



Universe SDK User Guide

Sensurity Ltd

January 24, 2016

For the avoidance of doubt, the use of this User Guide does not convey upon the reader any intellectual property rights of Sensurity Ltd. The contents of this User Guide including any copyright, trademarks and other intellectual property rights of Sensurity Ltd referred to herein are and remain property of Sensurity Ltd.

© 2016 Sensurity Ltd. All rights reserved.

Sensurity Ltd
Suite 6 Davidson House
Glenavy Road Business Park
Moir
Co. Armagh
Northern Ireland
United Kingdom
BT67 0LT

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Sensurity Ltd. Under the law, reproducing includes translating into another language or format.

Contents

1 Introduction.....	6
2 Installation.....	7
2.1 Windows.....	7
2.2 Linux.....	7
3 Service Architecture.....	7
4 Description of Operation.....	9
4.1 SRP PAKE Authentication.....	10
4.2 OSDP Network Installation.....	10
4.3 OSDP Peripheral Device Installation.....	11
5 Command/Response List.....	12
5.1 Authenticate.....	13
5.2 Cryptogram.....	13
5.3 Reject Authentication.....	14
5.4 Open Authentication.....	14
5.5 Get Service Information.....	14
5.6 Get Service Alarms.....	17
5.7 Clear Service Alarms.....	18
5.8 Get Service Events.....	19
5.9 Clear Service Events.....	20
5.10 Get Network Information.....	21
5.11 Add Network.....	22
5.12 Remove Network.....	23
5.13 Modify Network.....	23
5.14 Get Device Information.....	25
5.15 Find Device.....	26
5.16 Install Device.....	27
5.17 Remove Device.....	27
5.18 Upload Firmware Image.....	28
5.19 Delete Firmware Image.....	29
5.20 List Firmware Image.....	29
5.21 Update Device.....	30
5.22 Update Firmware Status.....	31
5.23 Get Firmware Status.....	32
5.24 Change Address.....	33
5.25 Soft Reset.....	33
5.26 Set Security.....	34
5.27 Get Security.....	35
5.28 Set IPv4 Network Configuration.....	36
5.29 Get IPv4.....	37
5.30 Set Microwave Intrusion Parameters.....	37
5.31 Get Microwave Intrusion Parameters.....	39
5.32 Set Relay Parameters.....	39
5.33 Get Relay Parameters.....	41
5.34 Set Deadzones.....	42
5.35 Get Deadzones.....	43

5.36 Get Deadzones Fitted.....	44
5.37 Set Alarms.....	44
5.38 Get Alarms.....	45
5.39 Set ADC.....	46
5.40 Get ADC.....	47
5.41 Set Polling Period.....	48
5.42 Get Polling Period.....	48
5.43 Set Algorithm.....	49
5.44 Get Algorithm.....	50
5.45 Update GPS.....	51
5.46 Get GPS.....	51
5.47 Set Trace Level.....	52
5.48 Get Trace Level.....	53
5.49 Get Trace.....	54
5.50 Set Enable Samples.....	54
5.51 Get Enable Samples.....	55
6 Push List.....	57
6.1 Heartbeat.....	57
6.2 Service Alarms.....	57
6.3 Service Events.....	57
6.4 Sample Data.....	57
7 Examples of Use.....	59
7.1 Secure Remote Password Authentication.....	59
7.2 Installing an OSDP Network.....	59
7.3 Installing an OSDP Peripheral Device.....	59
7.4 Retrieving Device Settings.....	59
7.5 Configuring Device Settings.....	59
7.6 Uploading a Firmware Image to the Service.....	59
7.7 Updating Device Firmware.....	59

1 Introduction

Sensurity Perimeter Intrusion Detection (PID) devices utilize the SIA Open Supervised Device Protocol (OSDP, [1]) to provide binary communications to those devices. Sensurity OSDP compliant devices support OSDP Secure Channel Session (OSDP-SCS) which utilizes 128-bit AES-CMAC. The Sensurity Service is a Windows Service/Linux daemon that operates in the background of a PC and operates an arbitrary number of OSDP Control Panels (CP's). These CP's each maintain control of up to 127 OSDP Peripheral Devices (PD's) with automated polling of alarms and an interface that allows commands to be issued to each PD to control and monitor their operation.

Interaction with the service is achieved through an asynchronous TCP/IP socket that is secured using Secure Socket Layer (SSL) and a username/password. Data in the form of XML strings are exchanged with the service using this socket.

The Universe Service socket interface and the data packets that are exchanged has been chosen to provide the greatest compatibility with a range of programming languages and target operating systems. XML data is readily understood, highly portable and provides a rapid deployment path for integration of Sensurity products into other systems using the Service.

2 Installation

2.1 Windows

A self-contained installer is provided by Sensurity Ltd to install and uninstall the Windows Service. In addition a configuration application is provided that permits a user to start, stop and configure the Sensurity Universe Service.

NOTE: The configuration tool must be started as an 'Administrator' user to enable the service start/stop controls, otherwise the service must be controlled using the services tool within Windows. The service must be restarted in order for any changes to take effect. The configuration tool will automatically restart the service if it has been started with Administrator privileges.

2.2 Linux

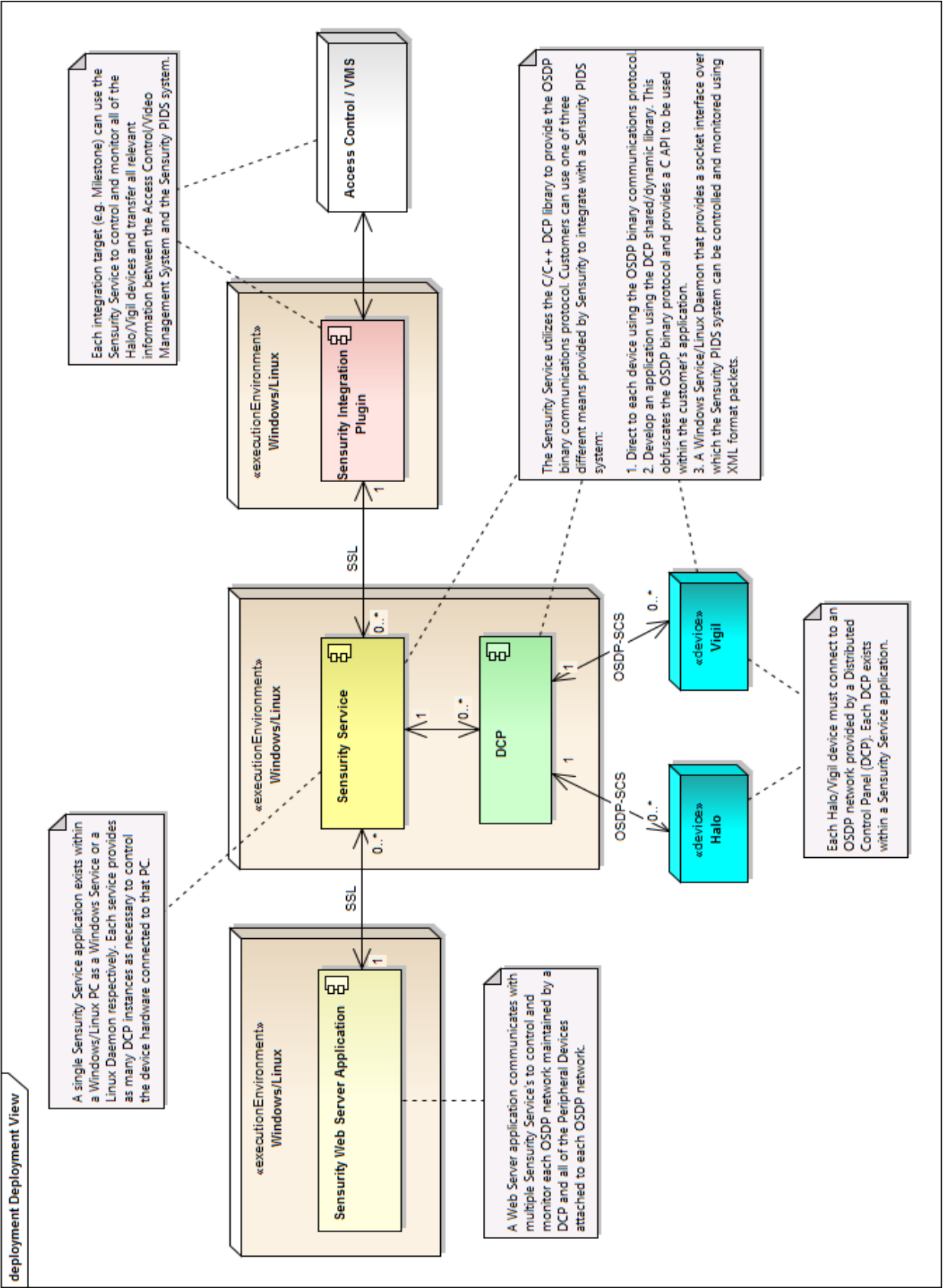
A compressed folder is provided with the installation files. This file should be decompressed and the installer bash script should be executed from the root directory of the resulting folder. This script will install the Universe daemon into the default system directory or the user can specify the installation directory from the prompt in the terminal window. The installation script will also initialize the system with a user name and password for the *admin* user.

