AR8x application note

This guide provides details about the JSON WCF RESTful service which allows you to monitor and configure the device

1. WEB SERVICe

The backend of the web config described in the previous chapters is a JSON WCF RESTful service, which can be accessed also from your own solutions. The nginx web server forwards the communication to the backend, so the same http authentication and https is used for backend communication as it is in the wb config UI. If we use the example address from the previous chapter, we can access these features by using HTTP GET and HTTP POST. So for example performing a simple “Ping”-request to the backend, we would simply send HTTP GET request to <https://ar8x567890.local/Core/Ping>, which would then respond with a JSON string containing “Pong”

1. SYSTEM
   1. INFO

System info can be requested with HTTP GET from:

[https://ar8x**567890**.local/Core/SysInfo](https://ar8x567890.local/Core/SysInfo)

SysInfo return value example:

{    
   **"AverageCpuUsage"**:"16.38,17.60,18.56",  
   **"CurrentCpuUsage"**:16.01,  
   **"CurrentMemoryUsage"**:67,  
   **"InstalledMemoryMB"**:434,  
   **"OSVersion"**:"Unix 4.4.10.0",  
   **"UpTimeSecs"**:85879  
}

CPU and memory usage history can requested with HTTP GET at:

[https://ar8x**567890**.local/Core/SysInfo/UsageHist](https://ar8x567890.local/Core/SysInfo/UsageHist)

UsageHist return value example:

{    
   **"CpuUsageHist"**:[    
      16.5121136,  
      17.00839,  
      17.00893  
   ],  
   **"MemoryUsageHist"**:[    
      16.58986,  
      16.8202763,  
      16.8202763  
   ]  
}

* 1. SYSTEM LOG

List of the available log files can retrieved with HTTP GET at:

<https://ar8x567890.local/Core/SysLog/List>

List returns a string array, JSON example below:

[    
   "auth.log",  
   "btmp",  
   "daemon.log",  
   "debug",  
 "kern.log",  
 "lastlog",  
 "mail.err",  
 "mail.info",  
 "mail.log",  
 "mail.warn",  
 "messages",  
 "syslog",  
 "syslog.1",  
 "user.log",  
 "wtmp"  
]

Individual log-file contents can be retrieved with HTTP POST at:

<https://ar8x567890.local/Core/SysLog/GetLog>

As a parameter you need need to pass a parameter where logname is one of the filenames found from the retrieved list. logpos indicates the byte position within the log i.e. 0 would indicate that the log needs to retrieved starting from the beginning of the file.

GetLog JSON parameter contents JSON example

{    
 **"logname"**:"user.log",  
 **"logpos"**:"0"  
}

GetLog JSON return value example:

{    
   **"LogText"**:"Jan  1 00:00:14 ar8x567890 NID-AUTOUPD: <info> Auto update: Nothing to do",  
   **"Position"**:306  
}

* 1. SYSTEM UPDATE

The system update features can be accessed with HTTP POST, requires the file stream to the zip-file containing the update as a parameter

[https://ar8x**567890**.local/Core/SysUpdate/Upload](https://ar8x567890.local/Core/SysUpdate/Upload)

Upload return value in JSON format:

{    
   **"error"**:"",  
   **"systemtype"**:"os",  
   **"version"**:"0.0.1"  
}

HTTP GET

[https://ar8x**567890**.local/Core/SysUpdate/Begin](https://ar8x567890.local/Core/SysUpdate/Begin)

Begin returns a single string to indicate the status:

"In progress"

[https://ar8x**567890**.local/Core/SysUpdate/Status](https://ar8x567890.local/Core/SysUpdate/Status)

Status returns a single string to indicate the status:

"In progress"

* 1. NETWORK
     1. LAN

LAN settings allow you to configure the TCP/IP settings for the eth0 adapter.

The settings can be retrieved using HTTP GET from:

[https://ar8x**567890**.local/Core/Network/Lan/GetProperties](https://ar8x567890.local/Core/Network/Lan/GetProperties)

The above request returns an array containing the settings formatted as the below object. The list contains also informational fields which are not used when changing the settings. You can separate the informational fields by checking the Name-field of each item, names ending with “.property” are meant to be used as settable values. So for example SettingContainer interfaces.eth0.dhcp.property –item is used for changing whether DHCP is enabled for the eth0-adapter.

