Onewire API

Confidential, Draft
RFC-0037

36 Pages

Abstract

This RFC contains a proposed specification for an onewire API. The purpose of the document is to develop standardized API, which allows OSGi services to interact with onewire devices.

0 Document Information

0.1 Table of Contents

All Page Within This Box

All Page Within This Box

All Page Within This Box

0.2 Status

This document specifies API's for the integration of onewire devices to the OSGi Framework for the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

0.3 Acknowledgement

I want to thank to technical writers team in ProSyst that helping me to write this RFC.

0.4 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [1].

`Source code is shown in this typeface.`

0.5 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	Nov 09 2001	Pavlin Dobrev, ProSyst Software AG p.dobrev@prosyst.com Dobrin Ivanov, ProSyst Software AG d_ivanov@prosyst.bg

1 Introduction

Sensors and modules are being connected to computers, often PCs, in increasing numbers to answer the demand for more communication links and instrumentation. Information typically flows between computer and sensor or module over multiconductor cable consisting of separate communication and power conductors such as the 2-wire I2C bus. Alternatively, an external source of power can be supplied along the cable as needed. A

clever circuit technique that “steals” power from the data line allows a versatile new communication system to transfer bidirectional data while supplying power to remote devices over a single conductor. This is the onewire bus (1-wire MicroLan). This technique can eliminate the cost of extra wires or remote power supplies. Significantly, each device contains a unique individual identification code and a self-timing controller.

2 Motivation and Rationale

A standardized onewire API is needed to provide to OSGi services to interact with onewire devices, such as digital temperature and humidity sensors, NVRAM memory, EPROM memory, EEPROM memory, cryptographic devices, digital switches, A/D converters, real-time-clocks, digital potentiometers.

3 Technical Discussion

3.1 OnewireBus

OnewireBus implementations are base drivers in the framework – so they are scanning the bus for devices and the set of found devices are presented in the Framework like Device services implementing `org.osgi.services.onewire.OnewireDevice` interface, and they also registers a OnewireBus device service implementing `org.osgi.services.onewire.OnewireBus` interface - this device can be used by refining drivers to present refined onewire devices.

3.2 OnewireDevice

`OnewireDevice` implementations are representation of onewire devices into the Framework.

3.3 OnewireAddress

`OnewireAddress` is identification for an iButton device. These objects represent the 64-bit lasered ROM unique serial number of an iButton or Onewire device.

3.4 OnewireCRC16

`OnewireCRC16` is an implementation of Cyclic-Redundency-Check for the iButtons.

All Page Within This Box

3.5 OnewireCRC8

`OnewireCRC8` is an implementation of Cyclic-Redundency-Check for the iButtons.

3.6 OnewireException

`OnewireException` is thrown to signal that an exception on the onewire bus has occurred

3.7 DS1920

The DS1920 Temperature iButton provides 9– bit temperature readings, which indicate the temperature of the device. Information is sent to/from the DS1920 over a 1– Wire interface. Power for reading, writing, and per-forming temperature conversions is derived from the data line itself. Because each DS1920 contains a unique silicon serial number, multiple DS1920s can exist on the same 1– Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environ-mental controls, sensing temperatures inside buildings, equipment or machinery, and in process monitoring and control.

3.8 DS2406

The DS2406 Dual Addressable Switch Plus Memory offers a simple way to remotely control a pair of open drain transistors and to monitor the logic level at each transistor's output via the Onewire bus for closed loop control. Each DS2406 has its own 64-bit ROM registration number that is factory lasered into the chip to provide a guaranteed unique identity for absolute traceability. The device's 1024 bits of EPROM can be used as electronic label to store information such as switch function, physical location, and installation date. Communication with the DS2406 follows the standard Dallas Semiconductor Onewire protocol and can be accomplished with minimal hardware such as a single port pin of a microcontroller. Multiple DS2406 devices can reside on a common Onewire network and be operated independently of each other.

3.9 DS2423

The DS2423 1 – Wire TM RAM With Counters is a fully static, read/write memory for battery operation. The memory is organized as sixteen pages of 256 bits each. In addition, the device has four counters, two of them with external trigger inputs called A and B. Each of the counters is associated with a memory page. A counter without external trigger input increments each time data is written to the page it is associated with (Write Cycle Counter). The counters triggered by inputs A and B, respectively, increment with every low – going pulse on their input. All counters are read – only. They are automatically cleared to zero when the battery is connected.

3.10 DS2438

The DS2438 Smart Battery Monitor provides several functions that are desirable to carry in a battery pack: a means of tagging a battery pack with a unique serial number, a direct-to-digital temperature sensor which eliminates the need for thermistors in the battery pack, an A/D converter which measures the battery voltage and current, an integrated current accumulator which keeps a running total of all current going into and out of the battery, an elapsed time meter, and 40 bytes of nonvolatile EEPROM memory for storage of important parameters such as battery chemistry, battery capacity, charging methodology and assembly date. Information is sent to/from the DS2438 over a Onewire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS2438. This means that battery packs need only have three output connectors: battery power, ground, and the Onewire interface.

4 API

4.1 org.osgi.services.onewire
Interface OnewireBus

public interface **OnewireBus**
A representation of Onewire Bus.

