zForce Message Specification

Definition and Encoding

The message structure that is used in the zForce communication protocol is defined in the zForce PDU definition file. PDU stands for Protocol Data Unit, the specific block of information that is going to be transferred. The definition is written according to Abstract Syntax Notation One (ASN.1), a standardized way to describe data regardless of language implementation, hardware system and operation system (ISO/IEC 8824).

To get the message data format used for information transfer, a set of encoding rules are applied to the ASN.1 definitions. The zForce communication protocol uses the Distinguished Encoding Rules (DER) to serialize the information that is going to be transferred. For more information, refer to Understanding the zForce ASN1 Protocol.

ASN.1 defines a number of universal types, from which all other types inherit traits. The result is that a set of encoding rules that covers the universal types can serialize any PDU, as long as the identifier numbers and the definition categories are available. The identifier numbers and definition categories for the zForce PDU are included in the PDU definition file.

Downloading the definition file

Download the zForce® PDU definition file here.

The zForce message

The communication protocol uses three types of messages:

- Requests
- Responses
- Notifications

The host sends a request to the sensor module, and the device responds with a response. The device may send notifications to the host at any time.

All messages contain a virtual device address. Virtual devices are functionally isolated from each other, and communicate separately with the host. There are two types of virtual devices:

- Platform – represents the system.
- Air – represents one Neonode Touch Sensor Module.

A sensor module will always contain one platform virtual device and can contain any number of instances of other virtual device types.

In the definition file, all sensor messages are described as instances of the top level PDU `ProtocolMessage` which have three child PDUs:

PDU	Content	Explanation
Request	deviceAddress	Specifies the virtual device within the sensor module that receives the request.
	command	The command specifies what is being requested.
Response	deviceAddress	Specifies the virtual device within the sensor module that sends the response.
	command	The command contains the result or requested information of the request.
Notification	deviceAddress	Specifies the virtual device within the sensor module that sends the notification.
	notificationMessage	The message from the sensor module to the host.
	notificationTimestamp (optional)	Timestamp.

PDU Description in GSER Notation

The PDU specifies the following message specification templates, notifications and pairs of requests and responses:

- Application Interface
- Device Information
- Device Count
- Frequency
- Touch Sensor
- Operation Mode
- Touch Format
- Enable Execution

- Touch Notifications
- Information
- Configuration

The messages described here are encoded using the man-readable Generic String Encoding Rules (GSER), as the messages encoded according to DER can be difficult to read for a human.

A short introduction to the GSER notation:

- Sequences and/or sets (items containing sub items) are shown as curly brackets: **{ <sub elements> }**
- Values encoded as octet strings are written as hexadecimal octets enclosed within single quotes and suffixed with H: **'FF9900'H**
- Bit strings are also shown as octet strings when the number of bits is a multiple of 8, otherwise each bit is shown as a single 1 or 0, and suffixed with B: **'11010101001'B**
- In a choice element, the selected type is denoted by its name followed by a colon: **request:**

Refer to Generic String Encoding Rules (GSER) for ASN.1 Types for a full reference.

The tool FFASN1Dump can transcode from GSER to DER:

```
ffasn1dump -I gser -O der zforce_pdu_def.asn ProtocolMessage <input file> <output file>
```

Currently ffasn1dump does not handle identifiers for Integer values. For this reason, they need to be replaced with numerical values.

Application Interface

The application interface specifies what requests can be made and what responses and notifications they activate. Messages are specified according to the following templates:

request

```
request: {
  deviceAddress <address>,
  <request command>
}
```

response

```
response: {
  deviceAddress <address>,
  <command response>
}
```

notification

```
notification: {
  deviceAddress <address>,
  <notification>
  notificationTimestamp <timestamp>
}
```

Where

- Address is an octet string with 2 octets, device type and index: '<device type><index>'H. The available device types are:

Device type	Device
0	Platform
1	Core
2	Air

3	Plus
4	Lightning

Example: ‘0000’H for platform (there can be only one platform device per sensor system).

- Timestamp is an integer representing int16 counting at 32768 Hz.