Return value in JSON format:

[    
   {    
      **"Name"**:"interfaces.eth0.Adapter",  
      **"Options"**:null,  
      **"Title"**:"Adapter name",  
      **"Type"**:"Label",  
      **"Validation"**:0,  
      **"Value"**:"eth0"  
   },  
   {    
      **"Name"**:"interfaces.eth0.dhcp.property",  
      **"Options"**:[    
         {    
            **"Name"**:"DHCP",  
            **"Value"**:true  
         },  
         {    
            **"Name"**:"Static",  
            **"Value"**:false  
         }  
      ],  
      **"Title"**:"DHCP",  
      **"Type"**:"Radio",  
      **"Validation"**:0,  
      **"Value"**:true  
   },  
   {    
      **"Name"**:"interfaces.eth0.ipaddress.property",  
      **"Options"**:null,  
      **"Title"**:"IP address",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"interfaces.eth0.netmask.property",  
      **"Options"**:null,  
      **"Title"**:"Netmask address",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"interfaces.eth0.gateway.property",  
      **"Options"**:null,  
      **"Title"**:"Gateway address",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:null  
   }  
]

Changing the above settings can be achieved by performing a HTTP POST with an array containing the Name and the Value of the property.

HTTP POST to: [https://ar8x**567890**.local/Core/Network/Lan/SetProperties](https://ar8x567890.local/Core/Network/Lan/SetProperties)

The parameter in JSON format:

[    
   {    
      **"Name"**:"interfaces.eth0.dhcp.property",  
      **"Value"**:"true"  
   },  
   {    
      **"Name"**:"interfaces.eth0.ipaddress.property",  
      **"Value"**:""  
   },  
   {    
      **"Name"**:"interfaces.eth0.netmask.property",  
      **"Value"**:""  
   },  
   {    
      **"Name"**:"interfaces.eth0.gateway.property",  
      **"Value"**:""  
   }  
]

As you can see, we set each IP-address setting as empty and set the dhcp.property to true. However you don’t need to set each of the settings if you are for example just changing from static IP to dynamic setting. Then the dhcp.property setting would be enough. If the settings are considered valid by the backend, the eth0 adapter will be reloaded.

The backend can also be commanded to reload the adapter properties by sending HTTP GET request to:

[https://ar8x**567890**.local/Core/Network/Lan/ReloadProperties](https://ar8x567890.local/Core/Network/Lan/ReloadProperties)

The operation returns a boolean value in JSON format. True for success and false for failure.

Each individual value shown in the above “GetProperties”-method can be also set separately using the “SetProperty”-method found from the backend service.

Setting each value can be done using the same Property-object as when setting multiple properties, but now you would just send a single Property as a parameter within HTTP POST.

https://ar8x**567890**.local/Core/Network/Lan/SetProperty

So setting one individual setting would look like this in JSON format:

{    
 **"Name"**:"interfaces.eth0.dhcp.property",  
 **"Value"**:"true"  
}

And each property-value can be retrieved by sending the name of the property as a parameter to the backend service with HTTP POST.

[https://ar8x**567890**.local/Core/Network/Lan/GetProperty](https://ar8x567890.local/Core/Network/Lan/GetProperty)

The parameter in JSON format:

{    
 **"name"**:"interfaces.eth0.dhcp.property",  
}

Return value would then be in JSON format as follows:

{    
 **"value"**:"true",  
}

* + 1. WLAN

The basic features for WLAN settings are identical to what they are for the ethernet-adapter, so please see the previous chapter for the documentation for these:

[https://ar8x**567890**.local/Core/Network/Wlan/SetProperties](https://ar8x567890.local/Core/Network/Wlan/SetProperties)

[https://ar8x**567890**.local/Core/Network/Wlan/ReloadProperties](https://ar8x567890.local/Core/Network/Wlan/ReloadProperties)

[https://ar8x**567890**.local/Core/Network/Wlan/SetProperty](https://ar8x567890.local/Core/Network/Wlan/SetProperty)

[https://ar8x**567890**.local/Core/Network/Wlan/GetProperty](https://ar8x567890.local/Core/Network/Wlan/GetProperty)

The usage for GetProperties is also the same as above, but below you can find an JSON formatted example of using:

[https://ar8x**567890**.local/Core/Network/Lan/GetProperties](https://ar8x567890.local/Core/Network/Lan/GetProperties)

Note that the TCP/IP-settings are exactly the same as for the eth0-adapter. The difference comes on the WPA-supplicant settings which extend the properties to list individual SSID-settings.