Field Summary

static int	**ALARM_PRESENCE** Onewire bus reset result = alarm presence
static byte	**ALARM_SEARCH_ROM_CODE** The iButton command code to start an alarm search.
static byte	**MATCH_ROM_CODE** The iButton command code to match a single device.
static int	**NO_PRESENCE** Onewire bus reset result = no presence
static int	**NORMAL_PRESENCE** Onewire bus reset result = normal presence
static byte	**READ_ROM_CODE** The iButton command code to read the ID of a single device bus.
static byte	**SEARCH_ROM_CODE** The iButton command code to start a search.
static byte	**SKIP_ROM_CODE** The iButton command code to skip addressing any devices.

Method Summary

void	**armAutoPullUp**(float duration) Arm automatic pull-up to 5V.
void	**beginExclusive**() Gets exclusive use of the Onewire to communicate with an iButton or Onewire Device.
boolean	**beginExclusiveNonBlocking**() Gets exclusive use of the bus to communicate with an iButton.
void	**close**() Closes this onewire bus.
void	**endExclusive**() Relinquishes exclusive control of the onewire bus.

All Page Within This Box

OnewireDevice	**getDevice**(byte[] idBytes) This method returns OnewireDevice object coresponding to its device ID.
OnewireDevice[]	**getDevices**() This method returns all onewire devices presented on the bus, found during the last scan on it.
boolean	**isPresent**(byte[] idBytes) Checks if device with givven ROM ID is presented on the onewire bus.
void	**pullUp**(float duration) Issue a strong pull-up to 5V.
void	**pulse**(float duration) Issue a programming pulse.
boolean	**readBit**() Reads a bit from the onewire bus.
int	**readByte**() Reads a byte from the onewire bus.
void	**readBytes**(byte[] b, int off, int len, boolean fully) Reads up to len bytes of data from the onewire bus into an array of bytes.
int	**resetBus**() Sends a Reset to the onewire bus.
OnewireDevice[]	**scan**(boolean alarm) Scan for onewire devices.
void	**select**(byte[] idBytes) Select (match) the given device.
void	**skip**() Send the skip command.
boolean	**writeBit**(boolean bitValue) Sends a bit to the onewire bus.
void	**writeByte**(int byteValue) Sends a byte to the onewire bus.
void	**writeBytes**(byte[] b, int off, int len) Writes len bytes from the specified byte array starting at offset off to the onewire bus.

Field Detail

4.1.1 SEARCH_ROM_CODE

`public static final byte` **`SEARCH_ROM_CODE`**

The iButton command code to start a search.

4.1.2 ALARM_SEARCH_ROM_CODE

`public static final byte` **`ALARM_SEARCH_ROM_CODE`**

The iButton command code to start an alarm search.

All Page Within This Box

4.1.3 MATCH_ROM_CODE

`public static final byte` **`MATCH_ROM_CODE`**

The iButton command code to match a single device.

4.1.4 READ_ROM_CODE

`public static final byte` **`READ_ROM_CODE`**

The iButton command code to read the ID of a single device bus.

4.1.5 SKIP_ROM_CODE

`public static final byte` **`SKIP_ROM_CODE`**

The iButton command code to skip addressing any devices.

4.1.6 NO_PRESENCE

`public static final int` **`NO_PRESENCE`**

Onewire bus reset result = no presence

4.1.7 NORMAL_PRESENCE

`public static final int` **`NORMAL_PRESENCE`**

Onewire bus reset result = normal presence

4.1.8 ALARM_PRESENCE

`public static final int` **`ALARM_PRESENCE`**

Onewire bus reset result = alarm presence

Method Detail

4.1.9 scan

`public OnewireDevice[] scan(boolean alarm)`
`throws java.io.IOException`

Scan for onewire devices. This method starts a scan on the bus and returns an array of OnewireDevice objects presented on the bus. This method uses the search ROM protocol if alarm is false, and the alarm search ROM protocol otherwise. If alarm search is performed it will return only alarming devices. Most onewire devices only respond to the alarm search if some condition is true, such as a temperature exceeding a certain range on a thermometer ds1820 or ds1920. Some devices never respond to an alarm search.

Parameters:

`alarm` - if true the alarm search ROM protocol is used, else search ROM protocol is used.

Returns:

an array of OnewireDevice objects.

Throws:

java.io.IOException - if an I/O error occurs.

All Page Within This Box

4.1.10 select

```
public void select(byte[] idBytes)
            throws java.io.IOException
```

Select (match) the given device. The bus checks to see if the given device is already selected, and if so, does nothing. Otherwise, it uses the match ROM protocol to select the device, which makes it ready for commands.

Parameters:
`idBytes` - The device ROM ID.

Throws:
java.io.IOException - If an I/O error occurs while contacting the device.

4.1.11 getDevice

```
public OnewireDevice getDevice(byte[] idBytes)
                        throws java.io.IOException
```

This method returns OnewireDevice object coresponding to its device ID.

Parameters:
`idBytes` - the onewire device ROM ID

Returns:
OnewireDevice object or null if device is not presented on the bus.

Throws:
java.io.IOException - If an I/O error occurs.

4.1.12 getDevices

```
public OnewireDevice[] getDevices()
```

This method returns all onewire devices presented on the bus, found during the last scan on it.

Returns:
array of onewire devices

4.1.13 isPresent

```
public boolean isPresent(byte[] idBytes)
                  throws java.io.IOException
```

Checks if device with givven ROM ID is presented on the onewire bus.

Parameters:
`idBytes` - the device ROM ID

Returns:
true if device is presented, else returns false

4.1.14 close

```
public void close()
            throws OnewireException
```

Closes this onewire bus.