Device Information

The deviceInformation command fetches for example the product id and the FW version. What information that is available differs depending on the type of device. The following example shows a response from a platform device.

request command

```
deviceInformation {
}
```

command response

```
deviceInformation {
  platformInformation {
    platformVersionMajor 7,
    platformVersionMinor 0,
    protocolVersionMajor 1,
    protocolVersionMinor 5,
    firmwareVersionMajor 1,
    firmwareVersionMinor 0,
    hardwareIdentifier "Demo board",
    hardwareVersion "R2",
    asicType nn1002,
    numberOfAsics 1,
    mcuUniqueIdentifier '340038000E51333439333633'H,
    projectReference "DEMO_1.0_REL",
    platformReference "734cebd",
    buildTime "16:01:14",
    buildDate "2016-07-01"
      }
}
```

The fields have the following meaning:

- platformVersion: FW platform version, version 7.0 in the example.
- protocolVersion: communication protocol version, version 1.5 in the example.
- firmwareVersion: product FW version, version 1.0 in the example.
- hardware: product hardware, configuration and revision.
- asicType: which type of the Neonode optical scanner ASIC is used, and count.
- mcuUniqueIdentifier: identifier created at mcu manufacturing.
- projectReference: FW GIT tags or hashes. Product specific. Uniquely identifies the FW revision.
- platformReference: FW GIT tags or hashes. Uniquely identifies generic firmware base commit for the platform.
- buildTime: time of build in Central European Time, a string.
- buildDate: date of build, a string.

Device Count

The deviceCount command enumerates the available virtual devices.

request command

```
deviceCount {
}
```

command response

```
deviceCount {
  totalNumberOfDevices 1,
  airDevices 1
}
```

Device type instances are indexed from zero. The response shown here means that the only virtual device available is Air[0].

Frequency

The frequency command changes the update frequency of all sensor modules globally, that is for all devices on all platforms.

The following update frequencies can be set, if enabled in the product:

- finger: activated when objects with characteristics matching regular fingers are detected.
- stylus: activated for narrow stylus-like objects. (Not enabled for Neonode Touch Sensor Module.)
- idle: activated when no objects are detected, in order to minimize power usage.

The unit is Hz.

request command

```
frequency {
  finger 30,
  idle 10
}
```

The response contains the current frequency settings of the product:

command response

```
frequency {
  finger 30,
  idle 10
}
```

In this example, the sensor module update frequency will be 30 Hz as long as finger-like objects were recently detected. When no objects are detected, the frequency will drop to 10 Hz.

Touch Sensor

There are a number of different sensor module products that can co-exist on the same physical device. There are some product-specific commands, but the ones listed here are general.

The Touch Sensor Module will be used as example, which means that the device address will be the first Air virtual device

address

```
'0200'H
```

Operation Mode

The operationMode command sets what processing to perform on the sensor modules signals, and what diagnostics that are exposed.

The following example sets the operation mode to normal object detection:

request command

```
operationMode {
  detection TRUE,
  signals FALSE,
  ledLevels FALSE,
  detectionHid FALSE,
  gestures FALSE
}
```

command response

```
operationMode {
  detection TRUE,
  signals FALSE,
  ledLevels FALSE,
  detectionHid FALSE
}
```

As can be seen gestures are missing in the response. This is a valid response, since the device is built with a subset of the protocol, or an older forward-compatible version.

Touch Format

The touchFormat command retrieves the binary format of the detected objects.

request command

```
touchFormat {
}
```

command response

```
touchFormat {
  touchDescriptor { id, event, loc-x-byte1, loc-x-byte2, loc-y-byte1, loc-y-byte2, size-x-byte1, size-y-byte1
}
}
```

The touchDescriptor is a bit string, where each bit signifies one byte of payload being included in the touchNotification octet strings. A touchNotification is the concatenation of those bytes. The following table lists all bits. Bits used in the example are marked green.

Name	Description	Comment
id	Touch Identifier	
event	Up/Down/Move	0=Down; 1=Move; 2=Up; 3=Invalid; 4=Ghost
loc-x-byte1	X coordinate	
loc-x-byte2	X expanded	for higher precision
loc-x-byte3	X expanded	for higher precision
loc-y-byte1	Y coordinate	
loc-y-byte2	Y expanded	for higher precision
loc-y-byte3	Y expanded	for higher precision

loc-z-byte1	Z coordinate	
loc-z-byte2	Z expanded	for higher precision
loc-z-byte3	Z expanded	for higher precision
size-x-byte1	X size	
size-x-byte2	X size	for higher precision
size-x-byte3	X size	for higher precision
size-y-byte1	Y size	
size-y-byte2	Y size	for higher precision
size-y-byte3	Y size	for higher precision
size-z-byte1	Z size	
size-z-byte2	Z size	for higher precision
size-z-byte3	Z size	for higher precision
orientation	Orientation	Hand orientation
confidence	Confidence	Ignore. This value is not reliable for the Touch Sensor Module.
pressure	Pressure	

Location and size coordinates can be specified with up to 3 bytes. The byte order in decreasing significance - big-endian. For example:

- 1 byte: location x = loc-x-byte1
- 2 bytes: location x = (loc-x-byte1 << 8) + loc-x-byte2
- 3 bytes: location x = (loc-x-byte1 << 16) + (loc-x-byte2 << 8) + loc-x-byte3

Location is signed, and size is not.

