentreOfPressure( Index, Sample );
end
```

</td></tr>
<tr><td>.NET</td><td>

A valid ForcePlateIndex is between 0 and GetForcePlateCount() – 1

A valid Subsample is between 0 and GetForcePlateSubsamples()-1

```
// public class Output_ GetGlobalCentreOfPressure
// {
//  public:
//    Result           Result;
//    array< double >^ CentreOfPressure;
// };
//
// Output_GetGlobalCentreOfPressure
//   GetGlobalCentreOfPressure( uint ForcePlateIndex, uint Subsample )
const;

ViconDataStreamSDK.DotNET.Client MyClient =  new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient. EnableDeviceData ();
MyClient.GetFrame();

uint Index = 0;
uint Samples =
MyClient.GetForcePlateSubsamples(ForcePlateIndex).ForcePlateSubsamples;
for (uint Sample = 0; Sample < Samples; ++ Sample)
{
  Output_GetGlobalCentreOfPressure Output = MyClient.
GetGlobalCentreOfPressure (Index,Sample);
}
```

</td></tr>
</table>

# GetEyeTrackerCount

| | | | |
|---|---|---|---|
| *Return the number of eye trackers available in the DataStream.*<br>*See Also : GetEyeTrackerGlobalGazeVector, GetEyeTrackerGlobalGazeVector* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | Eye Tracker Count | unsigned integer | The number of eye trackers |

C++

```
// class Output_GetEyeTrackerCount
// {
// public:
//   Result::Enum Result;
//   unsigned int EyeTrackerCount;
// };
//
// Output_GetEyeTrackerCount GetEyeTrackerCount() const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetDeviceCount Output = MyClient. GetEyeTrackerCount ();
// Output.Result == Success
// Output. EyeTrackerCount >= 0
```

MATLAB

```
% [Output] = GetEyeTrackerCount()

MyClient = Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output = MyClient.GetEyeTrackerCount();
// Output.Result == Success
// Output.EyeTrackerCount >= 0
```

.NET

```
// public class Output_GetEyeTrackerCount
// {
//   public Result Result;
//   public uint   EyeTrackerCount;
// };
//
// Output_GetEyeTrackerCount GetEyeTrackerCount();

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.EnableDeviceData();
MyClient.Connect( "localhost" );
MyClient.GetFrame();

Output_GetEyeTrackerCount Output = MyClient.GetEyeTrackerCount();
// Output.Result == Success
// Output.EyeTrackerCount >= 0
```

# GetEyeTrackerGlobalPosition

*Returns the location of the eye. The position is in Millimeters with respect to the global origin. The segment and device data need to be enabled to get the position.*

*See Also : GetEyeTrackerCount, GetEyeTrackerGlobalGazeVector*

| | | | |
|---|---|---|---|
| Input | EyeTrackerIndex | unsigned integer | The index of the eye tracker |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | Position | double[3] | The eye position |
| | Occluded | boolean | This is true if the segment that has the eye tracker attached is not visible. If true the position will be (0,0,0). |

**C++**

A valid EyeTrackerIndex is between 0 and GetEyeTrackerCount()-1

```
//  class Output_GetEyeTrackerGlobalPosition
//  {
//  public:
//    Result::Enum Result;
//    double       Position[ 3 ];
//    bool         Occluded;
//  };
//  Output_GetEyeTrackerGlobalPosition GetEyeTrackerGlobalPosition(
//
const unsigned int EyeTrackerIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableSegmentData ();
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetEyeTrackerGlobalPosition Output =
MyClient.GetEyeTrackerGlobalPosition ( 0 );
```

**MATLAB**

A valid EyeTrackerIndex is between 1 and GetEyeTrackerCount()

```
% [Output] = GetEyeTrackerGlobalPosition ( EyeTrackerIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableSegmentData ();
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output = MyClient. GetEyeTrackerGlobalPosition ( 1 );
```

**.NET**

A valid EyeTrackerIndex is between 0 and GetEyeTrackerCount() - 1

```
//  public ref class Output_GetEyeTrackerGlobalPosition
//  {
//  public:
//    Result           Result;
//    array< double >^ Position;
//    bool             Occluded;
```

```
//  };
//    Output_GetEyeTrackerGlobalPosition^ GetEyeTrackerGlobalPosition(
//                                                unsigned int
EyeTrackerIndex )