Throws:
OnewireException - if exclusive control of the bus is not relinquished

---

All Page Within This Box

4.1.15 resetBus

```
public int resetBus()
            throws java.io.IOException
```

Sends a Reset to the onewire bus.

Returns:

the result of the reset. Potential results are:

- NO_PRESENCE no devices present on the onewire bus.
- NORMAL_PRESENCE normal presence pulse detected on the onewire bus indicating there is a device present.
- ALARM_PRESENCE alarming presence pulse detected on the onewire bus indicating there is a device present and it is in the alarm condition.

Throws:

java.io.IOException - If an I/O error occurs.

4.1.16 writeBit

```
public boolean writeBit(boolean bitValue)
                throws java.io.IOException
```

Sends a bit to the onewire bus.

Parameters:

`bitValue` - the bit value to send to the onewire bus.

Throws:

java.io.IOException - If an I/O error occurs.

4.1.17 readBit

```
public boolean readBit()
                throws java.io.IOException
```

Reads a bit from the onewire bus.

Returns:

the bit value recieved from the onewire bus.

Throws:

java.io.IOException - If an I/O error occurs.

4.1.18 writeByte

```
public void writeByte(int byteValue)
              throws java.io.IOException
```

Sends a byte to the onewire bus.

Parameters:

`byteValue` - the byte value to send to the onewire bus.

Throws:

java.io.IOException - If an I/O error occurs.

4.1.19 writeBytes

```
public void writeBytes(byte[] b,
```

All Page Within This Box

```
                    int off,
                    int len)
             throws java.io.IOException
```

Writes len bytes from the specified byte array starting at offset off to the onewire bus.

Parameters:

b - the data.

`off` - the start offset in the data.

`len` - the number of bytes to write.

Throws:

java.io.IOException -

4.1.20 readByte

```
public int readByte()
            throws java.io.IOException
```

Reads a byte from the onewire bus.

Returns:

the byte value received from the onewire bus.

Throws:

java.io.IOException - If an I/O error occurs.

4.1.21 readBytes

```
public void readBytes(byte[] b,
                      int off,
                      int len,
                      boolean fully)
               throws java.io.IOException
```

Reads up to len bytes of data from the onewire bus into an array of bytes. This method blocks until some input is available.

Parameters:

b - the buffer into which the data is read.

`off` - the start offset of the data.

`len` - the maximum number of bytes read.

`len` - if true tries to read len bytes of data

Throws:

java.io.IOException - If an I/O error occurs.

4.1.22 skip

```
public void skip()
         throws java.io.IOException
```

Send the skip command. This method resets the bus, then transmits the skip ROM command on the bus. All devices on the bus respond as if they saw a match command with their own device IDs.

4.1.23 pulse

```
public void pulse(float duration)
          throws java.io.IOException
```

All Page Within This Box

Issue a programming pulse. Issue a 12V programming pulse on the bus. **Warning!** This may damage your busmaster or other iButtons! Use only with a single 12V-compatible iButton on the bus, such as the Crypto iButton.

Parameters:

`duration` - How long, in microseconds, to pulse the bus.

Throws:

java.io.IOException - If the 12V pulse can't be initiated.

4.1.24 pullUp

```
public void pullUp(float duration)
            throws java.io.IOException
```

Issue a strong pull-up to 5V. Strongly pull up the bus to 5V.

Parameters:

`duration` - How long, in milliseconds, to pulse the bus.

Throws:

java.io.IOException - If the strong pull-up to 5V can't be initiated.

4.1.25 armAutoPullUp

```
public void armAutoPullUp(float duration)
                    throws java.io.IOException
```

Arm automatic pull-up to 5V. With a positive *duration*, arm an automatic pull-up to 5V after every bit or byte transmitted on the bus. With a zero *duration*, disarm the automatic pull-up. Other operations, like a `#scan()` or `#alarmScan()`, will also disarm the automatic pullup.

Parameters:

`duration` - How long, in milliseconds to activate the strong pull-up after every bit or byte; zero disarms the pullup.

Throws:

java.io.IOException - If the automatic pull-up can't be armed or disarmed.

4.1.26 beginExclusive

```
public void beginExclusive()
                    throws OnewireException
```

Gets exclusive use of the Onewire to communicate with an iButton or Onewire Device. This method should be used for critical sections of code where a sequence of commands must not be interrupted by communication of threads with other iButtons, and it is permissible to sustain a delay in the special case that another thread has already been granted exclusive access and this access has not yet been relinquished.

Throws:

OnewireException - if an I/O error occurs or bus closed

4.1.27 beginExclusiveNonBlocking

```
public boolean beginExclusiveNonBlocking()
```

Gets exclusive use of the bus to communicate with an iButton. This method should be used for critical sections of code where a sequence of commands must not be interrupted by communication of threads with other iButtons, and it is not permissible to sustain a delay in the special case that another thread has already been granted exclusive access and this access has not yet been relinquished.

All Page Within This Box

Returns:
'true' if exclusive access to the port was aquired else 'false'
Throws:
OnewireException -

4.1.28 endExclusive

`public void` **`endExclusive`**`()`

Relinquishes exclusive control of the onewire bus. This command dynamically marks the end of a critical section and should be used when exclusive control is no longer needed.

4.2 org.osgi.services.onewire
Interface OnewireDevice

public interface **OnewireDevice**
A representation of Onewire device.

