



TECHNICAL REFERENCE MANUAL

SDK API Reference

v26.0.1

Jump to a language:

● Go

● Java

● Python

Go API Reference

Orbit OS SDK · Version 26.0.1 · Generated 2026-09-18

CONTENTS

Auth Manager	2	Camera Manager	48
System Manager	3	AI Manager	52
Package Manager	18	Firewall Manager	58
Ethernet Manager	20	VPN Manager	64
WiFi Manager	24	App Hub Manager	70
GPIO Manager	32	Event Manager	75
PWM Manager	35	Development Manager	76
UART Manager	37	Update Manager	78
I2C Manager	42	Power Manager	79
SPI Manager	45	Types	80

Auth Manager

Authentication and session management.

Login Authenticates a user and returns a session token.

SIGNATURE

```
func (m *AuthManager) Login(username, password string) (token string, expiresAt int64, err error)
```

PARAMETERS

NAME	TYPE	NOTES
username	string	
password	string	

RETURNS

NAME	TYPE	NOTES
token	string	
expiresAt	int64	Unix timestamp
error	error	

EXAMPLE

```
token, expiresAt, err := client.AuthManager.Login(username, password)
```

GO

Logout Invalidates a session token.

SIGNATURE

```
func (m *AuthManager) Logout(token string) error
```

PARAMETERS

NAME	TYPE	NOTES
token	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AuthManager.Logout(token)
```

System Manager

Device identity, hardware info, OS metadata and live metrics.

Permission: `SystemService/*`

GetApiVersion Returns the API version and revision running on the device.

SIGNATURE

```
func (s *SystemManager) GetApiVersion() (version int64, revision int64, err error)
```

RETURNS

NAME	TYPE	NOTES
version	int64	
revision	int64	
error	error	

EXAMPLE

```
version, revision, err := client.SystemManager.GetApiVersion()
```

GetApiVersionInfo Returns the API version as a formatted string.

SIGNATURE

```
func (s *SystemManager) GetApiVersionInfo() (string, error)
```

RETURNS

NAME	TYPE	NOTES
version	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetApiVersionInfo()
```

GetDeviceName Returns the configured device name.

SIGNATURE

```
func (s *SystemManager) GetDeviceName() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
name, err := client.SystemManager.GetDeviceName()
```

GetSocModel Returns the SoC model identifier.

SIGNATURE

```
func (s *SystemManager) GetSocModel() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
model, err := client.SystemManager.GetSocModel()
```

GetSocVendor Returns the SoC vendor name.

SIGNATURE

```
func (s *SystemManager) GetSocVendor() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
vendor, err := client.SystemManager.GetSocVendor()
```

GetBoardModel

Returns the board model identifier.

SIGNATURE

```
func (s *SystemManager) GetBoardModel() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
model, err := client.SystemManager.GetBoardModel()
```

GetBoardVendor

Returns the board vendor name.

SIGNATURE

```
func (s *SystemManager) GetBoardVendor() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
vendor, err := client.SystemManager.GetBoardVendor()
```

GetHardwareVersion

Returns the hardware revision string.

SIGNATURE

```
func (s *SystemManager) GetHardwareVersion() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetHardwareVersion()
```

GetHardwareModel Returns the hardware model identifier.

SIGNATURE

```
func (s *SystemManager) GetHardwareModel() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
model, err := client.SystemManager.GetHardwareModel()
```

GetSystemUuid Returns the system UUID.

SIGNATURE

```
func (s *SystemManager) GetSystemUuid() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
uuid, err := client.SystemManager.GetSystemUuid()
```

GetBoardSerial Returns the board serial number.

SIGNATURE

```
func (s *SystemManager) GetBoardSerial() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
serial, err := client.SystemManager.GetBoardSerial()
```

GetCpuSerial

Returns the CPU serial number.

SIGNATURE

```
func (s *SystemManager) GetCpuSerial() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
serial, err := client.SystemManager.GetCpuSerial()
```

GetMachineId

Returns the machine ID.

SIGNATURE

```
func (s *SystemManager) GetMachineId() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
id, err := client.SystemManager.GetMachineId()
```

GetArchitecture

Returns the CPU architecture (e.g. "arm64", "amd64").

