

JSPacket

Property Summary

Number

[index](#)
Returns the current position in the byte array of the packet.

Method Summary

byte[]

[getByteArray\(\)](#)
Returns the content of the package into a byte array.

String

[getHost\(\)](#)
Returns the name of the host that sent the packet.

Number

[getLength\(\)](#)
Returns the length of the packet in bytes.

Number

[getPort\(\)](#)
Returns the port where the packet originated from.

Number

[readByte\(\)](#)
Reads an 8 bits byte value from the packet, starting from the current index.

Number

[readInt\(\)](#)
Reads a 32 bits int value from the packet, starting from the current index.

Number

[readShort\(\)](#)
Reads a 32 bits short value from the packet, starting from the current index.

String

[readUTF\(\)](#)
Reads a UTF-8 string from the packet, starting from the current index.

String

[readUTF\(length\)](#)
Reads a UTF-8 string from the packet, starting from the current index.

void

[writeByte\(number\)](#)
Writes one byte into the packet, at the current index.

void

[writeBytes\(bytes\)](#)
Writes an array of bytes into the packet, at the current index.

void

[writeInt\(number\)](#)
Writes a 32 bits int into the packet, at the current index.

void

[writeShort\(number\)](#)
Writes a 16 bits short value into the packet, at the current index.

Number

[writeUTF\(string\)](#)
Writes an UTF-8 encoded string into the packet, at the current index.

Property Details

index

Returns the current position in the byte array of the packet. The next read/write operation will occur at this position.
This is a 0 based index.

Returns

Number

Sample

```
var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('an int is: ' + packet.readInt());
        application.output('moved to index: ' + packet.index);
        application.output('a short is: ' + packet.readShort());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
    }
    else {
        application.output('end of communication. ');
        break;
    }
}
```

Method Details**getBytesArray**

byte[] **getBytesArray** ()

Returns the content of the package into a byte array.

Returns

byte[]

Sample

```
var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        var bytes = packet.getBytesArray();
        application.output('received a packet of length: ' + bytes.length);
        for (var i=0; i<bytes.length; i++)
            application.output(bytes[i]);
    }
    else {
        application.output('end of communication. ');
        break;
    }
}
```

getHost

String **getHost** ()

Returns the name of the host that sent the packet.

Returns

String

Sample

```
var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('message is: ' + packet.readUTF());
    }
    else {
        application.output('end of communication.');
```

```
break;
    }
}
```

getLength**Number** `getLength()`

Returns the length of the packet in bytes.

Returns**Number****Sample**

```
var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('message is: ' + packet.readUTF());
    }
    else {
        application.output('end of communication.');
```

```
break;
    }
}
```

getPort**Number** `getPort()`

Returns the port where the packet originated from.

Returns**Number****Sample**

```
var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('message is: ' + packet.readUTF());
    }
    else {
        application.output('end of communication.');
```

```
break;
    }
}
```

readByte**Number** `readByte()`

Reads an 8 bits byte value from the packet, starting from the current index. Advances the index with one position.

Returns**Number**

Sample

```

var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('an int is: ' + packet.readInt());
        application.output('moved to index: ' + packet.index);
        application.output('a short is: ' + packet.readShort());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
    }
    else {
        application.output('end of communication. ');
        break;
    }
}

```

readInt**Number** readInt ()

Reads a 32 bits int value from the packet, starting from the current index. Advances the index with 4 positions.

Returns**Number****Sample**

```

var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('an int is: ' + packet.readInt());
        application.output('moved to index: ' + packet.index);
        application.output('a short is: ' + packet.readShort());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
    }
    else {
        application.output('end of communication. ');
        break;
    }
}

```

readShort**Number** readShort ()

Reads a 32 bits short value from the packet, starting from the current index. Advances the index with 2 positions.

Returns**Number**

Sample

```

var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('an int is: ' + packet.readInt());
        application.output('moved to index: ' + packet.index);
        application.output('a short is: ' + packet.readShort());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
        application.output('a byte is: ' + packet.readByte());
        application.output('moved to index: ' + packet.index);
    }
    else {
        application.output('end of communication.');
```

```

        break;
    }
}
```

readUTF**String readUTF ()**

Reads a UTF-8 string from the packet, starting from the current index. If an argument is specified, then it represents the length (in UTF-8 encoded bytes, not characters) of the string to read. If no argument is specified, then first a 32 bits (4 byte) int is read from the packet and that will be the byte length of the string. Advances the index with a number of positions that depends on the length of the read string.

Returns**String****Sample**

```

var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('message is: ' + packet.readUTF());
    }
    else {
        application.output('end of communication.');
```

```

        break;
    }
}
```

readUTF**String readUTF (length)**

Reads a UTF-8 string from the packet, starting from the current index. If an argument is specified, then it represents the length (in UTF-8 encoded bytes, not characters) of the string to read. If no argument is specified, then first a 32 bits (4 byte) int is read from the packet and that will be the byte length of the string. Advances the index with a number of positions that depends on the length of the read string.