Field Summary

`static java.lang.String`	**`FAMILY`** This is the name of the service property which gives information about the family of Onewire device.
`static int`	**`MATCH_FAMILY`** The driver matches the hardware device down to the exact Onewire family code.
`static int`	**`MATCH_GENERIC`** The driver is for Onewire devices in general..
`static int`	**`MATCH_SERIAL`** The driver matches the hardware device down to the exact serial number.
`static java.lang.String`	**`SERIAL`** This is the name of the service property which gives information about the serial number of Onewire device.

Method Summary

OnewireAddress	**`getAddress`**`()` Returns the onewire address of this device.
OnewireBus	**`getBus`**`()` Returns the onewire bus, to which this device belongs.
`void`	**`select`**`()` Select (match) this device.

Field Detail

4.2.1 MATCH_SERIAL

All Page Within This Box

`public static final int` **`MATCH_SERIAL`**
The driver matches the hardware device down to the exact serial number.

4.2.2 MATCH_FAMILY

`public static final int` **`MATCH_FAMILY`**
The driver matches the hardware device down to the exact Onewire family code.

4.2.3 MATCH_GENERIC

`public static final int` **`MATCH_GENERIC`**
The driver is for Onewire devices in general..

4.2.4 FAMILY

`public static final java.lang.String` **`FAMILY`**
This is the name of the service property which gives information about the family of Onewire device.

4.2.5 SERIAL

`public static final java.lang.String` **`SERIAL`**
This is the name of the service property which gives information about the serial number of Onewire device.

Method Detail

4.2.6 getAddress

`public` `OnewireAddress` **`getAddress`**`()`
Returns the onewire address of this device.
Returns:
the address of this onewire device.

4.2.7 getBus

`public` `OnewireBus` **`getBus`**`()`
Returns the onewire bus, to which this device belongs.
Returns:
the bus, to which this device belongs

4.2.8 select

`public void` **`select`**`()`
`throws java.io.IOException`
Select (match) this device. The bus checks to see if the given device is already selected, and if so, does nothing. Otherwise, it uses the match ROM protocol to select the device, which makes it ready for commands.
Throws:
java.io.IOException - If an I/O error occurs while contacting the device.

All Page Within This Box

4.3 org.osgi.services.onewire
Class OnewireAddress

```
java.lang.Object
  |
  +-org.osgi.services.onewire.OnewireAddress
```

public class **OnewireAddress**
extends java.lang.Object
An identification for an iButton device. These objects represent the 64-bit lasered ROM unique serial number of an iButton or Onewire device.

Field Summary	
static byte	**ADD_ONLY_16K_FAMILY_CODE** The family code for a 16Kbit Add-Only iButton (DS1985 and compatible).
static byte	**ADD_ONLY_1K_FAMILY_CODE** The family code for a 1Kbit Add-Only iButton (DS1982 and compatible).
static byte	**ADD_ONLY_64K_FAMILY_CODE** The family code for a 64Kbit Add-Only iButton (DS1986 and compatible).
static byte	**COUPLER_FAMILY_CODE** The family code for a MicroLAN coupler (DS2409 and compatible).
static byte	**DUAL_SWITCH_FAMILY_CODE** The family code for a dual switch (DS2407, DS2405A and compatible).
static int	**ID_LENGTH** How many bytes comprise a device ROM ID on the iButton bus (8).
static byte	**MEMORY_16K_FAMILY_CODE** The family code for a 16KBit memory iButton (DS1995 and compatible).
static byte	**MEMORY_1K_FAMILY_CODE** The family code for a 1Kbit memory iButton (DS1992 and compatible).
static byte	**MEMORY_4K_FAMILY_CODE** The family code for a 4KBit memory iButton (DS1993 and compatible).
static byte	**MEMORY_64K_FAMILY_CODE** The family code for a 64KBit memory iButton (DS1996 and compatible).
static byte	**MEMORY_PLUS_TIME_FAMILY_CODE** The family code for a memory? iButton (DS1994 and compatible).
static byte	**MULTIKEY_FAMILY_CODE** The family code for a multikey iButton (DS1991 and compatible).
static byte	**SERIAL_NUMBER_FAMILY_CODE** The family code for a silicon serial number device (DS1990A, DS2401 and compatible).
static byte	**SWITCH_FAMILY_CODE** The family code for a switch device (DS2405 and compatible).
static byte	**THERMOMETER_FAMILY_CODE** The family code for a thermometer device (DS1820, DS1920, and compatible).

Constructor Summary
OnewireAddress(byte[] id) Construct a device ID given an array of bytes.

Method Summary	
boolean	**equals**(java.lang.Object obj) Compares this object against the specified object.
byte[]	**getBytes**() Get the bytes that comprise this device ID.
byte	**getCRCCode**() Get the CRC code of the device ID.
byte	**getFamilyCode**() Get the family code of the device ID.
int	**hashCode**()
static boolean	**isValid**(byte[] address) Checks the CRC8 calculation of this Onewire Network address.
long	**serialAsLong**()
java.lang.String	**serialAsString**()
static long	**serialToLong**(byte[] address) Gets device serial number from Onewire Network Address and converts it to a long (little endian).
static java.lang.String	**serialToString**(byte[] id)
static byte[]	**toByteArray**(long address) Convert an iButton or Onewire device address as a long (little endian) into an array of bytes.
long	**toLong**() Return a long representation of this device ID.
static long	**toLong**(byte[] address) Converts a Onewire Network Address to a long (little endian).
java.lang.String	**toString**() Return a string representation of this device ID.