SIGNATURE

```
func (s *SystemManager) GetArchitecture() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
arch, err := client.SystemManager.GetArchitecture()
```

GetTotalRAM Returns total RAM in bytes.

SIGNATURE

```
func (s *SystemManager) GetTotalRAM() (uint64, error)
```

RETURNS

NAME	TYPE	NOTES
value	uint64	
error	error	

EXAMPLE

```
ram, err := client.SystemManager.GetTotalRAM()
```

GetCpuModel Returns the CPU model string.

SIGNATURE

```
func (s *SystemManager) GetCpuModel() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
model, err := client.SystemManager.GetCpuModel()
```

GetCpuCores Returns the number of physical CPU cores.

SIGNATURE

```
func (s *SystemManager) GetCpuCores() (int64, error)
```

RETURNS

NAME	TYPE	NOTES
value	int64	
error	error	

EXAMPLE

```
cores, err := client.SystemManager.GetCpuCores()
```

GetCpuThreads Returns the number of logical CPU threads.

SIGNATURE

```
func (s *SystemManager) GetCpuThreads() (int64, error)
```

RETURNS

NAME	TYPE	NOTES
value	int64	
error	error	

EXAMPLE

```
threads, err := client.SystemManager.GetCpuThreads()
```

GetCpuMinMhz Returns the minimum CPU frequency in MHz.

SIGNATURE

```
func (s *SystemManager) GetCpuMinMhz() (float64, error)
```

RETURNS

NAME	TYPE	NOTES
value	float64	
error	error	

EXAMPLE

```
minMhz, err := client.SystemManager.GetCpuMinMhz()
```

GetCpuMaxMhz Returns the maximum CPU frequency in MHz.

SIGNATURE

```
func (s *SystemManager) GetCpuMaxMhz() (float64, error)
```

RETURNS

NAME	TYPE	NOTES
value	float64	
error	error	

EXAMPLE

```
maxMhz, err := client.SystemManager.GetCpuMaxMhz()
```

GetOsName

Returns the OS name.

SIGNATURE

```
func (s *SystemManager) GetOsName() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
name, err := client.SystemManager.GetOsName()
```

GetOsVersion

Returns the OS version string.

SIGNATURE

```
func (s *SystemManager) GetOsVersion() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetOsVersion()
```

GetKernelVersion

Returns the kernel version string.

SIGNATURE

```
func (s *SystemManager) GetKernelVersion() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetKernelVersion()
```

GetDistro Returns the Linux distribution name.

SIGNATURE

```
func (s *SystemManager) GetDistro() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
distro, err := client.SystemManager.GetDistro()
```

GetDistroVersion Returns the distribution version string.

SIGNATURE

```
func (s *SystemManager) GetDistroVersion() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetDistroVersion()
```

GetRuntimeVersion Returns the Orbit runtime version.

SIGNATURE

```
func (s *SystemManager) GetRuntimeVersion() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetRuntimeVersion()
```

GetOsRevision Returns the logical OS revision string.

SIGNATURE

```
func (s *SystemManager) GetOsRevision() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
rev, err := client.SystemManager.GetOsRevision()
```

GetBuildVersion Alias for GetOsRevision.

SIGNATURE

```
func (s *SystemManager) GetBuildVersion() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
ver, err := client.SystemManager.GetBuildVersion()
```

GetRuntimeBuildDate Returns the runtime image build date.

SIGNATURE

```
func (s *SystemManager) GetRuntimeBuildDate() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
date, err := client.SystemManager.GetRuntimeBuildDate()
```

GetBuildDate

 Alias for GetRuntimeBuildDate.

SIGNATURE

```
func (s *SystemManager) GetBuildDate() (string, error)
```

RETURNS

NAME	TYPE	NOTES
value	string	
error	error	

EXAMPLE

```
date, err := client.SystemManager.GetBuildDate()
```

GetMetrics

 Returns live CPU usage, RAM and storage stats.

SIGNATURE

```
func (s *SystemManager) GetMetrics() (*MetricsInfoResponse, error)
```

RETURNS

NAME	TYPE	NOTES
*MetricsInfoResponse		
error	error	

EXAMPLE

```
metrics, err := client.SystemManager.GetMetrics()
```

Attach

 Attaches the SDK client to the running Gravity system service.