Shell scripts are provided in the installation directory that allow the Universe daemon to be manually stopped and started.

3 Service Architecture

All functionality to provide an OSDP Control Panel is contained within the Distributed Control Panel C/C++ library (DCP) developed at Sensurity [2]. This library provides a portable C API for an event-driven C++ application that performs all low level binary communications with each OSDP Peripheral Device (PD). Each DCP maintains a unique XML configuration file that stores details of all installed PDs and the configuration of the OSDP network. It should be noted that all OSDP networks MUST be identified with a unique name during configuration using the Service.

The Universe Service relies upon the DCP to provide the functionality for an arbitrary number of OSDP networks. This library is integrated into a C# application to provide the service executable. This C# application is responsible for initializing the system, starting/stopping instances of the OSDP networks from operating, maintaining a TCP socket server for communications and translating any XML data packets exchanged with a client application in order to operate each OSDP network.



4 Description of Operation

During installation the service is configured with a name and IP port on which to operate. The administration user account must also be created and supplied with a password. It is also necessary during installation to configure the service with the appropriate SSL certificate for the service, and to provide the SSL certificate containing the client application's public key. All configuration information is maintained in an XML file.

All further configuration of the Service is achieved using the TCP/IP socket. It should be noted that the client socket **MUST** be asynchronous in operation – this is to permit events to be “pushed” from the server to any connected client and to reduce the data bandwidth of any communications with the Service.

An authenticated connection with the Service requires the exchange and validation of both client and server X.509 certificates. The user will be supplied with example self-signed X.509 certificates for both the client and server sides. By default these certificates provide 256-bit AES encryption to the Sensurity Universe Service.

Once secure access to the service is achieved (i.e. communications are encrypted using SSL) an authenticated login involving a username and password is required. *This information will be used in future for auditing purposes, e.g. the service will record who and when device settings were modified.* This Service login handshake will close the socket if the username/password are not accepted. Please see Section 4.1 for a full description of the authentication process.

NOTE: The system will install a default username and password of *admin/1234* that can NOT be deleted.

Once a secure authenticated connection with a verified user has been established the client application can transfer XML formatted strings to the Service to instigate command operations on any of the connected OSDP Control Panel's and their connected devices. All commands will be responded to by the Service with a response packet.

Various commands require interaction with Sensurity devices to configure settings or retrieve data. These commands will provide an immediate response packet, however an event will subsequently be generated by the Service when that command has completed. These events are asynchronously transmitted to the client to indicate the outcome of the command.

In addition, alarm events are created by the Service to indicate the occurrence of significant events in the operation of a Sensurity PID System (e.g. device disconnection from a network, intruder alarms). Please refer to Section Push List for a full list of alarm events. These alarm events are also transmitted asynchronously to reduce latency and remove the need for a polling mechanism.

The three basic steps required to create an operating network are to authenticate with the service, install one or more OSDP networks and install Peripheral Devices onto those networks. Once this is achieved the Service is capable of providing alarms to any attached client application.

4.1 SRP PAKE Authentication

SRP (IETF RFC2945) requires the exchange of public keys and cryptograms between client and server. The operation of SRP is beyond the scope of this document, but a C# implementation is provided with the SDK examples.

A typical SRP handshake is as follows:

1. The user application sends the user name and public client key to the server using an *Authenticate* packet (see Section 5.1). Note that the username must be converted to a base64 string.

```
<universe cmd='Authenticate'>
  <authenticate user='67F32<1' clientkey='BA6901FBA53CE014846' />
</universe>
```

2. The Universe Service application will respond with an *Open Authentication* packet (see Section 5.4) if SRP is disabled or a *Reject Authentication* packet (see Section 5.3) if the user name is unknown. Otherwise the Service will respond with a complementary *Authenticate* packet containing the salt and public server key.

```
<universe response='Authenticate'>
  <authenticate salt='78451B9ACE512783524' serverkey='DF195BAC67C7E8163642' />
</universe>
```

3. The user application must generate the relevant client cryptogram for verification by the Service using a *Cryptogram* packet (see Section 5.2).

```
<universe cmd='Cryptogram'>
  <cryptogram client='FBA873178BCE90629AC7C7A484' />
</universe>
```

4. The Service will generate the server cryptogram if the client cryptogram is verified. If the client cryptogram fails verification the Service will respond with a *Reject Authentication* packet.

```
<universe response='Cryptogram'>
  <cryptogram server='6145AB834CE857AB344897256DF' />
</universe>
```

4.2 OSDP Network Installation

The Universe Service must be configured to provide an OSDP network prior to any Peripheral Devices being installed. Typically this is performed by issuing an *Add Network* command (see Section 5.11) and specifying the network configuration including its unique name. Other information provided by this command indicates if RS485 is to be enabled/disabled and the corresponding baud rate.

Each network that is added to the service has its details maintained in the “*universe_service.xml*” file. This file is loaded at start-up allowing the service to automatically start the networks as the

service itself commences operation. Therefore once added, a network will operate until it is modified or deleted using the *Modify Network* or *Remove Network* commands (see Sections 5.13 and 5.12 respectively). A *Modify Network* command will shut down the specified network, reconfigure its settings and start the network again, whilst a *Remove Network* command will stop the network and delete its settings from the “*universe_service.xml*” file.

The name of each network with a service must be unique, as the name is used for identification purposes.

4.3 OSDP Peripheral Device Installation

Each device will default to the OSDP address “0” and IP address 192.168.2.40, therefore conflicts between PD's with the same addresses should be avoided by initially powering devices in isolation and installing them sequentially.

A *Find Device* command (see Section 5.15) will instruct the OSDP Control Panel to search all available OSDP addresses for a response. If a response is received the client will be informed with the appropriate event message, however if the search fails the client will also receive an event indicating failure.

The OSDP address provided by the response of a *Find Device* command can then be used to issue an *Install Device* command (see Section 5.16) to permanently add a device to the OSDP network. Once installed the device will be interrogated by the OSDP Control Panel to determine its configuration settings and alarm polling will subsequently commence. At this point the client application can configure and control the PD.

Once each PD is installed and communications are established the OSDP address should be changed to a non-default address and the static IP address should also be modified from the default factory setting, allowing additional PD's to be installed without conflict. It is recommended that the OSDP address and the base IP address of the network are related somehow, e.g. OSDP address 2, 3 and 4 are configured to use IP addresses 192.168.2.42, 192.168.2.43 and 192.168.2.44.

The Universe Service maintains an XML configuration file detailing the settings of each installed PD, allowing each OSDP network and all attached PD's to start automatically with the system. Once installed and the devices are configured they will retain their settings and do not require further control.

As alarms are generated they are automatically transmitted to a client application which is connected to the Universe Service asynchronous SSL TCP connection.

5 Command/Response List

The XML Schema for the encapsulating XML element of all Universe Service packets is as follows:

```
<xs:complexType name="msgType" abstract="true">
</xs:complexType>

<xs:element name="universe" type="msgType" minOccurs="1" maxOccurs="1">
</xs:element>

<xs:complexType name="commandMsg" >
  <xs:complexContent>
    <xs:extension base=" msgType" >
      <xs:attribute name="cmd" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="responseMsg"
  <xs:complexContent>
    <xs:extension base=" msgType" >
      <xs:attribute name="response" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

This encapsulating XML element is not included in the following XML Schema of all communications protocol packets for reasons of brevity.

All commands and responses **MUST** be enclosed within the XML “universe” element and the closing tag </universe> **MUST** be used:

```
<universe cmd=[Command]>
</universe>
```

All commands will include the *cmd* attribute as shown above, whilst all responses will utilise the *response* attribute:

```
<universe response=[Response]>
</universe>
```

Commands which incur a delayed response from the Universe Service will provide an immediate response in the form of a command acceptance packet that adheres to the following XML Schema:

```
<xs:element name="return">
  <xs:complexType>
```

```

    <xs:attribute name="dcp" type="xs:string" use="required" minOccurs="0" />
    <xs:attribute name="value" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

```

5.1 Authenticate

- Implements the Secure Remote Password protocol.
- Used to initiate an authentication handshake with the service and for the service to respond to that handshake.
- The user name is encoded as a base64 string to avoid conflicts with XML escape codes.
- The client key is a public key encoded as an upper case hex string.
- The salt used to randomise the HMAC associated with each username is encoded as an upper case hex string.
- The server key is a public key represented as an upper case hex string.

XML Schema:

```

<xs:element name="authenticate">
  <xs:attribute name="user" type="xs:base64Binary" use="required"/>
  <xs:attribute name="clientkey" type="xs:string" use="required"/>
</xs:element>

```

Example:

```

<universe cmd='authenticate'>
  <authenticate user='67F32<1' clientkey='0123456789ABCDEF' />
</universe>

```