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

4.3.1 SWITCH_FAMILY_CODE

public static final byte **SWITCH_FAMILY_CODE**
The family code for a switch device (DS2405 and compatible).

All Page Within This Box

4.3.2 THERMOMETER_FAMILY_CODE

`public static final byte` **`THERMOMETER_FAMILY_CODE`**
The family code for a thermometer device (DS1820, DS1920, and compatible).

4.3.3 ADD_ONLY_1K_FAMILY_CODE

`public static final byte` **`ADD_ONLY_1K_FAMILY_CODE`**
The family code for a 1Kbit Add-Only iButton (DS1982 and compatible).

4.3.4 ADD_ONLY_16K_FAMILY_CODE

`public static final byte` **`ADD_ONLY_16K_FAMILY_CODE`**
The family code for a 16Kbit Add-Only iButton (DS1985 and compatible).

4.3.5 ADD_ONLY_64K_FAMILY_CODE

`public static final byte` **`ADD_ONLY_64K_FAMILY_CODE`**
The family code for a 64Kbit Add-Only iButton (DS1986 and compatible).

4.3.6 SERIAL_NUMBER_FAMILY_CODE

`public static final byte` **`SERIAL_NUMBER_FAMILY_CODE`**
The family code for a silicon serial number device (DS1990A, DS2401 and compatible).

4.3.7 DUAL_SWITCH_FAMILY_CODE

`public static final byte` **`DUAL_SWITCH_FAMILY_CODE`**
The family code for a dual switch (DS2407, DS2405A and compatible).

4.3.8 COUPLER_FAMILY_CODE

`public static final byte` **`COUPLER_FAMILY_CODE`**
The family code for a MicroLAN coupler (DS2409 and compatible).

4.3.9 MULTIKEY_FAMILY_CODE

`public static final byte` **`MULTIKEY_FAMILY_CODE`**
The family code for a multikey iButton (DS1991 and compatible).

4.3.10 MEMORY_1K_FAMILY_CODE

`public static final byte` **`MEMORY_1K_FAMILY_CODE`**
The family code for a 1Kbit memory iButton (DS1992 and compatible).

All Page Within This Box

4.3.11 MEMORY_4K_FAMILY_CODE

`public static final byte` **`MEMORY_4K_FAMILY_CODE`**

The family code for a 4KBit memory iButton (DS1993 and compatible).

4.3.12 MEMORY_PLUS_TIME_FAMILY_CODE

`public static final byte` **`MEMORY_PLUS_TIME_FAMILY_CODE`**

The family code for a memory? iButton (DS1994 and compatible).

4.3.13 MEMORY_16K_FAMILY_CODE

`public static final byte` **`MEMORY_16K_FAMILY_CODE`**

The family code for a 16KBit memory iButton (DS1995 and compatible).

4.3.14 MEMORY_64K_FAMILY_CODE

`public static final byte` **`MEMORY_64K_FAMILY_CODE`**

The family code for a 64KBit memory iButton (DS1996 and compatible).

4.3.15 ID_LENGTH

`public static final int` **`ID_LENGTH`**

How many bytes comprise a device ROM ID on the iButton bus (8).

Constructor Detail

4.3.16 OnewireAddress

`public` **`OnewireAddress`**`(byte[] id)`
`throws OnewireException`

Construct a device ID given an array of bytes.

Parameters:
`id` - array of ROM ID bytes.

Throws:
OnewireException - if address is invalid

Method Detail

4.3.17 equals

`public boolean` **`equals`**`(java.lang.Object obj)`

Compares this object against the specified object.

Overrides:
equals in class java.lang.Object

Parameters:
`obj` - the object to compare with.

Returns:
true if the objects are representing the same onewire address; false otherwise.

All Page Within This Box

4.3.18 isValid

`public static boolean` **`isValid`**`(byte[] address)`

Checks the CRC8 calculation of this Onewire Network address.

The address is valid if the CRC8 of the first seven bytes of the address gives a result equal to the eighth byte.

Parameters:
`address` - iButton or Onewire Network address to verify

Returns:
`true` if the family code is non-zero and the CRC8 calculation is correct.

See Also:
`com.dalsemi.onewire.utils.CRC8`

4.3.19 hashCode

`public int` **`hashCode`**`()`

Overrides:
hashCode in class java.lang.Object

4.3.20 getFamilyCode

`public byte` **`getFamilyCode`**`()`

Get the family code of the device ID.

Returns:
The family code part of the device ID.

4.3.21 getCRCCode

`public byte` **`getCRCCode`**`()`

Get the CRC code of the device ID.

Returns:
The CRC code.

4.3.22 getBytes

`public byte[]` **`getBytes`**`()`

Get the bytes that comprise this device ID.

Returns:
The bytes that comprise this device ID.

4.3.23 toLong

`public long` **`toLong`**`()`

Return a long representation of this device ID.

Returns:
A long representation of this device ID.

4.3.24 toString

`public java.lang.String` **`toString`**`()`

All Page Within This Box

Return a string representation of this device ID. The format of the string is each byte in hexadecimal, starting with the family code byte, followed by bytes 5 through 0 of the serial number, and ending with the CRC code, separated by "." charaters.
Overrides:
toString in class java.lang.Object
Returns:
A string representation of this device ID.

4.3.25 serialAsString

```
public java.lang.String serialAsString()
```

4.3.26 serialToString

```
public static java.lang.String serialToString(byte[] id)
```

4.3.27 toByteArray

```
public static byte[] toByteArray(long address)
```
Convert an iButton or Onewire device address as a long (little endian) into an array of bytes.