SIGNATURE

```
func (s *SystemManager) Attach() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.Attach()
```

EnableDevMode Enables developer mode on the device.

SIGNATURE

```
func (s *SystemManager) EnableDevMode() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.EnableDevMode()
```

DisableDevMode Disables developer mode on the device.

SIGNATURE

```
func (s *SystemManager) DisableDevMode() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.DisableDevMode()
```

IsDevModeEnabled Reports whether developer mode is currently enabled.

SIGNATURE

```
func (s *SystemManager) IsDevModeEnabled() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.IsDevModeEnabled()
```

EnableSSHServer Enables the SSH server on the device.

SIGNATURE

```
func (s *SystemManager) EnableSSHServer() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.EnableSSHServer()
```

DisableSSHServer Disables the SSH server on the device.

SIGNATURE

```
func (s *SystemManager) DisableSSHServer() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.DisableSSHServer()
```

IsSSHServerEnabled Reports whether the SSH server is currently enabled.

SIGNATURE

```
func (s *SystemManager) IsSSHServerEnabled() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.IsSSHServerEnabled()
```

EnableRebootOnFailure Enables automatic reboot when a critical failure is detected.

SIGNATURE

```
func (s *SystemManager) EnableRebootOnFailure() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.EnableRebootOnFailure()
```

DisableRebootOnFailure Disables automatic reboot on failure.

SIGNATURE

```
func (s *SystemManager) DisableRebootOnFailure() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.DisableRebootOnFailure()
```

IsRebootOnFailureEnabled Reports whether automatic reboot on failure is enabled.

SIGNATURE

```
func (s *SystemManager) IsRebootOnFailureEnabled() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.IsRebootOnFailureEnabled()
```

AllowUntrustedApps Allows installation of apps from untrusted sources.

SIGNATURE

```
func (s *SystemManager) AllowUntrustedApps() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.AllowUntrustedApps()
```

DisallowUntrustedApps Blocks installation of apps from untrusted sources.

SIGNATURE

```
func (s *SystemManager) DisallowUntrustedApps() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.DisallowUntrustedApps()
```

IsUntrustedAppsAllowed Reports whether untrusted app installation is currently allowed.

SIGNATURE

```
func (s *SystemManager) IsUntrustedAppsAllowed() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
allowed	bool	
error	error	

EXAMPLE

```
ok, err := client.SystemManager.IsUntrustedAppsAllowed()
```

Package Manager

Application package installation and management.

Permission: PackageService/*

ListInstalledPackages Returns all installed packages.

SIGNATURE

```
func (p *PackageManager) ListInstalledPackages() ([]*InstalledPackage, error)
```

RETURNS

NAME	TYPE	NOTES
[]InstalledPackage		
error	error	

EXAMPLE

```
pkgs, err := client.PackageManager.ListInstalledPackages()
```

InstallPackage Uploads and installs an .orb package from a local path.

SIGNATURE

```
func (p *PackageManager) InstallPackage(ctx context.Context, orbPath string) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
orbPath	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.PackageManager.InstallPackage(ctx, "/path/to/app.orb")
```

RemovePackage

Uninstalls a package by ID.

SIGNATURE

```
func (p *PackageManager) RemovePackage(ctx context.Context, packageID string) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
packageID	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.PackageManager.RemovePackage(ctx, packageID)
```

Ethernet Manager

Ethernet interface configuration and status.

Permission: EthernetService/*

ListEthernetInterfaces Returns all Ethernet interfaces available on the device.

SIGNATURE

```
func (e *EthernetManager) ListEthernetInterfaces() ([]*EthernetLinkProperties, error)
```

RETURNS

NAME	TYPE	NOTES
[]*EthernetLinkProperties		
error	error	

EXAMPLE

```
ifaces, err := client.EthernetManager.ListEthernetInterfaces()
```

IsEthernetConnected Reports whether the given interface has link.

SIGNATURE

```
func (e *EthernetManager) IsEthernetConnected(interfaceName string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
interfaceName	string	

RETURNS

NAME	TYPE	NOTES
connected	bool	
error	error	

EXAMPLE

```
ok, err := client.EthernetManager.IsEthernetConnected("eth0")
```

GetEthernetLinkProperties

Returns link properties (MAC, state, MTU, IP config) for an interface.

