

# Documentation for NORS Symbian Server

Author: Dirk Trossen  
Nokia Research Center / Helsinki

## 1. INTRODUCTION

This small document is intended for describing the functionality of the NORS Symbian server. This native OS software serves as a source for lower level information to be retrieved and delivered to appropriate NORS Java handlers. See also the NORS documentation for further information.

## 2. STRUCTURE OF THE SYMBIAN SERVER

The structure of the Symbian server is rather simple. It implements a single TCP server listening to port 4040. Upon connection establishment, the server receives commands over this connection (see Section 2) and replies with appropriate answers after executing the appropriate native functionality.

Each reply is constructed as a sequence of single-lined string, each line finalized with a '\n' and the reply finalized with a '\0'. The reply is being parsed accordingly in the Java part. The different parameters of each single-line reply are separated with a ','.

Each request-reply sequence is synchronous, i.e., it is serialized in the Java part.

With this, the server should support multiple Java clients (one per connection), each with the full command set.

## 3. COMMANDS

Commands are sent as byte codes, i.e., a single byte is received. The reply depends on the command. The following commands are supported, each with its individual reply:

Command	Reply	Description
0x01	cellID,areaID,signal,MCC,MNC,networkName\n\0	Reads cell information according to the appropriate native Symbian API
0x02	MAC,SSID,RSS\n MAC,SSID,RSS\n ... MAC,SSID,RSS\n\0	Reads the WLAN information according to the appropriate native Symbian API (available in S603.0 and above)
0x03	IMEI\n\0	Reads the mobile device's IMEI information according to the appropriate native Symbian API

## 4. READING IN JAVA

NORS provides the SymbianServer class as a helper class for sensor handlers to communicate with the Symbian server. The following snippet shows the usage of the helper class with the example for WLAN scans:

```

StringBuffer data = null;
String[] parts = new String[6];
String reading = null;
byte[] WLANreadings = null;

try
{
    // send command to get data
    SymbianServer.command(SymbianServer.WLANID);

    // initialize reading with Sensor symbol
    reading = new String(sensor);

    do
    {
        // receive reply as StringBuffer
        data = SymbianServer.reply();

        if (data != null)
        {
            // now separate the information
            parts = SymbianServer.parseData(data);
            // reply structure: MAC,SSID,RSSI
            switch(sensor.charAt(1))
            {
                case 'M':
                    reading = reading.concat(parts[0]).concat("\n");
                    break;
                case 'I':
                    if (parts.length>=1)
                        reading = reading.concat(parts[1]).concat("\n");
                    break;
                case 'S':
                    if (parts.length>=2)
                        reading = reading.concat(parts[2]).concat("\n");
                    break;
            }
        }
    }while (data != null); // read as long as there is data

    // now get return value!
    WLANreadings = reading.getBytes();
}
catch (Exception e)
{
    debug("LocationHandler::WLANReading: Cannot receive from Symbian
server");
}

```

A single-line reply of the SymbianServer is delivered in a StringBuffer class with `null` indicating the end of the reply. The `parseData` method of the helper class is used to parse a single parse into the different parameters, delivered as the return array (`parts`). Each of the parameters can then be used accordingly (in this case for the WLAN scanning).