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient. EnableSegmentData ();
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetEyeTrackerGlobalPosition Output =
MyClient.GetEyeTrackerGlobalPosition ( 0 );
```

# GetEyeTrackerGlobalGazeVector

*Returns the gaze direction as a unit vector in global coordinates. The gaze vector will be marked as occluded if the segment that has the eye tracker attached is not visible, the eye tracker is not calibrated or the pupil is not found. The segment and device data need to be enabled to get the gaze vector.*

*See Also : GetEyeTrackerCount, GetEyeTrackerGlobalPosition*

| | | | |
|---|---|---|---|
| Input | EyeTrackerIndex | unsigned integer | The index of the eye tracker |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | GazeVector | double[3] | The gaze direction vector |
| | Occluded | boolean | This is true if gaze vector could not be calculated. If false the position will be (0,0,0). |

**C++**

A valid EyeTrackerIndex is between 0 and GetEyeTrackerCount()-1

```
//  class Output_GetEyeTrackerGlobalGazeVector
//  {
//  public:
//    Result::Enum Result;
//    double       GazeVector [ 3 ];
//    bool         Occluded;
//  };
//  Output_GetEyeTrackerGlobalGazeVector GetEyeTrackerGlobalGazeVector(
//                                       const unsigned int
EyeTrackerIndex ) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableSegmentData ();
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetEyeTrackerGlobalPosition Output =
MyClient.GetEyeTrackerGlobalGazeVector ( 0 );
```

**MATLAB**

A valid EyeTrackerIndex is between 1 and GetEyeTrackerCount()

```
% [Output] = GetEyeTrackerGlobalGazeVector ( EyeTrackerIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableSegmentData ();
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output = MyClient. GetEyeTrackerGlobalGazeVector ( 1 );
```

**.NET**

A valid EyeTrackerIndex is between 0 and GetEyeTrackerCount() - 1

```
//  public ref class Output_GetEyeTrackerGlobalPosition
//  {
//  public:
//    Result           Result;
//    array< double >^ Position;
```

```
//    bool             Occluded;
//  };
//    Output_GetEyeTrackerGlobalPosition^ GetEyeTrackerGlobalPosition(
//                                        unsigned int EyeTrackerIndex )

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient. EnableSegmentData ();
MyClient. EnableDeviceData ();
MyClient.GetFrame();

Output_GetEyeTrackerGlobalPosition Output =
MyClient.GetEyeTrackerGlobalPosition ( 0 );
```

# GetCameraCount

| | | | |
|---|---|---|---|
| *Return the number of camera available in the DataStream.*<br>*See Also : GetCameraName, GetCentroidCount, GetCentroidPosition* | | | |
| Input | | | |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame` |
| | CameraCount | unsigned integer | The number of cameras |

C++

```
//  class Output_GetCameraCount
//  {
//  public:
//    Result::Enum Result;
//    unsigned int CameraCount;
//  };
//  Output_GetCameraCount GetCameraCount()

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

Output_GetCameraCount Output = MyClient.GetCameraCount();
// Output.Result == Success
// Output.CameraCount >= 0
```

MATLAB

```
% [Output] = GetCameraCount()

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData();
MyClient.GetFrame();

Output = MyClient. GetCameraCount();
% Output.Result == Success, Output.CameraCount >= 0
```

.NET

```
//  public ref class Output_GetCameraCount
//  {
//  public:
//    Result           Result;
//    unsigned int CameraCount;
//  };
//    Output_GetCameraCount GetCameraCount()

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

Output_GetCameraCount Output =  MyClient.GetCameraCount();
// Output.Result == Success
// Output.CameraCount >= 0
```

# GetCameraName

*Return the name of a camera. This name can be passed into centroid functions.*
*See Also : GetCameraCount, GetCentroidCount, GetCentroidPosition*

| | | | |
|---|---|---|---|
| Input | EyeTrackerIndex | unsigned integer | The index of the eye tracker |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidIndex` |
| | CameraName | string | The name of the camera |

**C++**

A valid CameraIndex is between 0 and GetCameraCount()-1

```
//  class Output_CameraName
//  {
//  public:
//    Result::Enum Result;
//    String       CameraName;
//  };
//  Output_GetCameraName GetCameraName( const unsigned int CameraIndex
) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

Output_GetCameraCount OutputGCC = MyClient.GetCameraCount();
                                    // OutputGCC.Result == Success
                                    // OutputGCC.CameraCount == 1

Output_GetCameraName OutputGCN;
OutputGCN = MyClient.GetCameraName( 0 )
```

**MATLAB**

A valid CameraIndex is between 1 and GetCameraCount()

```
% [Output] = GetCameraName ( CameraIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

OutputGCC = MyClient. GetCameraCount ( 1 );
                                    % OutputGCC.Result == Success
                                    % OutputGCC.CameraCount == 1
OutputGCN = MyClient.GetCameraName( 1 );
```

**.NET**

A valid CameraIndex is between 0 and GetCameraCount() - 1

```
//  public ref class Output_GetCameraName
//  {
//  public:
//    Result           Result;
//    String^          CameraName;
//  };
//    Output_GetCameraName^ GetCameraName( unsigned int CameraIndex )