SIGNATURE

```
func (e *EthernetManager) GetEthernetLinkProperties(interfaceName string)
(*EthernetLinkProperties, error)
```

PARAMETERS

NAME	TYPE	NOTES
interfaceName	string	

RETURNS

NAME	TYPE	NOTES
*EthernetLinkProperties		
error	error	

EXAMPLE

```
props, err := client.EthernetManager.GetEthernetLinkProperties("eth0")
```

SetEthernetConfig Applies static IP or DHCP settings to an interface.

SIGNATURE

```
func (e *EthernetManager) SetEthernetConfig(  
    interfaceName string, enable, dhcpEnable bool,  
    ipv4Address, ipv4Gateway string, ipv4Dns []string,  
) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
interfaceName	string	
enable	bool	
dhcpEnable	bool	
ipv4Address	string	CIDR notation — ignored when dhcpEnable is true
ipv4Gateway	string	
ipv4Dns	[]string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.EthernetManager.SetEthernetConfig(  
    "eth0", true, false,  
    "192.168.1.10/24", "192.168.1.1", []string{"8.8.8.8"},  
)
```

EnableEthernet Brings an Ethernet interface up.

SIGNATURE

```
func (e *EthernetManager) EnableEthernet(interfaceName string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
interfaceName	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.EthernetManager.EnableEthernet("eth0")
```

DisableEthernet Brings an Ethernet interface down.

SIGNATURE

```
func (e *EthernetManager) DisableEthernet(interfaceName string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
interfaceName	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.EthernetManager.DisableEthernet("eth0")
```

WiFi Manager

Wi-Fi interface management — client mode and access point.

Permission: `WiFiService/*`

ListInterfaces Returns all Wi-Fi interfaces on the device.

SIGNATURE

```
func (w *WiFiManager) ListInterfaces() ([]*WiFiLinkProperties, error)
```

RETURNS

NAME	TYPE	NOTES
<code>[]*WiFiLinkProperties</code>		
<code>error</code>	<code>error</code>	

EXAMPLE

```
ifaces, err := client.WiFiManager.ListInterfaces()
```

GO

GetLinkProperties Returns link-layer and mode properties for a Wi-Fi interface.

SIGNATURE

```
func (w *WiFiManager) GetLinkProperties(ifname string) (*WiFiLinkProperties, error)
```

PARAMETERS

NAME	TYPE	NOTES
<code>ifname</code>	<code>string</code>	

RETURNS

NAME	TYPE	NOTES
<code>*WiFiLinkProperties</code>		
<code>error</code>	<code>error</code>	

EXAMPLE

```
props, err := client.WiFiManager.GetLinkProperties("wlan0")
```

IsConnected Returns true if the interface has an active Wi-Fi connection.

SIGNATURE

```
func (w *WiFiManager) IsConnected(ifname string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
connected	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.IsConnected("wlan0")
```

GetMode Returns the current operating mode of the interface.

SIGNATURE

```
func (w *WiFiManager) GetMode(ifname string) (WiFiMode, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
mode	WiFiMode	
error	error	

EXAMPLE

```
mode, err := client.WiFiManager.GetMode("wlan0")
```

SetModeClient Switches the interface to client (station) mode.

SIGNATURE

```
func (w *WiFiManager) SetModeClient(ifname string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.SetModeClient("wlan0")
```

SetClientConfig

Configures and stores Wi-Fi client settings for the interface.

SIGNATURE

```
func (w *WiFiManager) SetClientConfig(
    ifname, ssid, password, security string,
    dhcpEnable bool,
    ipv4Address, ipv4Gateway string,
    ipv4Dns []string,
) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	
ssid	string	
password	string	
security	string	"none" "wpa2" "wpa3" "wpa2-wpa3"
dhcpEnable	bool	
ipv4Address	string	CIDR — ignored when dhcpEnable is true
ipv4Gateway	string	
ipv4Dns	[]string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.SetClientConfig(
    "wlan0", "MySSID", "pass", "wpa2",
    true, "", "", nil,
)
```

GetClientProperties Returns IP and radio properties for the current client connection.

SIGNATURE

```
func (w *WiFiManager) GetClientProperties(ifname string) (*ClientProperties, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
*ClientProperties		
error	error	

EXAMPLE

```
props, err := client.WiFiManager.GetClientProperties("wlan0")
```

Connect Brings up the Wi-Fi connection using the previously stored client config.

