WFA WLANConfig Service 1.0

Service Template Version 1.01

January 2006

Contents

List of Tables

1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0.

This service enables the configuration of IEEE 802.11 wireless stations and Access Points using IP based channels, such as Ethernet and provides a proxy function between the IP network and 802.11 Simple Config frames. It leverages the data structure and protocols defined for Simple Config to reduce device complexity for supporting the UPnP methods.

This service-type enables the following functions:

- Device type and information.
- Proxy function on Access Points and optionally other devices to enable IP based Registrars to configure WLAN Enrollees.
- Device information and capabilities.
- Secured exchange with UPnP devices for retrieving and configuring wireless parameters for Access Point management and wireless setup of Ethernet attached stations.

1.1. Change Log

- V0.1 (4/22/05), document created.
- V0.9 (6/3/05), edits for WFA first submittal
- V0.95 (8/5/05), updates, edits for WFA external review
- V0.98 (1/18/06), aligned with V0.98h simple config spec & comments

2. Service Modeling Definitions

2.1. ServiceType

The service is REQUIRED as specified in
urn:schemas-wifialliance-org:device:*WFADevice:1*

The following service type identifies a service that is compliant with this template:
urn:schemas-wifialliance-org:*WFAWLANConfig:1*

This service does not support the QueryStateVariable action.

2.2. State Variables

Table 1 shows all the state variables of the WFAWLANConfig service. These variables are represented in the order:

- The first set of variables is for message exchange
- The second set shows the device capabilities message.
- The third set has the device configuration messages
- The last set lists evented messages

Table 1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
Message	R	Bin.base64	See Simple Config	N/A	N/A
InMessage	R	Bin.base64	See Simple Config	N/A	N/A
OutMessage	R	Bin.base64	See Simple Config	N/A	N/A
DeviceInfo	R	Bin.base64		N/A	
APSettings	C	Bin.base64		N/A	N/A
APStatus	C	ui1	See Table 1.1	0	N/A
STASettings	C	Bin.base64		N/A	N/A
STAStatus	C	ui1	See Table 1.1	0	N/A
WLANEvent	C	Bin.base64	See Simple Config	N/A	
WLANEventType	C	ui1	See Table 1.2	N/A	
WLANEventMAC	C	String	MAC Address, "xx:xx:xx:xx:xx:xx", case-independent, 17 char	N/A	
Non-standard state variables implemented by an UPnP vendor go here.	*X*	*TBD*	*TBD*	*TBD*	*TBD*

[1] R = Required, O = Optional, C = Conditional, X = Non-standard.

[2] Values listed in this column are required. To specify standard optional values or to delegate assignment of values to the vendor, you must reference a specific instance from the appropriate table below.

2.2.1. Message

This variable contains the TLV format for the data components passed in the message exchange between the Registrar and the Enrollee. The Message identification (M2-M8) is contained in the TLV binary format. Messages received from WLAN devices via Probe Requests and Simple Config 802.1X/EAP methods are evented.

2.2.2. InMessage

This variable contains the TLV format for the data components passed in the message exchange between the Registrar and the Enrollee. The message identification (M2-M8) is contained in the TLV binary format. Messages received from WLAN devices via Probe Requests and Simple Config 802.1X/EAP methods are evented. This provides separation of state variables for actions that use both input and output messages, such as PutMessage.

2.2.3. OutMessage

This variable contains the TLV format for the data components passed in the message exchange between the Registrar and the Enrollee. The message identification (M2-M8) is contained in the TLV binary format. Messages received from WLAN devices via Probe Requests and Simple Config 802.1X/EAP methods are evented. This provides separation of state variables for actions that use both input and output messages, such as PutMessage.

2.2.4. DeviceInfo

This variable contains the Simple Config data components in TLV format that are provided in message M1. The allowed list is defined in the Wi-fi Simple Config specification.

2.2.5. APSettings

This variable contains the TLV format for the data components passed in the management interface between the Registrar and the Access Point and carries a message that is defined based on the action. This variable is required for Access Points.

AP settings are authenticated, and some parts are also encrypted. The symmetric keys for authentication and encryption are derived during the Registrar configuration process. The SID attribute included in the messages enables the AP and Registrar to determine which keys to use. The APSettings data components are transported in the Encrypted Settings data component value field.

The AP settings data components are described in the Simple Config Specification. Attributes retrieved by GetAPSettings are described as GetAPSettings Output Message. Those set by SetAPSettings, are described as SetAPSettings Message. Those removed by DelAPSettings are described as DelAPSettings Message.