4.3.28 toLong

```
public static long toLong(byte[] address)
```
Converts a Onewire Network Address to a long (little endian).

4.3.29 serialToLong

```
public static long serialToLong(byte[] address)
```
Gets device serial number from Onewire Network Address and converts it to a long (little endian).

4.3.30 serialAsLong

```
public long serialAsLong()
```

4.4 org.osgi.services.onewire
Class OnewireCRC16

```
java.lang.Object
  |
  +--org.osgi.services.onewire.OnewireCRC16
```

public class **OnewireCRC16**
extends java.lang.Object
CRC16 is an implementation of Cyclic-Redundency-Check for the iButtons.

All Page Within This Box

Constructor Summary

`OnewireCRC16()`

Method Summary

`static int`	`compute(byte[] data, int off, int len, int seed)` Perform the CRC16 on an array of data elements based on the provided seed.
`int`	`compute(int data)` Perform the CRC16 on the data element.
`static int`	`compute(int data, int seed)` Perform the CRC16 on the data element based on the provided seed.
`int`	`getCRC()` Retrieve the current CRC value
`void`	`setCRC(int crc)` Sets the CRC value.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

4.4.1 OnewireCRC16

`public OnewireCRC16()`

Method Detail

4.4.2 getCRC

`public int getCRC()`

Retrieve the current CRC value

Returns:

the current CRC

4.4.3 setCRC

`public void setCRC(int crc)`

Sets the CRC value.

Parameters:

`crc` - value to set

4.4.4 compute

`public static int compute(int data,`

All Page Within This Box

```
                        int seed)
```

Perform the CRC16 on the data element based on the provided seed.
CRC16 is based on the polynomial = X^16 + X^15 + X^2 + 1.

Parameters:
`data` - data element on which to perform the CRC16

Returns:
CRC16 value

4.4.5 compute

```
public int compute(int data)
```

Perform the CRC16 on the data element.

Parameters:
`data` - data element on which to perform the CRC16

Returns:
CRC16 value

4.4.6 compute

```
public static int compute(byte[] data,
                          int off,
                          int len,
                          int seed)
```

Perform the CRC16 on an array of data elements based on the provided seed.
CRC16 is based on the polynomial = X^16 + X^15 + X^2 + 1.

Parameters:
`data` - array of data elements on which to perform the CRC16
`off` - offset into the data array
`len` - length of data to CRC16
`seed` - seed to use for CRC16

Returns:
CRC16 value

4.5 org.osgi.services.onewire
Class OnewireCRC8

```
java.lang.Object
  |
  +-org.osgi.services.onewire.OnewireCRC8
```

---

public class **OnewireCRC8**
extends java.lang.Object
CRC8 is an implementation of Cyclic-Redundency-Check for the iButtons.

Constructor Summary
`OnewireCRC8()`

All Page Within This Box

Method Summary	
int	**computeCRC**(byte[] b) Perform the CRC8 on an array of data elements based on the private crc value.
int	**computeCRC**(byte[] b, int seed) Perform the CRC8 on an array of data elements based on the provided seed.
int	**computeCRC**(byte[] b, int off, int len) Perform the CRC8 on an array of data elements based on the private crc value.
int	**computeCRC**(byte[] b, int off, int len, int seed) Perform the CRC8 on an array of data elements based on the private crc value.
int	**computeCRC**(int b) Perform the CRC8 on the data element based on the private crc8 value.
int	**getCRC**() Retrieve the current CRC value
void	**setCRC**(int crc) Sets the CRC value.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

4.5.1 OnewireCRC8

`public OnewireCRC8()`

Method Detail

4.5.2 setCRC

`public void setCRC(int crc)`

Sets the CRC value.

Parameters:

`crc` - value to set

4.5.3 getCRC

`public int getCRC()`

Retrieve the current CRC value

Returns:

the current CRC

4.5.4 computeCRC

`public int computeCRC(int b)`

Perform the CRC8 on the data element based on the private crc8 value.

Parameters:

`b` - data element to perform crc8 on

All Page Within This Box

4.5.5 computeCRC

```
public int computeCRC(byte[] b)
```

Perform the CRC8 on an array of data elements based on the private crc value.

Parameters:

b - array of data elements to perform crc on

Returns:

4.5.6 computeCRC

```
public int computeCRC(byte[] b,
                      int off,
                      int len)
```

Perform the CRC8 on an array of data elements based on the private crc value.

Parameters:

b - array of data elements to perform crc on

off - the start offset of the data

len - the number of bytes included in CRC computation.

Returns:

4.5.7 computeCRC

```
public int computeCRC(byte[] b,
                      int off,
                      int len,
                      int seed)
```

Perform the CRC8 on an array of data elements based on the private crc value.

Parameters:

b - array of data elements to perform crc on

off - the start offset of the data

len - the number of bytes included in CRC computation.

seed - seed to use for crc

Returns:

4.5.8 computeCRC

```
public int computeCRC(byte[] b,
                      int seed)
```

Perform the CRC8 on an array of data elements based on the provided seed.