```

<universe response='authenticate'>
  <authenticate salt='67F32<1' serverkey='FEDCBA9876543210' />
</universe>

```

5.2 Cryptogram

- Implements the Secure Remote Password protocol.
- Used to transfer the client and server cryptograms between the client/server.
- The cryptogram is encoded as an upper case hex string.

Xml Schema:

```

<xs:element name="cryptogram">
  <xs:attribute name="client" type="xs:string" use="required"/>
  <xs:attribute name="server" type="xs:string" use="required"/>
</xs:element>

```

Example:

```

<universe cmd='cryptogram'>

```

```
<cryptogram client='0123456789ABCDEF' />
</universe>
```

```
<universe response='cryptogram'>
  <cryptogram server='FEDCBA9876543210' />
</universe>
```

5.3 Reject Authentication

- Used in the event of an authentication failure on the server side to indicate that the SRP PAKE handshake has failed.

Example:

```
<universe cmd='reject_auth'>
</universe>
```

- Used in the event of an authentication handshake attempt by a client in which the Universe Service has been configured with SRP PAKE disabled. This message indicates to the client side that an authentication handshake is unnecessary.

5.4 Open Authentication

Example:

```
<universe cmd='open_auth'>
</universe>
```

5.5 Get Service Information

- Used to retrieve information regarding the service machine itself and the OSDP networks associated with it.
 - The status of the Universe Service is identified with the following possible values:
 - “Setup” - System is initializing
 - “Running” - System has started and all OSDP CP's have commenced operation
 - The PC on which the service operates is identified according to its name.
 - The operating system is identified according to the following possible strings:
 - “windows”
 - “linux”
 - The OS version is detailed more specifically in the OS version string.
 - Used to retrieve the configuration and status of each Distributed Control Panel being maintained by the service.
 - The unique name of each OSDP CP instance is encoded as a string, as set by a user during network configuration.
 - The number of installed devices on the specified network is provided as an integer.
 - The status of each network is encoded as a string with the following possible values:
-

- “Connected”
 - “Disconnected”
- The master port is the TCP port to be used internally by the service to issue commands and receive replies from connected devices. This port is protected by mutual authentication using randomly generated public/private key pairs within an automated PAKE system that is robust to Man-in-the-middle (MITM) attacks. The port is encoded as an integer.
- The slave port is the TCP port to be used by all external devices to connect to the system. This port can only be used to receive replies on an OSDP network. This port is encoded as an integer.
- The delay between successive attempts to connect to a disconnected PD is described using the reconnection delay. This delay is described in *milliseconds* and is encoded as an integer.
- Support for TCP only communication is indicated using a boolean flag, i.e. this flag indicates if RS485 support on a network is enabled.
- The RS485 hardware interface is described using a string. The content of this string is OS dependent as per the following examples:
 - “COM3” - Windows
 - “/dev/ttyUSB0” - Linux
- The RS485 baud rate to be used by a network is encoded as an integer with an arbitrary value.

Xml Schema:

```

<xs:complexType name="commsType" abstract="true">
</xs:complexType>

<xs:complexType name="IpOnlyDcpType">
  <xs:complexContent>
    <xs:extension base="commsType">
      <xs:sequence>
        <xs:element name="tcp_only" type="xs:boolean" value="true" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="Rs485DcpType">
  <xs:complexContent>
    <xs:extension base="commsType">
      <xs:sequence>
        <xs:element name="tcp_only" type="xs:boolean" value="false" />
        <xs:element name="rs485_device" type="xs:string"/>
        <xs:element name="rs485_baud" type="xs:integer"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="service">
  <xs:attribute name="status" type="xs:string" use="required"/>
  <xs:attribute name="server" type="xs:string" use="required"/>

```

```

<xs:attribute name="os" type="xs:string" use="required"/>
<xs:attribute name="os_version" type="xs:string" use="required"/>
<xs:attribute name="srp" type="xs:string" use="required"/>
<xs:complexType>
  <xs:element name="dcp" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="num_pd" type="xs:integer"/>
        <xs:element name="status" type="xs:string"/>
        <xs:element name="master_port" type="xs:integer"/>
        <xs:element name="slave_port" type="xs:integer"/>
        <xs:element name="reconnection_delay" type="xs:integer"/>
        <xs:element name="name" type="commsType" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:complexType>
</xs:element>

```

Example 1:

```

<universe cmd='get_service_info'>
</universe>

```

```

<universe response='get_service_info'>
  <service status='Disconnected'/>
</universe>

```

Example 2:

```

<universe cmd='get_service_info'>
</universe>

```

```

<universe response='get_service_info'>
  <service status='Connected' server='SITEPC0' os='windows'
    os_version='Wicrosoft Windows NT 6.2.9200.0' srp='1' />
  <dcp>
    <name>Front entrance</name>
    <num_pd>6</num_pd>
    <status>Connected</status>
    <master_port>27043</master_port>
    <slave_port>27042</slave_port>
    <reconnection_delay>15000</reconnection_delay>
    <tcp_only>false</tcp_only>
    <rs485_device>COM5</rs485_device>
    <rs485_baud>115200</rs485_baud>
  </dcp>
  <dcp>
    <name>Perimeter</name>
    <num_pd>14</num_pd>
    <status>Connected</status>
  </dcp>
</universe>

```

```

    <master_port>27045</master_port>
    <slave_port>27044</slave_port>
    <reconnection_delay>15000</reconnection_delay>
    <tcp_only>true</tcp_only>
  </dcp>
</universe>

```

5.6 Get Service Alarms

- Request or receive a list of alarms generated by the service.
- Only alarms applicable to the requesting interface will be returned.
- Service alarms utilise the asynchronous TCP socket connection to transmit alarm messages with low latency, i.e. polling can be used but is not required to receive alarms. Alarms are generated when the OSDP networks raise an alarm associated with intrusion, access control or other activities which are construed as suspicious and warrant the attention of the controlling system.
- If a client is not connected to the service then the alarms are discarded. **NOTE: In future a mechanism may be employed to store these alarms for later distribution, e.g. a database, storage file etc.**
- Each message associated with an alarm event uses the network name and OSDP address of a device to uniquely identify that device. **NOTE: General support for OSDP compliant devices can be assured using these unique identifications.**
- The time associated with the alarm event is recorded using the date and time that the OSDP Control Panel detected the event.
- The type of alarm is encoded as a string according to the following permitted enumerations:
 - DCP_ALARM_HUB_DEVICE_CONNECT – An IP device has connected to a network
 - DCP_ALARM_HUB_DEVICE_DISCONNECT – An IP device has disconnected from a network
 - DCP_ALARM_CONNECT – An OSDP Peripheral Device has successfully connected to an OSDP network and alarm polling has commenced
 - DCP_ALARM_DISCONNECT – An OSDP Peripheral Device has disconnected from a OSDP network
 - DCP_ALARM_MICROWAVE – A microwave intrusion alarm has been triggered
 - DCP_ALARM_UPPER_DEADZONE – An upper deadzone intrusion alarm has been triggered
 - DCP_ALARM_LOWER_DEADZONE – A lower deadzone intrusion alarm has been triggered
 - DCP_ALARM_ENVIRONMENTAL – An environmental alarm has been triggered (e.g. excessive temperature gradient, low/high temperature threshold exceeded)
 - DCP_ALARM_ERROR – A device error alarm has been triggered (e.g. a device error has occurred)
 - DCP_ALARM_POWER – A device power alarm has occurred
 - DCP_ALARM_TAMPER – A device tamper alarm has occurred (e.g. Halo rear panel removed, abrupt shock)
 - DCP_ALARM_ADC_0 – An input device alarm has been triggered on input 0
 - DCP_ALARM_ADC_1 – An input device alarm has been triggered on input 1
 - DCP_ALARM_ADC_2 – An input device alarm has been triggered on input 2
 - DCP_ALARM_ADC_3 – An input device alarm has been triggered on input 3

- DCP_ALARM_HUB_CONNECT – The communications interface of an OSDP network has been initialised
- DCP_ALARM_HUB_DISCONNECT – The communications interface of an OSDP network has been stopped
- DCP_ALARM_UNKNOWN – An unspecified alarm event has occurred

Xml Schema:

```
<xs:element name="alarm" maxOccurs="unbounded" >
  <xs:complexType>
    <xs:sequence>
      <xs:attribute name="dcp" type="xs:string" use="required" />
      <xs:attribute name="pd" type="xs:integer" use="required" />
      <xs:attribute name="time" type="xs:string" use="required" />
      <xs:attribute name="id" type="xs:string" use="required" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe response='get_service_alarms'>
  <alarm dcp='Front Entrance' pd='5' time='18/08/2015 08:38:42'
    id='DCP_ALARM_TAMPER' />
</universe>
```

5.7 Clear Service Alarms

- Acknowledge and subsequently delete a contiguous range of alarms that were previously indicated to the connected interface.
- Only alarms applicable to the requesting interface will be acknowledged and deleted.
- The range of alarms are defined by *start* and *end* parameters that indicate an inclusive and contiguous range.

Xml Schema:

```
<xs:element name="clear_alarms">
  <xs:complexType>
    <xs:attribute name="start" type="xs:integer" use="required"/>
    <xs:attribute name="end" type="xs:integer" use="required"/>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe response='clear_alarms'>
  <clear_alarms start='0' end='3' />
</universe>
```

5.8 Get Service Events

- Request a list of events generated by the service.
 - Only events applicable to the requesting interface will be returned.
 - Service events utilise the asynchronous TCP socket connection to transmit alarm messages with low latency, i.e. polling can be used but is not required to receive events. These events are associated with the operation of the networks and are generally created as a result of the user issuing commands through the Universe Service interface.
 - If a client is not connected to the service then the events are discarded. **NOTE: In future a mechanism may be employed to store these events for later distribution, e.g. a database, storage file etc.**
 - Each message associated with an event uses the network name and OSDP address of a device to uniquely identify that device. **NOTE: General support for OSDP compliant devices can be assured using these unique identifications.**
 - The time associated with the event is recorded using the date and time that the OSDP Control Panel detected the event.
 - The type of event is encoded as a string according to the following permitted enumerations:
 - DCP_EVT_FIND – A *FindDevice* command has completed
 - DCP_EVT_INSTALL – An OSDP Peripheral Device install command has completed
 - DCP_EVT_REMOVE – An OSDP Peripheral Device uninstall command has completed
 - DCP_EVT_CHANGE_ADDR – A command to change the OSDP address of a Peripheral Device has completed
 - DCP_EVT_SET_PIDS – A command to set the general PIDS parameters of an OSDP Peripheral Device has completed
 - DCP_EVT_SET_IPV4 – A command to configure the IPv4 network settings of an OSDP Peripheral Device has completed
 - DCP_EVT_SOFT_RESET – A command to perform a software reset of an OSDP Peripheral Device has completed
 - DCP_EVT_SCS_HANDSHAKE – A user command to initiate a secure handshake with an OSDP Peripheral Device has completed
 - DCP_EVT_POLL – A user command to poll an OSDP Peripheral Device has completed
 - DCP_EVT_FW_DOWNLOAD – An event associated with the progress of a firmware update of an OSDP Peripheral Device
 - DCP_EVT_SAMPLE_STREAM – A command to stream sample data from an OSDP Peripheral Device has completed
 - DCP_EVT_DISABLE_ALARMS – A command to temporarily disable alarms for an OSDP Peripheral Device has completed
 - DCP_EVT_SECURE_SESSION – A command to configure the secure session duration for an OSDP Peripheral Device has completed
 - DCP_EVT_FW_STATUS – A command to retrieve the firmware status of an OSDP Peripheral Device has completed
 - DCP_EVT_GET_GPS – A command to retrieve the GPS information for an OSDP Peripheral Device has completed
 - DCP_EVT_SET_POLL_PERIOD – A command to set the polling period of an OSDP device has completed
 - DCP_EVT_SET_DEBUG_ALARMS – A command to simulate device alarms for an OSDP Peripheral Device has completed
 - DCP_EVT_SET_DEADZONES – A command to configure the deadzone parameters of
-

an OSDP Peripheral Device has completed

- DCP_EVT_SET_ALGORITHM – A command to set the microwave algorithm parameters of an OSDP Peripheral Device has completed
- DCP_EVT_SET_TRACE_LEVEL – A command to set the debug tracing level of the firmware operating on an OSDP Peripheral Device has completed
- DCP_EVT_UNKNOWN – An unspecified event has occurred

▪ **Xml Schema:**

- `<xs:element name="event" maxOccurs="unbounded" >`
- `<xs:complexType>`
- `<xs:sequence>`
- `<xs:attribute name="dcp" type="xs:string" use="required" />`
- `<xs:attribute name="pd" type="xs:integer" use="required" />`
- `<xs:attribute name="time" type="xs:string" use="required" />`
- `<xs:attribute name="msg" type="xs:string" use="required" />`
- `<xs:attribute name="id" type="xs:string" use="required" />`
- `</xs:sequence>`
- `</xs:complexType>`
- `</xs:element>`
-

▪ **Example:**

- `<universe response='get_service_events'>`
- `<event dcp='Front Entrance' pd='5' time='18/08/2015 08:38:42'`
- `msg='32.0854' id='DCP_EVT_FW_DOWNLOAD' />`
- `</universe>`

5.9 Clear Service Events

- Acknowledge and subsequently delete a contiguous range of events that were previously indicated to the connected interface.
- Only events applicable to the requesting interface will be acknowledged and deleted.
- The range of events are defined by *start* and *end* parameters that indicate an inclusive and contiguous range.

Xml Schema:

```
<xs:element name="clear_events">
  <xs:complexType>
    <xs:attribute name="start" type="xs:integer" use="required"/>
    <xs:attribute name="end" type="xs:integer" use="required"/>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe response='clear_events'>
  <clear_events start='0' end='0' />
</universe>
```

5.10 Get Network Information

- Used to retrieve the configuration and status of a specific OSDP network being maintained by the service using an instance of a DCP.
- Provides a list of every Peripheral Device connected to the specified OSDP network, and some basic information regarding the status of each of those devices.
- The OSDP address of each PD is provided as an integer in the range 0 to 126 inclusive.
- A boolean flag indicates if OSDP Secure Channel Session is enabled.
- The unique serial number of each device is encoded as an upper case hexadecimal string.
- The operating mode of each PD is described with a string using the following values:
 - “TRANSMITTER”
 - “RECEIVER”
- The status of each PD is described with a string using the following values:
 - “Connected”
 - “Disconnected”

Xml Schema:

```
<xs:element name="dcp">
  <xs:complexType>
    <xs:element name="name" type="xs:string"/>
  </xs:complexType>
</xs:element>

<xs:element name="pd">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="dcp" type="xs:string" />
      <xs:element name="osdp_addr" type="xs:integer" />
      <xs:element name="osdp_scs" type="xs:boolean" />
      <xs:element name="serial" type="xs:string" />
      <xs:element name="operating_mode" type="xs:string" />
      <xs:element name="status" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='get_network_info'>
  <dcp>
    <name>Front entrance</name>
  </dcp>
</universe>

<universe response='get_network_info'>
  <pd>
    <dcp>Front entrance</dcp>
    <osdp_addr>8</osdp_addr>
    <osdp_scs>0</osdp_scs>
    <serial>BEEF9137</serial>
```

```

    <operating_mode>TRANSMITTER</operating_mode>
    <status>Connected</status>
  </pd>
</universe>

```

5.11 Add Network

- A command to add an OSDP network to a Universe Service. This command will dynamically create a new OSDP network and save its configuration for automatic restart with the Universe Service.
- The response is identical to that of a Get Service Information command.
- Each network provides control for up to 127 OSDP Peripheral Devices.
- Each network **MUST** have a unique name for identification purposes.
- The master port is the TCP port to be used internally by the service to issue commands and receive replies from connected devices. This port is protected by mutual authentication using randomly generated public/private key pairs within an automated PAKE system that is robust to MITM attacks. The port is encoded as an integer.
- The slave port is the TCP port to be used by all external devices to connect to the system. This port can only be used to receive replies on an OSDP network. This port is encoded as an integer.
- The delay between successive attempts to connect to a disconnected PD is described using the reconnection delay. This delay is described in **milliseconds** and is encoded as an integer.
- Support for TCP only communication is indicated using a boolean flag, i.e. this flag indicates if RS485 support on a network is enabled.
- The RS485 hardware interface is described using a string. The content of this string is OS dependent, e.g. "COM3" (Windows) or "/dev/ttyUSB0" (Linux).
- The RS485 baud rate to be used by a network is encoded as an integer.

Xml Schema:

```

<xs:element name="dcp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="master_port" type="xs:integer"/>
      <xs:element name="slave_port" type="xs:integer"/>
      <xs:element name="reconnection_delay" type="xs:integer"/>
      <xs:element name="tcp_only" type="xs:boolean"/>
      <xs:element name="rs485_device" type="xs:string" minOccurs="0" />
      <xs:element name="rs485_baud" type="xs:integer" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='add_network'>
  <dcp>

```

```

    <name>Front entrance</name>
    <master_port>27043</master_port>
    <slave_port>27042</slave_port>
    <reconnection_delay>15000</reconnection_delay>
    <tcp_only>false</tcp_only>
    <rs485_device>COM3</rs485_device>
    <rs485_baud>115200</rs485_baud>
  </dcp>
</universe>

<universe cmd='add_network'>
  <dcp>
    <name>Front entrance</name>
    <master_port>27043</master_port>
    <slave_port>27042</slave_port>
    <reconnection_delay>15000</reconnection_delay>
    <tcp_only>true</tcp_only>
  </dcp>
</universe>

```

5.12 Remove Network

- A command to remove an OSDP network from a service. This will dynamically stop the service and delete its configuration such that it will no longer exist.
- The response is identical to that of a Get Service Information command.
- The network is identified using its unique name.

Xml Schema:

```

<xs:element name="dcp">
  <xs:complexType>
    <xs:element name="name" type="xs:string"/>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='remove_network'>
  <dcp>
    <name>Front entrance</name>
  </dcp>
</universe>

```

5.13 Modify Network

- A command to edit the configuration of an OSDP network.
- Please refer to Section 4.2 for a complete description of adding, removing and modifying OSDP networks.
- Upon acceptance of the command, the specified OSDP network will be stopped, reconfigured and then restarted.

Xml Schema:

```

<xs:complexType name="commsType" abstract="true">
</xs:complexType>

<xs:complexType name="IpOnlyDcpType">
  <xs:complexContent>
    <xs:extension base="commsType">
      <xs:sequence>
        <xs:element name="tcp_only" type="xs:boolean" value="true" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="Rs485DcpType">
  <xs:complexContent>
    <xs:extension base="commsType">
      <xs:sequence>
        <xs:element name="tcp_only" type="xs:boolean" value="false" />
        <xs:element name="rs485_device" type="xs:string"/>
        <xs:element name="rs485_baud" type="xs:integer"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="dcp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="modified_name" type="xs:string"/>
      <xs:element name="master_port" type="xs:integer"/>
      <xs:element name="slave_port" type="xs:integer"/>
      <xs:element name="reconnection_delay" type="xs:integer"/>
      <xs:element name="rs485" type="commsType" minOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example 1:

The name must be modified therefore the current anme and the modified name must both be supplied.

```

<universe cmd='modify_dcp'>
  <dcp>
    <name>Front entrance</name>
    <modified_name>Outer Perimeter</modified_name>
    <master_port>27043</master_port>
    <slave_port>27042</slave_port>
  </dcp>
</universe>

```

```

    <reconnection_delay>15000</reconnection_delay>
    <tcp_only>false</tcp_only>
    <rs485_device>COM3</rs485_device>
    <rs485_baud>115200</rs485_baud>
  </dcp>
</universe>

```

Example 2:

```

<universe cmd='modify_dcp'>
  <dcp>
    <name>Front entrance</name>
    <modified_name>Outer Perimeter</modified_name>
    <master_port>27043</master_port>
    <slave_port>27042</slave_port>
    <reconnection_delay>15000</reconnection_delay>
    <tcp_only>true</tcp_only>
  </dcp>
</universe>

```

5.14 Get Device Information

- This command will retrieve detailed information for the specified device.
- A device is identified using the network it is associated with and its serial number.
- The information is provided within an *info* element which is nested within a *pd* element.
- The vendor is an Organisationally Unique Identifier (OUI) associated with the Ethernet MAC address and is unique to each manufacturer. The Sensurity OUI is AC:E9:7F. The vendor OUI is encoded as an integer.
- The model indicates the model of the device and is encoded as an integer. The following values are supported:
 - 0 – Halo device
 - 1 – Vigil device
- The version indicates the model version of the device and is encoded as an integer. The values 0 to 255 inclusive are supported.
- The firmware version of the PD is encoded as an integer. This hexadecimal string is formatted in uppercase with a preceding “0x”. This is a 24-bit integer which is sub-divided into three bytes by interpreting the version as a big-endian integer, e.g. “0x00010003” would be interpreted as version number 1.0.3.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="info">
        <xs:complexType>
          <xs:attribute name="vendor" type="xs:integer" use="required"/>
          <xs:attribute name="model" type="xs:integer" use="required"/>

```

```

        <xs:attribute name="version" type="xs:integer" use="required"/>
        <xs:attribute name="firmware" type="xs:integer" use="required"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_device_info'>
  <pd dcp='FrontEntrance' serial='0x01234567' />
</universe>

```

```

<universe response='get_device_info'>
  <pd dcp='FrontEntrance' serial='0x01234567'>
    <info vendor='11331967' model='0' version='0' firmware='65539' />
  </pd>
</universe>

```

5.15 Find Device

- This will issue a command to search a network for any PD's that are not currently installed, i.e. it will search those OSDP addresses that have not been consumed by an installed PD.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated at an arbitrary time after an accepted command to indicate the outcome of the *FindDevice* command. This arbitrary time is dependant upon number of devices present and the RS485 baud rate (if used), typically no more than 2 minutes should elapse before a response is generated.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as "0" for success and "1" for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *FindDevice* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then a PD was NOT found.

Xml Schema:

```

<xs:element name="dcp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='find_device'>

```

```

    <dcp>
      <name>Front Entrance</name>
    </dcp>
  </universe>

  <universe response='find_device'>
    <return dcp='Front Entrance' value='1' />
  </universe>

```

5.16 Install Device

- A command used to install a PD that is located at a specified OSDP address.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *InstallDevice* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *InstallDevice* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then a PD was NOT installed.

Xml Schema:

```

<xs:element name="dcp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="address" type="xs:integer" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='install_device'>
  <dcp>
    <name>Front Entrance</name>
    <address>0</address>
  </dcp>
</universe>

<universe response='install_device'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.17 Remove Device

- Issue a command to remove a specified device, i.e. uninstall the PD from the network such that it is no longer available or polled for alarms.

- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *RemoveDevice* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *RemoveDevice* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then a PD was NOT removed.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='remove_device'>
  <pd dcp='Front Entrance' serial='0x89ABCDEF' />
</universe>

<universe response='remove_device'>
  <return dcp='Front Entrance' value='0' />
</universe>
```

5.18 Upload Firmware Image

- Transfers a firmware image intended for a Sensurity PIDS device to the Service.
- Stores the firmware image in the root file system for subsequent use.
- The uploaded file will replace any existing file of the same name.
- A CRC of the firmware image **MUST** be computed on the client side. This CRC will be verified by the Universe Service before it responds to the command with success or failure.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure. The optional *dcp* attribute will be omitted.

Xml Schema:

```
<xs:element name="fw_image">
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" />
    <xs:attribute name="crc" type="xs:integer" />
    <xs:sequence>
      <xs:element name="address" type="xs:base64Binary" />
    </xs:sequence>
  </xs:complexType>
```

```
</xs:element>
```

Example:

```
<universe cmd='upload_firmware_image'>
  <fw_image name='ms010003.bin' crc='4294967295' >
    ...
  </fw_image>
</universe>

<universe response='upload_firmware_image'>
  <return value='0' />
</universe>
```

5.19 Delete Firmware Image

- Delete a list of specified files that are contained by the Universe Service.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.

Xml Schema:

```
<xs:element name="fw_image">
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" />
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='delete_firmware_image'>
  <fw_image name='ms010003.bin' />
</universe>

<universe response='delete_firmware_image'>
  <return value='0' />
</universe>
```

5.20 List Firmware Image

- Retrieve a list of the files stored by the Universe Service.

Xml Schema:

```
<xs:element name="fw_image" minOccurs="0" maxOccurs="unbounded" >
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='list_firmware_image'>
</universe>
```

```
<universe response='list_firmware_image'>
  <fw_image name='ms010000.bin' />
  <fw_image name='ms010003.bin' />
  <fw_image name='ms010101.bin' />
</universe>
```

5.21 Update Device

- Issue a command to update the firmware of a specified device, where that firmware has previously been uploaded using an *Upload Firmware Image* command.
- An immediate response will be provided that indicates if the command was accepted.
- The command provides the name of the firmware image to be used for the update.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- Periodic events will be generated to indicate the status of the firmware update (see Service Events). These events will use the *message* element to indicate the progress of the firmware download to the device as a float string. A negative value for the progress indicates that the download has failed. If the progress value reaches 100 then the firmware download was successful.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="update">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='update_device'>
  <pd dcp='Front Entrance' serial='0x01234567' >
    <update name='ms010003.bin' />
  </pd>
</universe>
```

```
<universe response='update_device'>
  <return dcp='Front Entrance' value='0' />
```

```

</universe>

<universe response='get_service_events'>
  <event dcp='Front Entrance' pd='5' time='2015-08-01 12:00:00'
    msg='45.946' id='DCP_EVT_FIRMWARE_DOWNLOAD' />
</universe>

```

5.22 Update Firmware Status

- Issue a command to retrieve detailed information regarding the fundamental firmware status of a Sensurity PIDS device. This command will retrieve this information from a device and transfer it to the Universe Service, but a *Get Firmware Status* command will subsequently be required to obtain the information.
- An immediate response will be provided that indicates if the command was accepted.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Update Firmware Status* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the firmware status was NOT retrieved.