2.2.6. APStatus

This variable represents changes in the status of an AP. It is used in eventing to indicate changes on the AP that are related to configuration and attack mitigation. Table 1.1 indicates the flag values of this field. It is required for APs.

Table 1.1: allowedValueList for APStatus

Value	Req. or Opt. [1]	Description
0x00000001	R	Configuraiton Change
0x00000010	R	Failed auth threshold reached (AP Locked)

2.2.7. STASettings

This variable contains the TLV format for the data components passed in the configuration interface between the Registrar and an established Station on the WLAN and carries a message that is defined based on the action. This variable is required for 802.11 Stations that support Ethernet configuration.

STA settings are encrypted and authenticated. The symmetric key is derived during the Registrar configuration process and is referenced using the Key Identifier data component. The STASettings data components are transported in the Encrypted Settings data component value field (preceded by a Key Identifier).

The STA Settings data components are described in the Simple Config Specification in the section entitled STASettingsMessage.

2.2.8. STAStatus

This variable represents changes in the status of a Station. It is used in eventing to indicate changes on the Station that are related to configuration and attack mitigation. Table 1.1 indicates the flag values of this field. It is required for 802.11 Stations.

2.2.9. WLANEvent

This variable represents the concatentation of the WLANEventType, WLANEventMac and the 802.11 WSC message received from the Enrollee & forwarded by the proxy over UPnP. It is required for Access Points and other proxies. WLANEvent is represented as base64 (WLANEventType || WLANEventMAC || Enrollee's message).

2.2.10.WLANEventType

This variable represents the type of WLANEvent frame that was received by the proxy on the 802.11 network. The Table 1.1 shows the options for this variable. This variable is required for Access Points and any other device that implements the proxy function.

Table 1.2: allowedValueList for WLANEventType

Value	Req. or Opt. [1]	Description
1	R	802.11 Simple Config Probe Frame
2	R	802.11 Simple Config 802.1X/EAP Frame

2.2.11.WLANEventMAC

This variable represents the MAC address of the WLAN Enrollee that generated the 802.11 frame that was received by the proxy. This variable is required for Access Points and any other device that implements the proxy function.

2.3. Eventing and Moderation

Table 2 lists the Events generated in this service where the first three are generated as part of a single propertyset & the remainder are independently generated. A single event is generated by each WFA

Simple Config probe and 802.1X/Simple Config EAP frame received on the 802.11 network. The remaining state variables are not evented.

Table 2: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
WLANEvent	Yes	Yes	.2	OR	
APStatus	Yes	Yes	.5	OR	
STAStatus	Yes	Yes	.5	OR	
Non-standard state variables implemented by an UPnP vendor go here.	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

[1] Determined by N, where Rate = (Event)/(N secs).
[2] (N) * (allowedValueRange Step).

2.3.1. Event Model

Three state variables in the WFAWLANConfig service are simultaneously evented:

1. WLANEvent: This state variable event indicates that an Simple Config Probe-Request or an 802.1X/Simple Config EAP frame was received on the 802.11 network. The proxy service issues the event.
2. WLANEventType: Evented with the WLANEvent variable. This indicates the type of 802.11 frame received.
3. WLANEventMAC: Evented with the WLANEvent variable. This indicates the MAC address of the source of the 802.11 frame.

The event is moderated. The limit is 5 events per second.

2.4. Actions

Table 3 lists the required and optional actions for the WFA device WFAWLANConfig service. This is followed by detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 3: Actions

Name	Req. or Opt.
GetDeviceInfo	R
PutMessage	R
GetAPSettings	C
SetAPSettings	C
DelAPSettings	C

Name	Req. or Opt.
GetSTASettings	C
SetSTASettings	C
DelSTASettings	C
PutWLANResponse	C
SetSelectedRegistrar	C
RebootAP	O
ResetAP	C
RebootSTA	O
ResetSTA	C

[1] R = Required, O = Optional, C = Conditional.

2.4.1. GetDeviceInfo

This action retrieves the device's M1 data. It is required for APs and Ethernet attached stations.

2.4.1.1. Arguments

Table 3.6: Arguments for GetDeviceInfo

Argument	Direction	relatedStateVariable
NewDeviceInfo	OUT	DeviceInfo

2.4.1.2. Dependency on State (if any)

2.4.1.3. Effect on State (if any)

2.4.1.4. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.2. PutMessage

This action continues the Simple Config exchange over UPnP to permit establishment of a Registrar with an Access Point and to configure an Ethernet (IP) attached wireless station. A PutMessage is issued only after receiving M1 from a device from a GetDeviceInfo. PutMessage is used first to send M2 from the Registrar to the device (for which a M3 response is expected). Additional round trips in the Registration Protocol are accomplished with subsequent calls to PutMessage.