[    
   {    
      **"Name"**:"interfaces.wlan0.Adapter",  
      **"Options"**:null,  
      **"Title"**:"Adapter name",  
      **"Type"**:"Label",  
      **"Validation"**:0,  
      **"Value"**:"wlan0"  
   },  
   {    
      **"Name"**:"interfaces.wlan0.dhcp.property",  
      **"Options"**:[    
         {    
            **"Name"**:"DHCP",  
            **"Value"**:true  
         },  
         {    
            **"Name"**:"Static",  
            **"Value"**:false  
         }  
      ],  
      **"Title"**:"DHCP",  
      **"Type"**:"Radio",  
      **"Validation"**:0,  
      **"Value"**:true  
   },  
   {    
      **"Name"**:"interfaces.wlan0.ipaddress.property",  
      **"Options"**:null,  
      **"Title"**:"IP address",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"interfaces.wlan0.netmask.property",  
      **"Options"**:null,  
      **"Title"**:"Netmask address",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"interfaces.wlan0.gateway.property",  
      **"Options"**:null,  
      **"Title"**:"Gateway address",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"wlan.profile.title",  
      **"Options"**:null,  
      **"Title"**:"WLAN profiles ",  
      **"Type"**:"Label",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"wlan.profile.0.title.Title",  
      **"Options"**:null,  
      **"Title"**:"Network 0",  
      **"Type"**:"Label",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"wlan.profile.0.ssid.property",  
      **"Options"**:null,  
      **"Title"**:"SSID",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:"1234"  
   },  
   {    
      **"Name"**:"wlan.profile.0.key\_mgmt.property",  
      **"Options"**:[    
         {    
            **"Name"**:"WEP\/OPEN",  
            **"Value"**:"NONE"  
         },  
         {    
            **"Name"**:"WPA",  
            **"Value"**:"WPA-PSK"  
         },  
         {    
            **"Name"**:"WPA2",  
            **"Value"**:"WPA2-PSK"  
         }  
      ],  
      **"Title"**:"Security type",  
      **"Type"**:"Select",  
      **"Validation"**:0,  
      **"Value"**:"WPA2-PSK"  
   },  
   {    
      **"Name"**:"wlan.profile.0.psk.property",  
      **"Options"**:null,  
      **"Title"**:"Security key",  
      **"Type"**:"Password",  
      **"Validation"**:0,  
      **"Value"**:"123"  
   },  
   {    
      **"Name"**:"wlan.profile.0.scan\_ssid.property",  
      **"Options"**:[    
         {    
            **"Name"**:"checked",  
            **"Value"**:1  
         },  
         {    
            **"Name"**:"unchecked",  
            **"Value"**:0  
         }  
      ],  
      **"Title"**:"Hidden network",  
      **"Type"**:"Check",  
      **"Validation"**:0,  
      **"Value"**:"0"  
   },  
   {    
      **"Name"**:"wlan.profile.1.title.Title",  
      **"Options"**:null,  
      **"Title"**:"Network 1",  
      **"Type"**:"Label",  
      **"Validation"**:0,  
      **"Value"**:null  
   },  
   {    
      **"Name"**:"wlan.profile.1.ssid.property",  
      **"Options"**:null,  
      **"Title"**:"SSID",  
      **"Type"**:"Text",  
      **"Validation"**:0,  
      **"Value"**:"SSID2"  
   },  
   {    
      **"Name"**:"wlan.profile.1.key\_mgmt.property",  
      **"Options"**:[    
         {    
            **"Name"**:"WEP\/OPEN",  
            **"Value"**:"NONE"  
         },  
         {    
            **"Name"**:"WPA",  
            **"Value"**:"WPA-PSK"  
         },  
         {    
            **"Name"**:"WPA2",  
            **"Value"**:"WPA2-PSK"  
         }  
      ],  
      **"Title"**:"Security type",  
      **"Type"**:"Select",  
      **"Validation"**:0,  
      **"Value"**:"NONE"  
   },  
   {    
      **"Name"**:"wlan.profile.1.psk.property",  
      **"Options"**:null,  
      **"Title"**:"Security key",  
      **"Type"**:"Password",  
      **"Validation"**:0,  
      **"Value"**:"123"  
   },  
   {    
      **"Name"**:"wlan.profile.1.scan\_ssid.property",  
      **"Options"**:[    
         {    
            **"Name"**:"checked",  
            **"Value"**:1  
         },  
         {    
            **"Name"**:"unchecked",  
            **"Value"**:0  
         }  
      ],  
      **"Title"**:"Hidden network",  
      **"Type"**:"Check",  
      **"Validation"**:0,  
      **"Value"**:"0"  
   }  
]

The supplicant settings also allow you to directly modify the network settings i.e. to add/remove network profiles and request the details of visible SSIDs nearby. These 3 methods can be accessed with

[https://ar8x**567890**.local/Core/Network/Wlan/CreateProfile](https://ar8x567890.local/Core/Network/Wlan/CreateProfile)