Xml Schema:

```

<xs:element name="dcp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="address" type="xs:integer" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='update_firmware_status'>
  <dcp>
    <name>Front Entrance</name>
    <address>0</address>
  </dcp>
</universe>

<universe response='install_device'>
  <return dcp='Front Entrance' value='0' />
</universe>

<universe response='get_service_events'>
  <event dcp='Front Entrance' pd='0' time='2015-08-01 12:00:00'
    msg="" id='DCP_EVT_FW_STATUS' />
</universe>

```

5.23 Get Firmware Status

- Issue a command to retrieve the buffered firmware status of a device. The firmware status must be previously retrieved using the *Update Firmware Status* command.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="firmware">
        <xs:complexType>
          <xs:attribute name="device_uptime" type="xs:string" use="required"/>
          <xs:attribute name="mem_core_free" type="xs:string" use="required"/>
          <xs:attribute name="mem_heap_fragments" type="xs:string" use="required"/>
          <xs:attribute name="mem_heap_free_total" type="xs:string" use="required"/>
          <xs:attribute name="num_threads" type="xs:string" use="required"/>
          <xs:attribute name="fs_mounted" type="xs:string" use="required"/>
          <xs:attribute name="state" type="xs:string" use="required"/>
          <xs:attribute name="thresh_high" type="xs:string" use="required"/>
          <xs:attribute name="thresh_low" type="xs:string" use="required"/>
          <xs:attribute name="filtered_signal" type="xs:string" use="required"/>
          <xs:attribute name="dir_path_rssi" type="xs:string" use="required"/>
          <xs:attribute name="threshold" type="xs:string" use="required"/>
          <xs:attribute name="thresh_ok" type="xs:string" use="required"/>
          <xs:attribute name="tx_active" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='get_firmware_status'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_firmware_status'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <firmware device_uptime='115274'
      mem_core_free='24765'
      mem_heap_fragments='0'
      mem_heap_free_total='0'
      num_threads='23'
      fs_mounted='true'
      state='ST_NORMAL'
      thresh_high='2115'
      thresh_low='2021'
      filtered_signal='2075'
    >
```

```

        dir_path_rssi='634'
        threshold='50'
        thresh_ok='true'
        tx_active='true'
    />
</pd>
</universe>

```

5.24 Change Address

- Issue a command to change the OSDP address of a specified device.
- An immediate response will be provided that indicates if the command was accepted.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Change Address* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the OSDP address of the PD was NOT modified.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:attribute name="osdp_addr" type="xs:integer" use="required"/>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='change_address'>
  <pd dcp='Front Entrance' serial='0x01234567' osdp_addr='20' />
</universe>

<universe response='change_address'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.25 Soft Reset

- Issue a command to perform a soft reset of a specified device.
- An immediate response will be provided that indicates if the command was accepted.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.

Xml Schema:

```

<xs:element name="pd">

```

```

    <xs:complexType>
      <xs:attribute name="dcp" type="xs:string" use="required"/>
      <xs:attribute name="serial" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

```

Example:

```

<universe cmd='soft_reset'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

```

```

<universe response='soft_reset'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.26 Set Security

- Issue a command to enable/disable OSDP Secure Channel Session (OSDP-SCS) to provide authenticated encryption between the OSDP Control Panel and the Peripheral Devices.
- The OSDP-SCS session lifetime MUST also be defined (using units of minutes), with a minimum possible period of 30 minutes and a maximum of 1440 minutes. When the OSDP-SCS lifetime expires a handshake will occur between the Control Panel and the PD to establish a new session key.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Security* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Security* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the OSDP-SCS configuration of the PD was NOT modified.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="security">
        <xs:complexType>
          <xs:attribute name="enable" type="xs:integer" use="required"/>
          <xs:attribute name="lifetime" type="xs:integer" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

```

</xs:element>

Example:

```
<universe cmd='set_security'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <security enable='1' lifetime='480' />
  </pd>
</universe>
```

```
<universe response='set_security'>
  <return dcp='Front Entrance' value='-1' />
</universe>
```

5.27 Get Security

- Retrieve the OSDP Secure Channel Session (OSDP-SCS) configuration.
- An *enable* attribute indicates if the feature is enabled, where “1” indicates it is enabled and “0” indicates it is disabled.
- The session lifetime is described using an integer that represents the number of minutes between secure sessions.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="security">
        <xs:complexType>
          <xs:attribute name="enable" type="xs:integer" use="required"/>
          <xs:attribute name="lifetime" type="xs:integer" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='get_security'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>
```

```
<universe cmd='get_security'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <security enable='0' lifetime='0' />
  </pd>
</universe>
```

5.28 Set IPv4 Network Configuration

- Issue a command to set the IPv4 configuration.
- ***DHCP must be defined as false and is reserved for future use.***
- The IP address and TCP port of the OSDP Control Panel server interface must be defined.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set IPv4* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set IPv4* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the IPv4 configuration of the PD was NOT modified.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="ipv4">
        <xs:complexType>
          <xs:attribute name="dhcp" type="xs:boolean" use="required"/>
          <xs:attribute name="address" type="xs:string" use="required"/>
          <xs:attribute name="netmask" type="xs:string" use="required"/>
          <xs:attribute name="gateway" type="xs:string" use="required"/>
          <xs:attribute name="server_address" type="xs:string" use="required"/>
          <xs:attribute name="server_port" type="xs:integer" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='set_ipv4'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <ipv4 dhcp='false'
      address='192.168.1.100' netmask='255.255.255.0' gateway='192.168.1.254'
      server_address='192.168.1.10' server_port='27042' />
  </pd>
</universe>

<universe response='set_ipv4'>
  <return dcp='Front Entrance' value='0' />
</universe>
```

5.29 Get IPv4

- Retrieve the IPv4 settings of the selected PD.
- ***The DHCP flag is reserved for future use.***
- Basic IP settings include the IP address, netmask and gateway of the PD in order to provide a static IP address.
- The IP address and port of the OSDP Control Panel must be defined to allow the PD to automatically connect to the OSDP network upon startup.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="ipv4">
        <xs:complexType>
          <xs:attribute name="dhcp" type="xs:boolean" use="required"/>
          <xs:attribute name="address" type="xs:string" use="required"/>
          <xs:attribute name="netmask" type="xs:string" use="required"/>
          <xs:attribute name="gateway" type="xs:string" use="required"/>
          <xs:attribute name="server_address" type="xs:string" use="required"/>
          <xs:attribute name="server_port" type="xs:integer" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='get_ipv4'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_ipv4'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <ipv4 dhcp='false'
      address='192.168.1.100' netmask='255.255.255.0' gateway='192.168.1.254'
      server_address='192.168.1.10' server_port='27042' />
  </pd>
</universe>
```

5.30 Set Microwave Intrusion Parameters

- Issue a command to set the microwave intrusion detection parameters.
- The operating mode **MUST** be one of the following strings:
 - “TRANSMITTER”

- “RECEIVER”
- The channel of a transmitter/receiver pair must use the following permitted values:
 - “CHANNEL_<1:37>”
- The range must be defined to represent the distance between a receiver and its associated transmitter.
- The sensitivity is used to control the aggressiveness of the intrusion detection algorithm. Permitted values are as follows:
 - “SENSITIVITY_VERY_LOW”
 - “SENSITIVITY_LOW”
 - “SENSITIVITY_BELOW_AVERAGE”
 - “SENSITIVITY_AVERAGE”
 - “SENSITIVITY_ABOVE_AVERAGE”
 - “SENSITIVITY_HIGH”
 - “SENSITIVITY_VERY_HIGH”
 - “SENSITIVITY_EXTREME”
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Microwave* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Microwave* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the microwave settings of the PD have NOT been modified.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="microwave">
        <xs:complexType>
          <xs:attribute name="mode" type="xs:string" use="required"/>
          <xs:attribute name="ch" type="xs:string" use="required"/>
          <xs:attribute name="range" type="xs:integer"/>
          <xs:attribute name="sensitivity" type="xs:string"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_microwave'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <microwave mode='TRANSMITTER' ch='CHANNEL_3' />
  </pd>
</universe>

```

```

</universe>

<universe response='set_microwave'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.31 Get Microwave Intrusion Parameters

- Retrieve the microwave intrusion detection settings.
- The response is provided immediately.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="microwave">
        <xs:complexType>
          <xs:attribute name="mode" type="xs:boolean" use="required"/>
          <xs:attribute name="ch" type="xs:string" use="required"/>
          <xs:attribute name="range" type="xs:integer"/>
          <xs:attribute name="sensitivity" type="xs:string"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_microwave'>
  <pd dcp='Front Entrance' serial='0x89ABCDEF' />
</universe>