2.4.2.1. Arguments

Table 3.2: Arguments for PutMessage

Argument	Direction	relatedStateVariable
NewInMessage	IN	InMessage
NewOutMessage	OUT	OutMessage

2.4.2.2. Dependency on State (if any)

2.4.2.3. Effect on State (if any)

2.4.2.4. Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

2.4.3. GetAPSettings

This action is used to directly retrieve settings from an Access Point for which the Registrar already has established a secure association with. The input message is the GetAPSettings Input Message in the Simple Config specification. The output message is the GetAPSettings Output Message in the Simple Config specification. This action is required for Access Points.

2.4.3.1. Arguments

Table 3.3: Arguments for GetAPSettings

Argument	Direction	relatedStateVariable
NewMessage	IN	Message
NewAPSettings	OUT	APSettings

2.4.3.2. Dependency on State (if any)

2.4.3.3. Effect on State (if any)

2.4.3.4. Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

2.4.4. SetAPSettings

This action is used to set settings on an Access Point for which the Registrar already has established a secure association with. The input message is the SetAPSettings Message in the Simple Config specification. This action is required for Access Points.

2.4.4.1. Arguments

Table 3.4: Arguments for SetAPSettings

Argument	Direction	relatedStateVariable
NewAPSettings	IN	APSettings

2.4.4.2. Dependency on State (if any)

2.4.4.3. Effect on State (if any)

2.4.4.4. Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

2.4.5. DelAPSettings

This action is used to delete settings or entries on an Access Point for which the Registrar already has established a secure association with. The input message is the DelAPSettings Message in the Simple Config specification. This action is required for Access Points.

2.4.5.1. Arguments

Table 3.4: Arguments for DelAPSettings

Argument	Direction	relatedStateVariable
NewAPSettings	IN	APSettings

2.4.5.2. Dependency on State (if any)

2.4.5.3. Effect on State (if any)

2.4.5.4. Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

501	Action Failed	See UPnP Device Architecture section on Control.

2.4.6. GetSTASettings

This action retrieves the settings from an Ethernet attached wireless device for which a Registrar has already established a secure association with. The input message is the GetSTASettings Input Message in the Simple Config specification. The output message is the GetSTASettings Output Message in the Simple Config specification. It is required for Ethernet attached wireless stations.

2.4.6.1. Arguments

Table 3.5: Arguments for GetSTASettings

Argument	Direction	relatedStateVariable
Message	IN	Message
NewSTASettings	OUT	STASettings

2.4.6.2. Dependency on State (if any)

2.4.6.3. Effect on State (if any)

2.4.6.4. Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

2.4.7. SetSTASettings

This action sets settings on an Ethernet connected wireless device for which a Registrar has already established a secure association with. The input message is the SetSTASettings Input Message in the Simple Config specification. It is required for Ethernet attached wireless stations.

2.4.7.1. Arguments

Table 3.6: Arguments for SetSTASettings

Argument	Direction	relatedStateVariable
NewSTASettings	IN	STASettings

2.4.7.2. Dependency on State (if any)

2.4.7.3. Effect on State (if any)

2.4.7.4. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.8. DelSTASettings

This action is used to delete settings or entries on an Access Point for which the Registrar already has established a secure association with. The input message is the DelSTASettings Input Message in the Simple Config specification. This action is required for Ethernet attached wireless stations..

2.4.8.1. Arguments

Table 3.4: Arguments for DelSTASettings

Argument	Direction	relatedStateVariable
NewSTASettings	IN	STASettings

2.4.8.2. Dependency on State (if any)

2.4.8.3. Effect on State (if any)

2.4.8.4. Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

2.4.9. PutWLANResponse

This action is used to send messages to a wireless Enrollee via the proxy function. The Registrar must subscribe to events from the proxy function which forward Enrollee requests and the Registrar responds via PutWLANResponse to continue the Registration protocol exchange.

2.4.9.1. Arguments

Table 3.6: Arguments for PutWLANResponse

Argument	Direction	relatedStateVariable
NewMessage	IN	Message

Argument	Direction	relatedStateVariable
NewWLANEventType	IN	WLANEventType
NewWLANEventMAC	IN	WLANEventMAC

2.4.9.2. Dependency on State (if any)

2.4.9.3. Effect on State (if any)

2.4.9.4. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.10.SetSelectedRegistrar

This action is used to inform the proxy that the Registrar has been selected (eg the Registrar button has been pushed) & that the proxy indicates to Enrollees that a Registrar subscribed to its events has been selected. It is required for proxies. The input message is the SetSelectedRegistrar Message in the Simple Config specification.