SIGNATURE

```
func (w *WiFiManager) Connect(ifname string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.Connect("wlan0")
```

Disconnect Disconnects from the current Wi-Fi network.

SIGNATURE

```
func (w *WiFiManager) Disconnect(ifname string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.Disconnect("wlan0")
```

StartAP Configures and starts an access point on the given interface.

SIGNATURE

```
func (w *WiFiManager) StartAP(iframe, ssid, password, band string, channel int32) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
iframe	string	
ssid	string	
password	string	
band	string	"2.4GHz" or "5GHz"
channel	int32	0 = auto

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.StartAP("wlan0", "MyAP", "pass", "5GHz", 0)
```

StopAP Stops the access point on the given interface.

SIGNATURE

```
func (w *WiFiManager) StopAP(iframe string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
iframe	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.WiFiManager.StopAP("wlan0")
```

GetAPPProperties Returns properties of the active access point.

SIGNATURE

```
func (w *WiFiManager) GetAPPProperties(ifname string) (*APPProperties, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	

RETURNS

NAME	TYPE	NOTES
*APPProperties		
error	error	

EXAMPLE

```
props, err := client.WiFiManager.GetAPPProperties("wlan0")
```

Scan Scans for nearby Wi-Fi networks. forceRescan=true triggers a new hardware scan (~3 s).

SIGNATURE

```
func (w *WiFiManager) Scan(ifname string, forceRescan bool) ([]*ScannedNetwork, error)
```

PARAMETERS

NAME	TYPE	NOTES
ifname	string	
forceRescan	bool	

RETURNS

NAME	TYPE	NOTES
[]*ScannedNetwork		
error	error	

EXAMPLE

```
nets, err := client.WiFiManager.Scan("wlan0", true)
```

GPIO Manager

General-purpose I/O pin control.

Permission: `GpioService/*`

ListPins Returns all GPIO pins available on the device.

SIGNATURE

```
func (m *GpioManager) ListPins() ([]*GpioPin, error)
```

RETURNS

NAME	TYPE	NOTES
<code>[]GpioPin</code>		
<code>error</code>	<code>error</code>	

EXAMPLE

```
pins, err := client.GpioManager.ListPins()
```

GetDirection Reads the current direction of a pin.

SIGNATURE

```
func (m *GpioManager) GetDirection(pin *GpioPin) (GpioDirection, error)
```

PARAMETERS

NAME	TYPE	NOTES
<code>pin</code>	<code>*GpioPin</code>	

RETURNS

NAME	TYPE	NOTES
<code>dir</code>	<code>GpioDirection</code>	
<code>error</code>	<code>error</code>	

EXAMPLE

```
dir, err := client.GpioManager.GetDirection(pin)
```

SetDirection Configures a pin as input or output.

SIGNATURE

```
func (m *GpioManager) SetDirection(pin *GpioPin, dir GpioDirection) error
```

PARAMETERS

NAME	TYPE	NOTES
pin	*GpioPin	
dir	GpioDirection	GPIO_DIR_IN GPIO_DIR_OUT

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.GpioManager.SetDirection(pin, GPIO_DIR_OUT)
```

GetLevel Reads the current logic level of a pin.

SIGNATURE

```
func (m *GpioManager) GetLevel(pin *GpioPin) (GpioLevel, error)
```

PARAMETERS

NAME	TYPE	NOTES
pin	*GpioPin	

RETURNS

NAME	TYPE	NOTES
level	GpioLevel	
error	error	

EXAMPLE

```
level, err := client.GpioManager.GetLevel(pin)
```

SetLevel Sets the logic level of an output pin.