To create a new profile with the above method, you need to pass a JSON string similar to the example below:

[    
   {    
      **"Name"**:"wlan.profile.new.ssid.property",  
      **"Value"**:"myssid123"  
   },  
   {    
      **"Name"**:"wlan.profile.new.key\_mgmt.property",  
      **"Value"**:"WPA2-PSK"  
   },  
   {    
      **"Name"**:"wlan.profile.new.psk.property",  
      **"Value"**:"password"  
   },  
   {    
      **"Name"**:"wlan.profile.new.scan\_ssid.property",  
      **"Value"**:0  
   }  
]

HTTP POST [https://ar8x**567890**.local/Core/Network/Wlan/RemoveProfile](https://ar8x567890.local/Core/Network/Wlan/RemoveProfile)

The parameter for the RemoveProfile is a single integer value which is the index for the particular profile which can be found from each individual network profile setting. For example from the SSID-setting for network profile 1, we can find it as the 3rd field(separated by “.”):

wlan.profile.1.scan\_ssid.property

So if we would want to remove this profile, we would simply send 1 in JSON format to the method.

The request the available networks, you can use the method below:

[https://ar8x**567890**.local/Core/Network/Wlan/GetAvailableNetworks](https://ar8x567890.local/Core/Network/Wlan/GetAvailableNetworks)

The return value is an array containing 0..n NetworkInformation-objects, example in JSON format:

[    
   {    
      **"ssid"**:"MyVisibleSSID1",  
      **"bssid"**:"00.11.22.33.44.55",  
      **"key\_mgmt"**:"WPA2-PSK",  
      **"signallevel"**:"-49",  
   },  
   {    
      **"ssid"**:"MyVisibleSSID2",  
      **"bssid"**:"66.77.88.99.00.11",  
      **"key\_mgmt"**:"WPA2-PSK",  
      **"signallevel"**:"-64",  
   },  
]

* 1. ACCESS

The access feature on the web service allows you to add, remove and list the users created to .htpasswd.

[https://ar8x**567890**.local/Core/UserManagement/AddUser](https://ar8x567890.local/Core/UserManagement/AddUser)

[https://ar8x**567890**.local/Core/UserManagement/RemoveUser](https://ar8x567890.local/Core/UserManagement/RemoveUser)

[https://ar8x**567890**.local/Core/UserManagement/GetUsers](https://ar8x567890.local/Core/UserManagement/GetUsers)

For both AddUser and RemoveUser you need to send HTTP POST request with a parameter containing a object similar to below:

{    
 **"username"**:"username",  
 **"password"**:"pwd",  
}

The return value is a UserMgmtResponse containing the results as a boolean value and a error string in case such occurred. Result in JSON format:

{    
 **"result"**:true,   
 **"error"**:"",  
}

List of users can be retrieved with HTTP GET request using GetUsers-method. The return value in JSON format is as follows:

[    
   {    
      **"username"**:"username1",  
      **"password"**:"pwd1"  
   },  
   {    
      **"username"**:"username2",  
      **"password"**:"pwd2"  
   }  
]

* 1. TIME SERVICES

Time services allows you to change the NTP settings and to change the system & RTC time setting.

[https://ar8x**567890**.local/Core/SysTime/GetTime](https://ar8x567890.local/Core/SysTime/GetTime)

Sending a HTTP GET request to the address above will return the current time information from the platform in the following format:

{    
 **"date"**:"2016-11-14",  
 **"epoch"**:"1479115388",  
 **"error"**:"",  
 **"sysdate"**:"Mon Nov 14 09:23:09 UTC 2016",  
 **"time"**:"09:23:08"  
}

The same TimeContainer-parameter is to be used when setting the system time, however only the epoch-value of it will be used. So HTTP POST:

[https://ar8x**567890**.local/Core/SysTime/SetTime](https://ar8x567890.local/Core/SysTime/SetTime)

SetTime return value is in the below below format:

{    
 **"success"**:true,  
 **"error"**:null  
}

GetNTPSettings lists you the current NTP servers used and whether NTP is enabled. You can request the settings with HTTP GET from:

[https://ar8x**567890**.local/Core/SysTime/GetNTPSettings](https://ar8x567890.local/Core/SysTime/GetNTPSettings)

Return value xample in JSON format:

{    
   **"NTPAddresses"**:[    
      "0.debian.pool.ntp.org",  
      "1.debian.pool.ntp.org",  
      "2.debian.pool.ntp.org",  
      "3.debian.pool.ntp.org"  
   ],  
   **"NTPEnabled"**:true,  
   **"error"**:null  
}

To set the NTP settings you can send the the above NTPSettings-object as a paramer to SetNTPSettings:

[https://ar8x**567890**.local/Core/SysTime/SetNTPSettings](https://ar8x567890.local/Core/SysTime/SetNTPSettings)

Return value of SetNTPSettings is the same as for SetTime, i.e. an object containing a boolean to indicate whether the operation succeeded and an error string to describe the error if such occurred.