Parameters:

b - array of data elements to perform crc on

seed - seed to use for crc

Returns:

CRC value

All Page Within This Box

4.6 org.osgi.services.onewire
Class OnewireException

```
java.lang.Object
  |
  +-java.lang.Throwable
        |
        +-java.lang.Exception
              |
              +-org.osgi.services.onewire.OnewireException
```

public class **OnewireException**
extends java.lang.Exception
OnewireException is thrown to signal that an exception on the onewire bus has occurred.
See Also:
Serialized Form

Constructor Summary
OnewireException()
OnewireException(java.lang.String err)

Method Summary	
java.lang.String	**toString**()

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

4.6.1 OnewireException

public **OnewireException**()

4.6.2 OnewireException

public **OnewireException**(java.lang.String err)

Method Detail

4.6.3 toString

All Page Within This Box

```
public java.lang.String toString()
```

Overrides:
toString in class java.lang.Throwable

4.7 org.osgi.services.onewire
Interface DS1920

public interface **DS1920**
The DS1920 Temperature iButton provides 9– bit temperature readings which indicate the temperature of the device. Information is sent to/from the DS1920 over a 1– Wire interface. Power for reading, writing, and performing temperature conversions is derived from the data line itself. Because each DS1920 contains a unique silicon serial number, multiple DS1920s can exist on the same 1– Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environ-mental controls, sensing temperatures inside buildings, equipment or machinery, and in process monitoring and control.

Method Summary

OnewireAddress	**getAddress**() Returns the onewire address of this ds1920 device.
double	**getTemperature**() Returns the current temperature value.

Method Detail

4.7.1 getTemperature

```
public double getTemperature()
                     throws OnewireException,
                            java.io.IOException
```

Returns the current temperature value.
Returns:
current temperature value
Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.7.2 getAddress

```
public OnewireAddress getAddress()
```

Returns the onewire address of this ds1920 device.
Returns:
the onewire address of this device.

4.8 org.osgi.services.onewire
Interface DS2406

public interface **DS2406**

All Page Within This Box

The DS2406 Dual Addressable Switch Plus Memory offers a simple way to remotely control a pair of open drain transistors and to monitor the logic level at each transistor's output via the Onewire bus for closed loop control. Each DS2406 has its own 64-bit ROM registration number that is factory lasered into the chip to provide a guaranteed unique identity for absolute traceability. The device's 1024 bits of EPROM can be used as electronic label to store information such as switch function, physical location, and installation date. Communication with the DS2406 follows the standard Dallas Semiconductor Onewire protocol and can be accomplished with minimal hardware such as a single port pin of a microcontroller. Multiple DS2406 devices can reside on a common Onewire network and be operated independently of each other.

Method Summary

OnewireAddress	**getAddress**() Returns the onewire address of this ds2406 device.
boolean	**readChannelA**() Returns the status of the channel A.
boolean	**readChannelB**() Returns the status of the channel B.
void	**switchChannelA**(boolean on) Set the status of the channel A.
void	**switchChannelB**(boolean on) Set the status of the channel B.

Method Detail

4.8.1 readChannelA

```
public boolean readChannelA()
                    throws OnewireException,
                           java.io.IOException
```

Returns the status of the channel A.

Returns:

the status of the channel A

Throws:

OnewireException - if an I/O error occurs or bus closed or CRC error

java.io.IOException - if an I/O error occurs

4.8.2 readChannelB

```
public boolean readChannelB()
                    throws OnewireException,
                           java.io.IOException
```

Returns the status of the channel B.

Returns:

the status of the channel B

Throws:

OnewireException - if an I/O error occurs or bus closed or CRC error

java.io.IOException - if an I/O error occurs

All Page Within This Box

4.8.3 switchChannelA

```
public void switchChannelA(boolean on)
                    throws OnewireException,
                           java.io.IOException
```

Set the status of the channel A.

Parameters:
on - value for channel A

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.8.4 switchChannelB

```
public void switchChannelB(boolean on)
                    throws OnewireException,
                           java.io.IOException
```

Set the status of the channel B.

Parameters:
on - value for channel B

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.8.5 getAddress

```
public OnewireAddress getAddress()
```

Returns the onewire address of this ds2406 device.

Returns:
the onewire address of this device.

4.9 org.osgi.services.onewire
Interface DS2423

---

public interface **DS2423**
The DS2423 1– Wire TM RAM With Counters is a fully static, read/write memory for battery operation. The memory is organized as sixteen pages of 256 bits each. In addition, the device has four counters, two of them with external trigger inputs called A and B. Each of the counters is associated with a memory page. A counter without external trigger input increments each time data is written to the page it is associated with (Write Cycle Counter). The counters triggered by inputs A and B, respectively, increment with every low– going pulse on their input. All counters are read– only. They are automatically cleared to zero when the battery is connected.

Method Summary

OnewireAddress	**getAddress**() Returns the onewire address of this ds2423 device.
long	**readCounter**(int counterPage) Returns the current value of the page counter for specified page.

All Page Within This Box

Method Detail

4.9.1 readCounter

```
public long readCounter(int counterPage)
                 throws OnewireException,
                        java.io.IOException
```

Returns the current value of the page counter for specified page.

Parameters:
`counterPage` - the page number

Returns:
page counter value

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.9.2 getAddress

```
public OnewireAddress getAddress()
```

Returns the onewire address of this ds2423 device.

Returns:
the onewire address of this device.