2.4.10.1.Arguments

Table 3.6: Arguments for RebootAP

Argument	Direction	relatedStateVariable
NewMessage	IN	Message

2.4.10.2.Dependency on State (if any)

2.4.10.3.Effect on State (if any)

2.4.10.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.11.RebootAP

This action requests the Access Point to reboot and may optionally provide a new configuration instead of an existing configuration to be set prior to reboot. This command requires a previously established secure association between the Access Point and the Registrar. The AP must confirm the validity of the request by checking the hash of the message prior to performing a reboot. The input message is the ResetAP Message in the Simple Config specification.

2.4.11.1.Arguments

Table 3.6: Arguments for RebootAP

Argument	Direction	relatedStateVariable
NewAPSettings	IN	APSettings

2.4.11.2.Dependency on State (if any)

2.4.11.3.Effect on State (if any)

2.4.11.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.12.ResetAP

This action requests the Access Point to reset its configuration to factory settings. The APSetting variable is provided to authenticate the source of the request. . This action requires a previously established secure association between the Access Point and the Registrar. The AP must confirm the validity of the request by checking the hash of the message prior to performing a reboot. The input message is the ResetAP Message in the Simple Config specification. This action is required for APs.

2.4.12.1.Arguments

Table 3.6: Arguments for ResetAP

Argument	Direction	relatedStateVariable
NewMessage	IN	Message

2.4.12.2.Dependency on State (if any)

2.4.12.3.Effect on State (if any)

2.4.12.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.13.RebootSTA

This action requests the UPnP station to reboot and may optionally provide a new configuration instead of an existing configuration to be set prior to reboot. This action requires a previously established secure

association between the Station and the Registrar. The input message is the ResetAP Message in the Simple Config specification. The Station must confirm the validity of the request by checking the hash of the message prior to performing a reboot.

2.4.13.1.Arguments

Table 3.6: Arguments for RebootSTA

Argument	Direction	relatedStateVariable
NewSTASettings	IN	STASettings

2.4.13.2.Dependency on State (if any)

2.4.13.3.Effect on State (if any)

2.4.13.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.14.ResetSTA

This action requests the Station to reset its configuration to factory settings. The STASettings variable is provided to authenticate the source of the request. . This action requires a previously established secure association between the Station and the Registrar. The input message is the ResetAP Message in the Simple Config specification. The Station must confirm the validity of the request by checking the hash of the message prior to performing a reboot. This action is required for Stations that implement the WFAWLANConfig Service..

2.4.14.1.Arguments

Table 3.6: Arguments for ResetSTA

Argument	Direction	relatedStateVariable
NewMessage	IN	Message

2.4.14.2.Dependency on State (if any)

2.4.14.3.Effect on State (if any)

2.4.14.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

2.4.15.Non-Standard Actions Implemented by a UPnP Vendor

To facilitate certification, non-standard actions implemented by UPnP vendors should be included in this service template. The UPnP Device Architecture lists naming requirements for non-standard actions (see the section on Description).

2.4.16.Common Error Codes

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 4: Common Error Codes for all actions

ErrorCode	errorDescription	Description
401	Invalid Action	See UPnP Device Architecture section on Control.
402	Invalid Args	See UPnP Device Architecture section on Control.
404	Invalid Var	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
600-699	TBD	Common action errors. Defined by UPnP Forum Technical Committee.
701-799		Common action errors defined by the UPnP Forum working committees.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.5. Theory of Operation

2.5.1. Establishing a Registrar with an Access Point and AP management.

When an Access Point initializes on the network, it advertises the WFADevice and the WFAWLANConfig service to the UPnP network. A device detects the notification and retrieves the DeviceInfo (M1 message in the Registration protocol) to determine that its an AP that it wants to establish itself as a Registrar with and follows the same WFA Simple Config process, validating with the AP via a PIN (or password if set) over the PutMessage UPnP action . Once the secure association is established between the pair, the Registrar may add or remove station configurations or reconfigure the WLAN utilizing the TLV over UPnP methods defined herein, referencing the secure association via the Session ID (SID) utilizing the various AP configuration actions.