<universe response='get_microwave'>
  <pd dcp='Front Entrance' serial='0x89ABCDEF'>
    <microwave mode='RECEIVER' ch='CHANNEL_3'
      range='50' sensitivity='SENSITIVITY_AVERAGE' />
  </pd>
</universe>

```

5.32 Set Relay Parameters

NOTE: The 2nd relay on a Sensurity Vigil device is optional.

- Issue a command to configure the two relays on a Sensurity PD.
- The mode used to control the triggering of the relay must be one of the following:
 - “DISABLED”

- “NORMALLY_ON”
- “NORMALLY_OFF”
- “ENABLED”
- The duration defines the relay switching period in seconds. During this switching period further trigger events are discarded. The permitted range of values is 1 to 255 seconds.
- The trigger event is a 16-bit field that is used to define the enabled triggers that can cause a relay switching event. This bit-field is defined as follows:

EVENT	BITMASK	DESCRIPTION
Power	0x0001	Occurs during system initialisation if the PD has previously been subjected to a brown-out
Tamper	0x0002	Occurs when the rear panel optical sensor (if present) or the accelerometer are disturbed
Microwave	0x0004	Occurs when the microwave intrusion detection algorithm indicates that an intruder has passed through the microwave beam between a pair of devices.
Upper Deadzone	0x0008	Occurs when the upper deadzone (if fitted) is disturbed.
Lower Deadzone	0x0010	Occurs when the lower deadzone (if fitted) is disturbed.
Environmental	0x0020	Occurs when an unusual temperature fluctuation is detected.
Error	0x0040	Occurs when a system error occurs (i.e. SD card is not detected, system restarted due to an internal error).
Auxilliary 1	0x0080	An external signal can be used to trigger a relay event on a Sensurity PD. This signal is regularly sampled by an ADC on the Sensurity PD and if the signal fluctuates above or below a threshold value it is used to trigger an alarm event.
Auxilliary 2	0x0100	See Auxilliary 1
Auxilliary 3	0x0200	See Auxilliary 1
Auxilliary 4	0x0400	See Auxilliary 1

- For example, to enable trigger events for power and microwave the bitmask should be set to $0x0001 \mid 0x0004 = 0x0005$.
- For example, to enable trigger events for microwave and both deadzones the bitmask should be set to $0x0004 \mid 0x0008 \mid 0x0010 = 0x001C$.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Relays* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Relays* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the relay

settings of the PD have NOT been modified.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="relays">
        <xs:complexType>
          <xs:attribute name="mode1" type="xs:string" use="required"/>
          <xs:attribute name="dur1" type="xs:integer" use="required"/>
          <xs:attribute name="trig1" type="xs:integer" use="required" />
          <xs:attribute name="mode2" type="xs:string" />
          <xs:attribute name="dur2" type="xs:integer" />
          <xs:attribute name="trig2" type="xs:integer" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='set_relays'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <microwave mode1='NORMALLY_ON' dur1='5' trig1='12' />
  </pd>
</universe>

<universe response='set_relays'>
  <return dcp='Front Entrance' value='0' />
</universe>
```

5.33 Get Relay Parameters

- Retrieve the relay settings for the specified Sensurity PD.
- The response is provided immediately.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="relays">
        <xs:complexType>
          <xs:attribute name="mode1" type="xs:string" use="required"/>
          <xs:attribute name="dur1" type="xs:integer" use="required"/>
          <xs:attribute name="trig1" type="xs:integer" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```

        <xs:attribute name="mode2" type="xs:string" />
        <xs:attribute name="dur2" type="xs:integer" />
        <xs:attribute name="trig2" type="xs:integer" />
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_relays'>
  <pd dcp='Front Entrance' serial='0x89ABCDEF' />
</universe>

<universe response='get_relays'>
  <pd dcp='Front Entrance' serial='0x89ABCDEF'>
    <microwave mode1='NORMALLY_ON' dur1='5' trig1='12'
      mode2='NORMALLY_OFF' dur2='3' trig2='32' />
  </pd>
</universe>

```

5.34 Set Deadzones

- Issue a command to set the Upper and Lower Deadzones.
- The deadzones can be enabled (if fitted).
- The sensitivity of the lower and upper thresholds are reserved parameters that will be configurable in the future.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Deadzones* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Deadzones* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the deadzone settings of the PD have NOT been modified.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="deadzones">
        <xs:complexType>
          <xs:attribute name="enable_upper" type="xs:boolean" use="required"/>
          <xs:attribute name="sensitivity_upper" type="xs:integer" />
          <xs:attribute name="enable_lower" type="xs:boolean" use="required"/>

```

```

        <xs:attribute name="sensitivity_lower" type="xs:integer" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_deadzones'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <deadzones enable_upper='1' sensitivity_upper='0' enable_lower='0' />
  </pd>
</universe>

<universe response='set_deadzones'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.35 Get Deadzones

- Retrieve the deadzone configuration of the selected Sensurity PD.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="deadzones">
        <xs:complexType>
          <xs:attribute name="enable_upper" type="xs:boolean" use="required"/>
          <xs:attribute name="sensitivity_upper" type="xs:integer" use="required"/>
          <xs:attribute name="enable_lower" type="xs:boolean" use="required"/>
          <xs:attribute name="sensitivity_lower" type="xs:integer" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_deadzones'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_deadzones'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <deadzones enable_upper='1' sensitivity_upper='0'

```

```

        enable_lower='0' sensitivity_lower='0' />
    </pd>
</universe>

```

5.36 Get Deadzones Fitted

- Retrieve flags to indicate if the upper and/or lower deadzones are fitted.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="deadzones">
        <xs:complexType>
          <xs:attribute name="fitted_upper" type="xs:boolean" use="required"/>
          <xs:attribute name="fitted_lower" type="xs:boolean" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_deadzones_fitted'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_deadzones_fitted'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <deadzones fitted_upper='1' fitted_lower='0' />
  </pd>
</universe>

```

5.37 Set Alarms

- Issue a command to configure which of the six alarm groups are enabled on a Sensurity PD.
- The alarm groups are classified as follows:
 - Intruder – microwave and upper/lower deadzone
 - Environmental – temperature
 - Error – SD card not mounted
 - Power – Power supply interruption
 - Tamper – Rear panel removed (if present), device movement
 - External – External ADC's
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Alarms* command.

- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Alarms* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the alarms settings of the PD have NOT been modified.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="alarms">
        <xs:complexType>
          <xs:attribute name="intruder" type="xs:boolean" use="required"/>
          <xs:attribute name="env" type="xs:boolean" use="required" />
          <xs:attribute name="error" type="xs:boolean" use="required"/>
          <xs:attribute name="power" type="xs:boolean" use="required" />
          <xs:attribute name="tamper" type="xs:boolean" use="required" />
          <xs:attribute name="ext" type="xs:boolean" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='set_alarms'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <alarms intruder='1' env='1' error='1' power='1' tamper='1' ext='0' />
  </pd>
</universe>

<universe response='set_alarms'>
  <return dcp='Front Entrance' value='0' />
</universe>
```

5.38 Get Alarms

- Retrieve the enabled alarm classes on a Sensurity PD.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
```

```

    <xs:element name="alarms">
      <xs:complexType>
        <xs:attribute name="intruder" type="xs:boolean" use="required"/>
        <xs:attribute name="env" type="xs:boolean" use="required" />
        <xs:attribute name="error" type="xs:boolean" use="required"/>
        <xs:attribute name="power" type="xs:boolean" use="required" />
        <xs:attribute name="tamper" type="xs:boolean" use="required" />
        <xs:attribute name="ext" type="xs:boolean" use="required" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_alarms'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_alarms'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <alarms intruder='1' env='1' error='1' power='1' tamper='1' ext='0' />
  </pd>
</universe>

```

5.39 Set ADC

- Issue a command to configure which of the External ADC's are enabled on a Sensurity PD.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set ADC* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set ADC* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the ADC settings of the PD have NOT been modified.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="adc">
        <xs:complexType>
          <xs:attribute name="adc0" type="xs:boolean" />
          <xs:attribute name="adc1" type="xs:boolean" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        <xs:attribute name="adc2" type="xs:boolean"/>
        <xs:attribute name="adc3" type="xs:boolean" />
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_adc'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <adc adc0='1' adc1='0' adc2='0' adc3='0' />
  </pd>
</universe>

<universe response='set_adc'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.40 Get ADC

- Retrieve the ADC configuration of the specified Sensurity PD.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="adc">
        <xs:complexType>
          <xs:attribute name="adc0" type="xs:boolean" use="required"/>
          <xs:attribute name="adc1" type="xs:boolean" use="required"/>
          <xs:attribute name="adc2" type="xs:boolean" use="required"/>
          <xs:attribute name="adc3" type="xs:boolean" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_adc'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe cmd='get_adc'>
  <pd dcp='Front Entrance' serial='0x01234567'>

```

```

        <adc adc0='1' adc1='0' adc2='0' adc3='0' />
    </pd>
</universe>

```

5.41 Set Polling Period

- Configure the polling period for a specified PD. This polling period is defined in *milliseconds* and represents the time duration between polling of a PD by the OSDP Control Panel.
- In a large network of devices the system may require a larger latency between polling events to ensure that overall system performance remains acceptable and is not undermined by rapidly polling PD's for alarms at a stressful rate.
- Each PD can be individually assigned a different polling period to ensure that those PD's that are located at a sensitive location on the perimeter can indicate alarms with lower latency than those PD's that are considered to be of lesser importance.
- The default polling period when a PD is installed is 1000 ms.
- An immediate response will be provided that indicates if the command was accepted.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="polling">
        <xs:complexType>
          <xs:attribute name="period" type="xs:integer" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_polling_period'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <polling period='1000' />
  </pd>
</universe>