4.10 org.osgi.services.onewire
Interface DS2438

---

public interface **DS2438**
The DS2438 Smart Battery Monitor provides several functions that are desirable to carry in a battery pack: a means of tagging a battery pack with a unique serial number, a direct-to-digital temperature sensor which eliminates the need for thermistors in the battery pack, an A/D converter which measures the battery voltage and current, an integrated current accumulator which keeps a running total of all current going into and out of the battery, an elapsed time meter, and 40 bytes of nonvolatile EEPROM memory for storage of important parameters such as battery chemistry, battery capacity, charging methodology and assembly date. Information is sent to/from the DS2438 over a Onewire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS2438. This means that battery packs need only have three output connectors: battery power, ground, and the Onewire interface.

Method Summary

void	**convertT**() This command begins a temperature conversion.
void	**convertV**() This command instructs the DS2438 to initiate a voltage analog-to-digital conversion cycle.
void	**copySP**(int page) This command copies the scratchpad page x into the EEPROM / SRAM memory page x of the DS2438.

All Page Within This Box

OnewireAddress	**getAddress**() Returns the onewire address of this ds2438 device.
int	**getHumidity**() Returns the current humidity value.
double	**getTemperature**() Returns the current temperature value.
double	**getVoltage**() Returns the current voltage value.
byte[]	**readSP**(int page) This command reads the contents of the scratchpad page on the DS2438.
void	**recallMemory**(int page) This command recalls the stored values in EEPROM / SRAM page x to the scratchpad page x.
void	**writeSP**(int page, byte[] buf, int ofs, int len) This command writes to the scratchpad page of the DS2438.

Method Detail

4.10.1 writeSP

```
public void writeSP(int page,
                    byte[] buf,
                    int ofs,
                    int len)
             throws OnewireException,
                    java.io.IOException
```

This command writes to the scratchpad page of the DS2438. The entire 8-byte scratchpad space may be written, but all writing begins with the byte present at address 0 of the selected scratchpad. Valid page numbers for writing are 0 - 7.

Parameters:

`page` - the page number

`buf` - array of data bytes

`ofs` - offset of the begining in array of data bytes

`len` - the length of bytes to be written in the scratchpad

Throws:

OnewireException - if an I/O error occurs or bus closed or CRC error

java.io.IOException - if an I/O error occurs

4.10.2 readSP

```
public byte[] readSP(int page)
              throws OnewireException,
                     java.io.IOException
```

This command reads the contents of the scratchpad page on the DS2438. The entire 8-byte scratchpad will be read. Valid page numbers are 0 - 7.

Parameters:

`page` - the page number

Returns:

array of page data

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.10.3 copySP

```
public void copySP(int page)
            throws OnewireException,
                   java.io.IOException
```

This command copies the scratchpad page x into the EEPROM / SRAM memory page x of the DS2438. Valid page numbers are 0 - 7.

Parameters:
page - the page number

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.10.4 convertT

```
public void convertT()
              throws OnewireException,
                     java.io.IOException
```

This command begins a temperature conversion. No further data is required.

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.10.5 convertV

```
public void convertV()
              throws OnewireException,
                     java.io.IOException
```

This command instructs the DS2438 to initiate a voltage analog-to-digital conversion cycle. When the A/D conversion is done, the current voltage value is placed in the VOLTAGE REGISTER of page 0. While an A/D conversion is taking place, all other memory functions are still available for use.

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

4.10.6 recallMemory

```
public void recallMemory(int page)
                  throws OnewireException,
                         java.io.IOException
```

This command recalls the stored values in EEPROM / SRAM page x to the scratchpad page x. Valid page numbers are 0 - 7.

Parameters:
page - the page number

Throws:
OnewireException - if an I/O error occurs or bus closed or CRC error
java.io.IOException - if an I/O error occurs

All Page Within This Box

4.10.7 getTemperature

```
public double getTemperature()
                    throws OnewireException,
                           java.io.IOException
```

Returns the current temperature value.

Returns:

current temperature value

Throws:

OnewireException - if an I/O error occurs or bus closed or CRC error

java.io.IOException - if an I/O error occurs

4.10.8 getVoltage

```
public double getVoltage()
                  throws OnewireException,
                         java.io.IOException
```

Returns the current voltage value.

Returns:

the current voltage value

Throws:

OnewireException - if an I/O error occurs or bus closed or CRC error

java.io.IOException - if an I/O error occurs

4.10.9 getHumidity

```
public int getHumidity()
                throws OnewireException,
                       java.io.IOException
```

Returns the current humidity value. Use this method only if the Honeywell HIH-3605 humidity sensor is attached to the ds2438 chip.

Returns:

the current humidity value

Throws:

OnewireException - if an I/O error occurs or bus closed or CRC error

java.io.IOException - if an I/O error occurs

4.10.10 getAddress

```
public OnewireAddress getAddress()
```

Returns the onewire address of this ds2438 device.

Returns:

the onewire address of this device

All Page Within This Box

5 Security Considerations

Security is outside of the scope of this RFC.

6 Document Support

6.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. ProSyst mBedded Server documentation, http://www.mBeddedServer.com

6.2 Author's Address

Name	Pavlin Dobrev
Company	ProSyst Software AG
Address	Gustav-Heinemann-Ufer 54, 50968 Cologne, Germany
Voice	+359 2 963 10 04
e-mail	p.dobrev@prosyst.com

Name	Dobrin Ivanov
Company	ProSyst Software AG
Address	Gustav-Heinemann-Ufer 54, 50968 Cologne, Germany
Voice	+359 2 963 10 04
e-mail	d_ivanov@prosyst.bg

All Page Within This Box

6.3 Acronyms and Abbreviations

OSGi	Open Service Gateway Initiative

6.4 End of Document

All Page Within This Box