2.5.2. Proxy Function

The Proxy Function bridges the UPnP network and the UPnP network to forward notification of the existence of a device that wants to receive WLAN configuration. A Registrar receives the requests by subscribing to the UPnP event service provided by the Proxy Function and performs the WFA Simple Config exchange with the Enrollee over UPnP by using the WLANEvent/WLANEventType/WLANEventMAC event and the PutWLANResponse action.

The Proxy Function must cache Registrar responses to Probe-request events and to EAP message exchanges so that the Enrollee may get information serially for each Registrar that is utilizing the Proxy Function on a device. The 802.11 mechanisms are described in the WFA Simple Config specification.

2.5.3. Initialization and Configuration of the Ethernet connected wireless Device

When an Ethernet connected wireless device initializes on the network, it advertises WFADevice and the WFAWLANConfig service to the UPnP network. A device detects the notification and retrieves the DeviceInfo to determine that it's a wireless station that needs wireless settings over Ethernet and follows the WFA Simple Config process, validated the station with a PIN (or password if set) over the UPnP GetMessage and PutMessage actions. Once the secure association association is established between the pair, the device can be accessed directly via GetSTASettings and SetSTASettings at a later time if returned to the Ethernet network or optionally over the wireless network by referencing the Session ID (SID).

3. XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <actionList>
        <action>
            <name>GetDeviceInfo</name>
            <argumentList>
                <argument>
                    <name>NewDeviceInfo</name>
                    <direction>out</direction>
                  <relatedStateVariable>DeviceInfo</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>PutMessage</name>
            <argumentList>
                <argument>
                    <name>NewInMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>InMessage</relatedStateVariable>
                </argument>
                <argument>
                    <name>NewOutMessage</name>
                    <direction>out</direction>
                  <relatedStateVariable>OutMessage</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>GetAPSettings</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable>
                </argument>
                <argument>
                    <name>NewAPSettings</name>
                    <direction>out</direction>
                  <relatedStateVariable>APSettings</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
```

```
<action>
    <name>SetAPSettings</name>
    <argumentList>
        <argument>
            <name>APSettings</name>
            <direction>in</direction>
          <relatedStateVariable>APSettings</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>DelAPSettings</name>
    <argumentList>
        <argument>
            <name>NewAPSettings</name>
            <direction>in</direction>
          <relatedStateVariable>APSettings</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>GetSTASettings</name>
    <argumentList>
        <argument>
            <name>NewMessage</name>
            <direction>in</direction>
          <relatedStateVariable>Message</relatedStateVariable>
        </argument>
        <argument>
            <name>NewSTASettings</name>
            <direction>out</direction>
          <relatedStateVariable>STASettings</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>SetSTASettings</name>
    <argumentList>
        <argument>
            <name>NewSTASettings</name>
            <direction>out</direction>
          <relatedStateVariable>STASettings</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>DelSTASettings</name>
    <argumentList>
        <argument>
            <name>NewSTASettings</name>
            <direction>in</direction>
          <relatedStateVariable>STASettings</relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>PutWLANResponse</name>
    <argumentList>
        <argument>
            <name>NewMessage</name>
            <direction>in</direction>
          <relatedStateVariable>Message</relatedStateVariable>
        </argument>
        <argument>
            <name>NewWLANEventType</name>
            <direction>in</direction>
          <relatedStateVariable>WLANEventType</relatedStateVariable>
        </argument>
        <argument>
            <name>NewWLANEventMAC</name>
            <direction>in</direction>
          <relatedStateVariable>WLANEventMAC</relatedStateVariable>
        </argument>
    </argumentList>
```

```
        </action>
        <action>
            <name>SetSelectedRegistrar</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>RebootAP</name>
            <argumentList>
                <argument>
                    <name>NewAPSettings</name>
                    <direction>in</direction>
                  <relatedStateVariable>APSettings</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>ResetAP</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>RebootSTA</name>
            <argumentList>
                <argument>
                    <name>NewSTASettings</name>
                    <direction>in</direction>
                  <relatedStateVariable>APSettings</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>ResetSTA</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
    </actionList>
    <serviceStateTable>
        <stateVariable sendEvents="no">
            <name>Message</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>InMessage</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>OutMessage</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>DeviceInfo</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>APSettings</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
```

```
            <name>APStatus</name>
            <dataType>ui1</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>STASettings</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>STAStatus</name>
            <dataType>ui1</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>WLANEvent</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>WLANEventType</name>
            <dataType>ui1</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>WLANEventMAC</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>WLANResponse</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
    </serviceStateTable>
</scpd>
```

4. Test

No semantic tests have been defined for this service.