Parameters**{Number} length****Returns****String**

Sample

```

var packet;
while (packet = plugins.udp.getReceivedPacket()) {
    application.output('packet received from ' + packet.getHost() + ':' + packet.getPort());
    if (packet.getLength() > 0) {
        application.output('message is: ' + packet.readUTF());
    }
    else {
        application.output('end of communication. ');
        break;
    }
}

```

writeByte

void **writeByte** (number)

Writes one byte into the packet, at the current index. The index is advanced with one position.

Parameters

{[Number](#)} number

Returns

void

Sample

```

if (!plugins.udp.startSocket('5555', packetReceived)) {
    application.output('Failed to start socket. ');
} else {
    var packet = plugins.udp.createNewPacket();
    packet.writeUTF('hello world!');
    plugins.udp.sendPacket('localhost', packet, 1234);
    packet = plugins.udp.createNewPacket();
    packet.writeByte(0xFF);
    packet.writeShort(10001);
    packet.writeInt(2000000001);
    plugins.udp.sendPacket('localhost', packet, 1234);
    var imgBytes = plugins.file.readFile('logo.jpg', 1024);
    packet = plugins.udp.createNewPacket();
    packet.writeBytes(imgBytes);
    plugins.udp.sendPacket('localhost', packet, 1234);
    plugins.udp.stopSocket();
}

```

writeBytes

void **writeBytes** (bytes)

Writes an array of bytes into the packet, at the current index. The index is advanced with a number of positions equal to the length of the written array.

Parameters

{[byte\[\]](#)} bytes

Returns

void

Sample

```

if (!plugins.udp.startSocket('5555', packetReceived)) {
    application.output('Failed to start socket.');
```

```

} else {
    var packet = plugins.udp.createNewPacket();
    packet.writeUTF('hello world!');
    plugins.udp.sendPacket('localhost', packet, 1234);
    packet = plugins.udp.createNewPacket();
    packet.writeByte(0xFF);
    packet.writeShort(10001);
    packet.writeInt(2000000001);
    plugins.udp.sendPacket('localhost', packet, 1234);
    var imgBytes = plugins.file.readFile('logo.jpg', 1024);
    packet = plugins.udp.createNewPacket();
    packet.writeBytes(imgBytes);
    plugins.udp.sendPacket('localhost', packet, 1234);
    plugins.udp.stopSocket();
}

```

writeInt

void **writeInt** (number)

Writes a 32 bits int into the packet, at the current index. The index is advances with 4 positions.

Parameters

{[Number](#)} number

Returns

void

Sample

```

if (!plugins.udp.startSocket('5555', packetReceived)) {
    application.output('Failed to start socket.');
```

```

} else {
    var packet = plugins.udp.createNewPacket();
    packet.writeUTF('hello world!');
    plugins.udp.sendPacket('localhost', packet, 1234);
    packet = plugins.udp.createNewPacket();
    packet.writeByte(0xFF);
    packet.writeShort(10001);
    packet.writeInt(2000000001);
    plugins.udp.sendPacket('localhost', packet, 1234);
    var imgBytes = plugins.file.readFile('logo.jpg', 1024);
    packet = plugins.udp.createNewPacket();
    packet.writeBytes(imgBytes);
    plugins.udp.sendPacket('localhost', packet, 1234);
    plugins.udp.stopSocket();
}

```

writeShort

void **writeShort** (number)

Writes a 16 bits short value into the packet, at the current index. The index is advances with 2 positions.

Parameters

{[Number](#)} number

Returns

void

Sample

```

if (!plugins.udp.startSocket('5555', packetReceived)) {
    application.output('Failed to start socket.');
```

```

} else {
    var packet = plugins.udp.createNewPacket();
    packet.writeUTF('hello world!');
    plugins.udp.sendPacket('localhost', packet, 1234);
    packet = plugins.udp.createNewPacket();
    packet.writeByte(0xFF);
    packet.writeShort(10001);
    packet.writeInt(2000000001);
    plugins.udp.sendPacket('localhost', packet, 1234);
    var imgBytes = plugins.file.readFile('logo.jpg', 1024);
    packet = plugins.udp.createNewPacket();
    packet.writeBytes(imgBytes);
    plugins.udp.sendPacket('localhost', packet, 1234);
    plugins.udp.stopSocket();
}

```

writeUTF

Number **writeUTF** (string)

Writes an UTF-8 encoded string into the packet, at the current index. First the length of the string is written on 4 bytes, then the string is written. The index is advanced with a number of positions equal to the length of the string plus 4.

Parameters

{[String](#)} string

Returns

[Number](#)

Sample

```

if (!plugins.udp.startSocket('5555', packetReceived)) {
    application.output('Failed to start socket.');
```

```

} else {
    var packet = plugins.udp.createNewPacket();
    packet.writeUTF('hello world!');
    plugins.udp.sendPacket('localhost', packet, 1234);
    packet = plugins.udp.createNewPacket();
    packet.writeByte(0xFF);
    packet.writeShort(10001);
    packet.writeInt(2000000001);
    plugins.udp.sendPacket('localhost', packet, 1234);
    var imgBytes = plugins.file.readFile('logo.jpg', 1024);
    packet = plugins.udp.createNewPacket();
    packet.writeBytes(imgBytes);
    plugins.udp.sendPacket('localhost', packet, 1234);
    plugins.udp.stopSocket();
}

```