1. APPLICATIONS

The application page allows you to manage and update the applications running on the platform.

To upload an application, use the below HTTP POST request. The parameter for the method is the filestream to the zip-file to be uploaded.

[https://ar8x**567890**.local/Core/AppManagement/Upload](https://ar8x567890.local/Core/AppManagement/Upload)

In return value:

{    
   **"error"**:"",  
   **"manifest"**:"ar8xapplication↵appexample↵1.0.0",  
   **"configuration"**:"",  
   **"version"**:"1.0.0"  
}

When the previous method Upload has finished, you’ll need to inform the backend to begin the installation process. This can be done with HTTP GET request to:

[https://ar8x**567890**.local/Core/AppManagement/Begin](https://ar8x567890.local/Core/AppManagement/Begin)

The return value is a string containing the status of the installation, i.e. “Installing” or “FAIL: File not found” in case of an error.

The status of the install process can be requested with HTTP GET using the below path:

[https://ar8x**567890**.local/Core/AppManagement/Status](https://ar8x567890.local/Core/AppManagement/Status)

As with begin, the return value string containing the status. When the install has finished without any errors, the result will be “Idle”.

To get a list of all the currently installed applications and their status, use GetApplications with HTTP GET:

[https://ar8x**567890**.local/Core/AppManagement/GetApplications](https://ar8x567890.local/Core/AppManagement/GetApplications)

The return value in JSON format:

[    
   {    
      **"backendActive"**:true,  
      **"backendAddr"**:"\/app\/backend\/appexample",  
      **"backendPort"**:10001,  
      **"isManaged"**:true,  
      **"isSystemApp"**:false,  
      **"name"**:"appexample",  
      **"procInfo"**:[    
         {    
            **"command"**:"\/bin\/sh \/usr\/data\/apps\/appexample\/bin\/run start ",  
            **"processId"**:28051,  
            **"startTime"**:"07:23"  
         },  
         {    
            **"command"**:"mono \/usr\/data\/apps\/appexample\/bin\/ThirdPartyAppPing.exe ",  
            **"processId"**:28053,  
            **"startTime"**:"07:23"  
         }  
      ],  
      **"version"**:"1.0.0"  
   },  
   {    
      **"backendActive"**:true,  
      **"backendAddr"**:"\/app\/backend\/appexample2",  
      **"backendPort"**:10123,  
      **"isManaged"**:true,  
      **"isSystemApp"**:false,  
      **"name"**:"appexample2",  
      **"procInfo"**:[    
  
      ],  
      **"version"**:"1.0.0"  
   }  
]

The previous JSON example tells us that there’s 2 different applications installed, appexample and appexample2. The first has 2 different processes running: a mono application called ThirdPartyAppPing.exe and the startup script. The script is still active since the mono process was not started to the background, so it is blocked until the ThirdPartyAppPing.exe exits.

Applications can be started and stopped using the 2 methods below with HTTP POST.

[https://ar8x**567890**.local/Core/AppManagement/StartApplication](https://ar8x567890.local/Core/AppManagement/StartApplication)

[https://ar8x**567890**.local/Core/AppManagement/StopApplication](https://ar8x567890.local/Core/AppManagement/StopApplication)

Both take the ApplicationStartStopInfo-object as parameter and return a boolean value to indicate whether the process was started successfully.

So to start or stop “appexample”-applicatio, you would need to pass the name in the below format to the service:

{    
 **"name"**:"appexample"   
}

If the “run”-script in your application zip does not implement the “Stop”-clause properly i.e. your applications are not signaled to stop, you can kill the processes manually. However it is recommended that the stop-signalling is implemented so that each application gets closed gracefully. Process killing can be achieved by sending a HTTP POST request to:

[https://ar8x**567890**.local/Core/AppManagement/KillProcess](https://ar8x567890.local/Core/AppManagement/KillProcess)

As the parameter you need to send:

{    
 **"appName"**:"appexample",  
 **"processId"**:1234  
}

Applications can be deleted by sending a HTTP POST request to:

[https://ar8x**567890**.local/Core/AppManagement/DeleteApplication](https://ar8x567890.local/Core/AppManagement/DeleteApplication)

As a parameter you need to send the same content as you would when starting or stopping an application.