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
```

| | |
|---|---|
| | MyClient. EnableCentroidData ();<br>MyClient.GetFrame();<br><br>Output_GetCameraCount OutputGCC = MyClient.GetCameraCount();<br>// OutputGCC.Result == Success<br>// OutputGCC.CameraCount == 1<br><br>Output_GetCameraName OutputGCN;<br>OutputGCN = MyClient.GetCameraName( 0 ) |

# GetCentroidCount

*Returns the number of centroids reported by a named camera. The centroid data needs to be enabled to get the number of centroids.*

*See Also : GetCameraCount, GetCameraName, GetCentroidPosition*

| | | | |
|---|---|---|---|
| Input | CameraName | string | The name of the camera |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidCameraName` |
| | CentroidCount | unsigned integer | The number of centroids |

C++

```
//  class Output_GetCentroidCount
//  {
//  public:
//    Result::Enum Result;
//    unsigned int CentroidCount;
//  };
//  Output_GetCentroidCount GetCentroidCount( const std::string &
CameraName ) const

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

Output_GetCameraCount OutputGCC = MyClient.GetCameraCount();
for( unsigned int CameraIndex = 0; CameraIndex < OutputGCC.CameraCount;
++CameraIndex )
{
  Output_GetCameraName OutputGCN = MyClient.GetCameraName( CameraIndex
);
  Output_GetCentroidCount OutputGCeC = MyClient.GetCentroidCount(
OutputGCN.CameraName );
                                                  // OutputGCeC.Result ==
Success
                                                  // OutputGCeC.CentroidCount
>= 0
}
```

MATLAB

```
% [Output] = GetCentroidCount( CameraName )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData();
MyClient.GetFrame();

OutputGCC = MyClient.GetCameraCount();
for CameraIndex = 1:OutputGCC.CameraCount
  OutputGCN = MyClient.GetCameraName( CameraIndex );
  OutputGCeC = MyClient.GetCentroidCount( OutputGCN.CameraName )
                                                  % OutputGCeC.Result ==
Success
                                                  % OutputGCeC.CentroidCount
>= 0
End
```

.NET

```
//  public ref class Output_GetCentroidCount
//  {
```

```
//  public:
//    Result           Result;
//    unsigned int  CentroidCount;
//  };
//    Output_GetCentroidCount GetCentroidCount( String^ CameraName )

ViconDataStreamSDK.DotNET.Client MyClient = new
ViconDataStreamSDK.DotNET.Client();
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

Output_GetCameraCount OutputGCC =  MyClient.GetCameraCount();
for( unsigned int CameraIndex = 0; CameraIndex < OutputGCC.CameraCount;
++CameraIndex )
{
  OutputGCN = MyClient.GetCameraName( CameraIndex );
  OutputGCeC = MyClient.GetCentroidCount( OutputGCN.CameraName )
                                                % OutputGCeC.Result ==
Success
                                                % OutputGCeC.CentroidCount
>= 0
}
```

# GetCentroidPosition

*Returns the position and radius of the centroid in camera coordinates. The centroid data needs to be enabled to get the centroid position and radius.*

*See Also : GetCameraCount, GetCameraName, GetCentroidCount*

| | | | |
|---|---|---|---|
| Input | CameraName | string | The name of the camera |
| | CentroidIndex | unsigned integer | The index of the centroid |
| Output | Result | Result | `Result.Success`<br>`Result.NotConnected`<br>`Result.NoFrame`<br>`Result.InvalidCameraName`<br>`Result.InvalidIndex` |
| | CentroidPosition | double[2] | The position of the centroid |
| | Radius | double | The radius of the centroid |

**C++**

A valid CameraName is obtained from GetCameraName( CameraIndex )

A valid CentroidIndex is between 0 and GetCentroidCount( CameraName )-1

```
//  class Output_GetCentroidPosition
//  {
//  public:
//    Result::Enum Result;
//    double       CentroidPosition [ 2 ];
//    double       Radius;
//  };
//  Output_GetCentroidPosition GetCentroidPosition (
//……..const std::string & CameraName
//                                      const unsigned int CentroidIndex
) const;

ViconDataStreamSDK::CPP::Client MyClient;
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

Output_GetCameraName OutputGCN = MyClient.GetCameraName( 0 );
Output_GetCentroidPosition Output = MyClient.GetCentroidPosition(
OutputGCN.CameraName, 0 );
```

**MATLAB**

A valid CameraName is obtained from GetCameraName( CameraIndex )

A valid CentroidIndex is between 1 and GetCentroidCount( CameraName )

```
% [Output] = GetCentroidPosition( CameraName, CentroidIndex )

MyClient = Client();
MyClient.Connect( "localhost" );
MyClient. EnableCentroidData ();
MyClient.GetFrame();