<universe response='set_polling_period'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.42 Get Polling Period

- Retrieve the polling period to be used for a specified PD.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="polling">
        <xs:complexType>
          <xs:attribute name="period" type="xs:integer" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_polling_period'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_polling_period'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <polling period='1000' />
  </pd>
</universe>

```

5.43 Set Algorithm

- Issue a command to select which microwave intruder algorithm is used to detect intruders.
- The possible algorithms must be in the range of 0 to 2 inclusive.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Algorithm* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Algorithm* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the microwave intruder algorithm of the PD has NOT been modified.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>

```

```

    <xs:element name="algorithm">
      <xs:complexType>
        <xs:attribute name="id" type="xs:integer" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_algorithm'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <algorithm id='2' />
  </pd>
</universe>

<universe response='set_algorithm'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.44 Get Algorithm

- Retrieve the microwave intruder algorithm to be used by the selected PD.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="algorithm">
        <xs:complexType>
          <xs:attribute name="id" type="xs:integer" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_algorithm'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_algorithm'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <algorithm id='0' />
  </pd>

```

</universe>

5.45 Update GPS

- Issue a command to retrieve the location from a Sensurity PD.
- Under normal operation the OSDP Control Panel provided by the Universe Service does not regularly update the location of each PD. Before using the *GetGPS* command described below it is necessary to have updated the PD location using the *UpdateGPS* command.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Update GPS* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Update GPS* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the location of the PD has NOT been updated.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='update_gps'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='update_gps'>
  <return dcp='Front Entrance' value='0' />
</universe>
```

5.46 Get GPS

- Retrieve the location of the specified Sensurity PD.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="gps">
```

```

    <xs:complexType>
      <xs:attribute name="fix" type="xs:boolean" use="required" />
      <xs:attribute name="day" type="xs:integer" use="required" />
      <xs:attribute name="month" type="xs:integer" use="required" />
      <xs:attribute name="year" type="xs:integer" use="required" />
      <xs:attribute name="hour" type="xs:integer" use="required" />
      <xs:attribute name="min" type="xs:integer" use="required" />
      <xs:attribute name="sec" type="xs:integer" use="required" />
      <xs:attribute name="latitude" type="xs:float" use="required" />
      <xs:attribute name="longitude" type="xs:float" use="required" />
      <xs:attribute name="altitude" type="xs:float" use="required" />
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_gps'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_gps'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <gps fix='1' day='15' month='8' year='2015' hour='16' min='3' sec='41'
      latitude='54.504852' longitude='-6.212623' altitude='71.0' />
  </pd>
</universe>

```

5.47 Set Trace Level

- Issue a command to set the debug trace level of messages emanating from the embedded software of the Sensurity PD's.
- An immediate response will be provided that indicates if the command was accepted.
- An event will be generated some time after an accepted command to indicate the outcome of the *Set Trace Level* command.
- The response to this command will be a return packet, described in . The string value returned will be interpreted as “0” for success and “1” for failure.
- If a command is issued successfully an event will subsequently be generated (see Service Events). This event will detail the results of the *Set Trace Level* command, indicating the name of the OSDP network and the OSDP address of the PD. If the OSDP address is -1 then the debug trace level of the PD has NOT been updated.
- Possible trace levels are as follows:
 - TRACE_DISABLED
 - TRACE_ERROR
 - TRACE_WARNING
 - TRACE_NOTICE
 - TRACE_DEBUG

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="trace">
        <xs:complexType>
          <xs:attribute name="level" type="xs:string" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_trace_level'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <trace level='TRACE_NOTICE' />
  </pd>
</universe>

<universe response='set_trace_level'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.48 Get Trace Level

- Retrieve the debug trace level of a Sensurity PD.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="trace">
        <xs:complexType>
          <xs:attribute name="level" type="xs:string" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_trace_level'>

```

```

    <pd dcp='Front Entrance' serial='0x01234567' />
</universe>

<universe response='get_trace_level'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <trace level='TRACE_NOTICE' />
  </pd>
</universe>

```

5.49 Get Trace

- Retrieve a debug trace message containing a single human readable trace message from a Sensurity PD.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="trace">
        <xs:complexType>
          <xs:attribute name="msg" type="xs:string" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='get_trace'>
  <pd dcp='Front Entrance' serial='0x01234567' /pd>
</universe>

<universe response='get_trace'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <trace msg='Hello world from Sensurity Halo' />
  </pd>
</universe>

```

5.50 Set Enable Samples

- Issue a command to enable the streaming of sample data from a Sensurity PD.
- NOTE: Sample type is reserved for future use, currently only microwave samples can be selected for streaming.**
- An immediate response will be provided that indicates if the command was accepted.
- Subsequent to enabling or disabling the streaming of sample data the Universe Service will commence or cease streaming respectively.

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="samples">
        <xs:complexType>
          <xs:attribute name="type" type="xs:string" use="required" />
          <xs:attribute name="enable" type="xs:bool" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Example:

```

<universe cmd='set_enable_samples'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <samples type='microwave' level='TRACE_NOTICE' />
  </pd>
</universe>

<universe response='set_enable_samples'>
  <return dcp='Front Entrance' value='0' />
</universe>

```

5.51 Get Enable Samples

- Retrieve a flag indicating if sample streaming is enabled for a specified Sensurity PD.
- The maximum depth of the buffer into which sample data is stored by the Sensurity Service is indicated in terms of the number of samples.
- The current fill level of the buffer is indicated in terms of the number of samples.
- **NOTE: Sample type is reserved for future use, currently only microwave samples can be selected for streaming.**

Xml Schema:

```

<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="samples">
        <xs:complexType>
          <xs:attribute name="type" type="xs:string" use="required" />
          <xs:attribute name="enable" type="xs:boolean" use="required" />
          <xs:attribute name="depth" type="xs:integer" use="required" />
          <xs:attribute name="level" type="xs:integer" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
```

Example:

```
<universe cmd='get_enable_samples'>
  <pd dcp='Front Entrance' serial='0x01234567' />
</universe>
```

```
<universe response='get_enable_samples'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <samples type='microwave' enable='1' depth="8192" level="4231" />
  </pd>
</universe>
```

6 Push List

6.1 Heartbeat

- Once an authenticated user has established a connection (or SRP PAKE is disabled and a connection has been established) the service will begin to emit heartbeat event messages every 15 seconds.
- These messages are to provide confidence that the service is operating whilst it is otherwise silent due to no other events or alarms from the server side or commands received from the client side.

Example:

```
<universe event='heartbeat'>
</universe>
```

6.2 Service Alarms

- See Section Get Service Alarms for a full description.

Example:

```
<universe response='get_service_alarms'>
  <alarm dcp='Front Entrance' pd='5' time='18/08/2015 08:38:42'
    id='DCP_ALARM_TAMPER' />
</universe>
```

6.3 Service Events

- See Section Get Service Events for a full description.

Example:

```
<universe response='get_service_events'>
  <event dcp='Front Entrance' pd='5' time='18/08/2015 08:38:42'
    msg='32.0854' id='DCP_EVT_FW_DOWNLOAD' />
</universe>
```

6.4 Sample Data

- Sample data captured by Sensurity PIDS devices can be streamed from the device, buffered using the Universe Service and subsequently relayed to connected client applications.
 - **NOTE: Currently the type must be microwave, this attribute is reserved for future use such that different types of sample data can be viewed by the user.**
 - The sample data is periodically pushed by the Universe Service to the client application if it has requested that sample streaming be enabled using the *Set Enable Samples* command.
-

- The sample data is base64 encoded and should subsequently be decoded into an array of 16-bit integers.

Xml Schema:

```
<xs:element name="pd">
  <xs:complexType>
    <xs:attribute name="dcp" type="xs:string" use="required"/>
    <xs:attribute name="serial" type="xs:string" use="required"/>
    <xs:sequence>
      <xs:element name="samples">
        <xs:complexType>
          <xs:attribute name="type" type="xs:string" use="required" />
          <xs:element name="data" type="xs:base64Binary" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example:

```
<universe response='get_sample_data'>
  <pd dcp='Front Entrance' serial='0x01234567'>
    <samples type='microwave' >
      <data>
        ... base64 encoded samples ...
      </data>
    </samples>
  </pd>
</universe>
```

7 Examples of Use

The following examples are intended to illustrate typical conversations that a client application would have with the Sensurity Universe Service.

7.1 Secure Remote Password Authentication

TODO

7.2 Installing an OSDP Network

TODO

7.3 Installing an OSDP Peripheral Device

TODO

7.4 Retrieving Device Settings

TODO

7.5 Configuring Device Settings

TODO

7.6 Uploading a Firmware Image to the Service

TODO

7.7 Updating Device Firmware

TODO

Bibliography

- 1: SIA, Open Supervised Device Protocol (OSDP) Version 2.1.5, 2012
- 2: Sensurity Ltd., Distributed Control Panel API User Guide, 2015