The location coordinate scale is one of two systems, depending on which detector is used:

- Physical: Robair Air and Core detectors: The unit is 0.1 mm. A coordinate value of 463 thus means 46.3 mm from origin.
- Relative: Triangles and Shape Air detectors: Fraction of the largest screen dimension as fixed point with 14 bits after the radix point (q14). On a widescreen display, the horizontal axis ranges [0, 2^{14}[, and vertical [0, 2^{14} * 9/16[([0, 16383], [0, 9215]).

Touch Sensor Module uses Robair, thus the unit is 0.1 mm.

Size is in mm.

Confidence and pressure are fractions of the full values, in percent.

Enable Execution

The enable command activates the Touch Sensor Module, and notifications of detections start to stream.

request command

```
enable {
  enable 0
}
```

command response

```
enable {
  enable
}
```

To deactivate the Touch Sensor Module, send the disable command:

request command

```
enable {
  disable NULL
}
```

command response

```
enable {
  disable NULL
}
```

Touch Notifications

A detected object is reported with a touchNotification. The touchNotification payload is a touchDescriptor bit string. Every concurrently tracked object is represented by its own touchNotification payload.

notification

```
notificationMessage touchNotifications: {
    '0001013600730A0A64'H
  },
```

The following table shows the value of the example payload interpreted with the touch descriptor.

Name	Description	Comment	Value
id	Touch Identifier		0
event	Up/Down/Move	0=Down; 1=Move; 2=Up; 3=Invalid; 4=Ghost	1
loc-x-byte1	X coordinate		1
loc-x-byte2	X expanded	for higher precision	54
loc-y-byte1	Y coordinate		0
loc-y-byte2	Y expanded	for higher precision	115
size-x-byte1	X size		10
size-y-byte1	Y size		10

The touchNotification is from a Core device and translates to "Object 0 moved. Location is (31.0, 11.5) mm. Size is 10x10 mm."

Information

The command deviceInformation retrieves some information about the virtual device instance.

request command

```
deviceInformation {
}
```

command response

```
deviceInformation {
  deviceInstanceInformation {
    productVersionMajor 1,
    productVersionMinor 38,
    physicalWidth 1584,
    physicalHeight 1341,
    numberOfSignalAxes 0
  }
}
```

The response contains the deviceInstanceInformation structure, with the following parts:

Part	Description
productVersion	The specific type version of the virtual device.
physical	Size in unit 0.1 mm. See section Touch Format for the relationship to location coordinates.
numberOfSignalAxes	Only applicable for Core devices. The number of sensor arrays, each monitoring one dimension/axis of a touch sensor. Generally 2.

Configuration

Some configurations of the Touch Sensor Module can be changed at run-time. The deviceConfiguration request command and command response are identical, except some configuration items in the request may be omitted in order to leave them in their current state.

For instance, to set object size restrictions only, omit all other items:

request command

```
deviceConfiguration {
  sizeRestriction {
    maxSizeEnabled TRUE,
    maxSize 100,
    minSizeEnabled FALSE
  }
}
```

The command response contains the state of all configuration items:

command response

```
deviceConfiguration {
  subTouchActiveArea {
    lowBoundX 0,
    lowBoundY 0,
    highBoundX 1584,
    highBoundY 1341,
    reverseX FALSE,
    reverseY FALSE,
    flipXY FALSE,
    offsetX 0,
    offsetY 0
  },
  sizeRestriction {
    maxSizeEnabled FALSE,
    maxSize 0,
    minSizeEnabled FALSE,
    minSize 0
  },
  detectionMode default,
  numberOfReportedTouches 2,
  hidDisplaySize {
    x 1584,
    y 1341
  }
}
```

The items are:

- subTouchActiveArea: Crop the sensor module to a rectangle between the specified low and high coordinates in each dimension. Offset can be applied and flip the X and Y axis. Origin of reported locations is set to low coordinates, or if reversed, the high coordinate with increasing coordinates toward low.
- sizeRestriction: Limit detection to objects within this size range. Unit is 0.1 mm.
- detectionMode, one of the following:
    - default: finger and stylus
    - finger: Finger only
    - mergeTouches: Merges all touch objects into one
    - insensitiveFTIR: Unsupported
- numberOfReportedTouches: Maximum number of reported tracked objects.
- hidDisplaySize: Scaling the coordinate system when using the sensor module in HID Touch Digitizer mode.

Read More about the Communication Protocol

- Serialization Protocol Quick Start
- USB HID Transport
- I2C Transport
- zForce Message Specification
- Understanding the zForce ASN.1 Serialization Protocol

Read More