OutputGCN = MyClient.GetCameraName( 1 );
Output = MyClient.GetCentroidPosition( OutputGCN.CameraName, 1 );
```

**.NET**

A valid CameraName is obtained from GetCameraName( CameraIndex )

| | |
|---|---|
| | A valid CentroidIndex is between 0 and GetCentroidCount( CameraName )-1<br>`//  public ref class Output_GetCentroidPosition`<br>`//  {`<br>`//  public:`<br>`//    Result           Result;`<br>`//    array< double >^ Position;`<br>`//    double                 Radius;`<br>`//  };`<br>`//    Output_GetCentroidPosition^ GetCentroidPosition(`<br>`//                                    String^ CameraName`<br>`//                                    unsigned int CentroidIndex )`<br><br>`ViconDataStreamSDK.DotNET.Client MyClient = new`<br>`ViconDataStreamSDK.DotNET.Client();`<br>`MyClient.Connect( "localhost" );`<br>`MyClient. EnableCentroidData ();`<br>`MyClient.GetFrame();`<br><br>`Output_GetCameraName OutputGCN = MyClient.GetCameraName( 0 );`<br>`Output_GetCentroidPosition Output = MyClient.GetCentroidPosition(`<br>`OutputGCN.CameraName, 0 );` |

# Appendix A – What's New

## What's New in Version 1.0

- Full access to analog device data in Nexus.  This can be scaled data or raw voltages.
- One SDK for all applications.
- Four segment rotation options: Quaternion, 3x3 row-major Matrix, Helical, and EulerXYZ format.
- Support streaming, request, and pre-fetch modes.
- Formats specific to C++, MATLAB and .NET.
- Version control.
- Result feedback for success criteria.

## What's New in Version 1.0.1

- C++ programs that access the DS-SDK dll files can now be complied in Debug mode.
- New function calls for Vicon Tracker ***
  - ConnectToMulticast
  - StartTransmittingMulticast
  - StopTransmittingMulticast
  - GetLatencyTotal
  - GetLatencySampleCount
  - GetLatencySampleName
  - GetLatencySampleValue

  *** These functions will not work with Vicon Nexus 1.4 and Vicon Blade 1.6.

## What's New in Version 1.1.0

- Release of C++ and .NET SDKs on Windows x64.
- Release of C++ SDK on Linux x86.
- New function calls
  - DisableSegmentData
  - DisableMarkerData
  - DisableUnlabeledMarkerData
  - DisableDeviceData
  - GetMarkerParentName
  - GetSubjectRootSegmentName
  - GetSegmentParentName
  - GetSegmentChildCount
  - GetSegmentChildName
  - GetSegmentStaticTranslation
  - GetSegmentStaticRotationHelical
  - GetSegmentStaticRotationMatrix
  - GetSegmentStaticRotationQuaternion

- GetSegmentStaticRotationEulerXYZ

- Corrected some units. The values given by the SDK have not changed – they were incorrectly labeled in previous versions.
  - "NewtonMillimetre" has become "NewtonMeter"
  - "Millimetre" has become "Meter"

- Corrected segment rotations following calls to SetAxisMapping()
- Added command-line options for the Test programs to specify a host to connect to.

## What's New in Version 1.2.0

- Added C++ Linux x64 support
- Fix to support of .NET under Windows x64
- New function calls:
  - GetForcePlateCount
  - GetGlobalForceVector
  - GetGlobalMomentVector
  - GetGlobalCentreOfPressure

- Minor improvements to documentation.

## What's New in Version 1.3.0

- New function calls:
  - GetFrameRate
  - GetEyeTrackerCount
  - GetEyeTrackerGlobalPosition
  - GetEyeTrackerGlobalGazeVector
  - GetDeviceOutputSubsamples
  - GetForcePlateSubsamples

- New overrides to function calls to allow access to all the analogue data:
  - GetDeviceOutputValue
  - GetGlobalForceVector
  - GetGlobalMomentVector
  - GetGlobalCentreOfPressure

- Minor improvements to documentation.
- Added Mac OSX support.

## What's New in Version 1.4.0

- New function calls:
  - SetApexDeviceFeedback

# What's new in version 1.5.0

- New function calls:
  - EnableCentroidData
  - DisableCentroidData
  - IsCentroidDataEnabled
  - GetCameraCount
  - GetCameraName
  - GetCentroidCount
  - GetCentroidPosition

# What's new in version 1.6.0

SetAxisMapping() now works correctly with Blade (3.4+) regardless of whether the coordinate system in Blade is set to Y Up or Z Up.
New functions that Expose Marker Labels per Centroid information are now implemented. These allow the user to obtain data on the camera contributions of each marker within a client based upon the SDK.

- EnableMarkerRayData
- DisableMarkerRayData
- IsMarkerRayDataEnabled
- GetMarkerRayContributionCount
- GetMarkerRayContribution