SIGNATURE

```
func (m *GpioManager) SetLevel(pin *GpioPin, level GpioLevel) error
```

PARAMETERS

NAME	TYPE	NOTES
pin	*GpioPin	
level	GpioLevel	GPIO_LEVEL_LOW GPIO_LEVEL_HIGH

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.GpioManager.SetLevel(pin, GPIO_LEVEL_HIGH)
```

PWM Manager

PWM channel configuration and control.

Permission: PwmService/*

ListChannels Returns all available PWM channels.

SIGNATURE

```
func (m *PwmManager) ListChannels() ([]*PwmChannel, error)
```

RETURNS

NAME	TYPE	NOTES
[]PwmChannel		{ Channel, Name }
error	error	

EXAMPLE

```
channels, err := client.PwmManager.ListChannels()
```

GetProperties Reads the current configuration of a PWM channel.

SIGNATURE

```
func (m *PwmManager) GetProperties(ch *PwmChannel) (*PwmProperties, error)
```

PARAMETERS

NAME	TYPE	NOTES
ch	*PwmChannel	

RETURNS

NAME	TYPE	NOTES
props	*PwmProperties	{ Channel, Enabled, DutyCycle, FrequencyHz }
error	error	

EXAMPLE

```
props, err := client.PwmManager.GetProperties(ch)
```

SetPwm Configures and starts a PWM output on the given channel.

SIGNATURE

```
func (m *PwmManager) SetPwm(ch *PwmChannel, dutyCycle, frequencyHz float64) error
```

PARAMETERS

NAME	TYPE	NOTES
ch	*PwmChannel	
dutyCycle	float64	0.0–1.0
frequencyHz	float64	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.PwmManager.SetPwm(ch, 0.5, 1000.0)
```

StopPwm Disables PWM output on the given channel.

SIGNATURE

```
func (m *PwmManager) StopPwm(ch *PwmChannel) error
```

PARAMETERS

NAME	TYPE	NOTES
ch	*PwmChannel	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.PwmManager.StopPwm(ch)
```

UART Manager

Serial port (UART) communication. Open returns a UartPort handle — all port operations are called on the handle.

Permission: UartService/*

ListPorts Returns the names of all available UART ports.

SIGNATURE

```
func (m *UartManager) ListPorts() ([]string, error)
```

RETURNS

NAME	TYPE	NOTES
[]string		
error	error	

EXAMPLE

```
ports, err := client.UartManager.ListPorts()
```

Open Opens and configures a UART port. Returns a UartPort handle for all subsequent operations.

SIGNATURE

```
func (m *UartManager) Open(cfg UartConfig) (*UartPort, error)
```

PARAMETERS

NAME	TYPE	NOTES
cfg	UartConfig	

RETURNS

NAME	TYPE	NOTES
*UartPort		
error	error	

EXAMPLE

```
port, err := client.UartManager.Open(UartConfig{
    Port:      "ttyS0",
    Baudrate:  115200,
    DataBits:  8,
    Parity:    UartParityNone,
    StopBits:  UartStopBits1,
})
```

Close Closes the port.

SIGNATURE

```
func (p *UartPort) Close() error
```

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := port.Close()
```

GetConfig Returns the current configuration of the port.

SIGNATURE

```
func (p *UartPort) GetConfig() (*UartConfig, error)
```

RETURNS

NAME	TYPE	NOTES
*UartConfig		
error	error	

EXAMPLE

```
cfg, err := port.GetConfig()
```

Write Writes raw bytes to the port. Returns the number of bytes written.

SIGNATURE

```
func (p *UartPort) Write(data []byte) (int, error)
```

PARAMETERS

NAME	TYPE	NOTES
data	[]byte	

RETURNS

NAME	TYPE	NOTES
bytesWritten	int	
error	error	

EXAMPLE

```
n, err := port.Write([]byte("hello"))
```

Listen Streams incoming bytes, calling onChunk for each chunk. Blocks until ctx is cancelled or the stream ends.

SIGNATURE

```
func (p *UartPort) Listen(ctx context.Context, maxChunkSize int, onChunk func([]byte))
error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
maxChunkSize	int	
onChunk	func([]byte)	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := port.Listen(ctx, 256, func(b []byte) {
    fmt.Printf("rx: %q", b)
})
```

ListenAsync

Like Listen but returns a channel instead of a callback. Channel is closed when the stream ends or ctx is cancelled.

SIGNATURE

```
func (p *UartPort) ListenAsync(ctx context.Context, maxChunkSize int) (<-chan []byte, error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
maxChunkSize	int	

RETURNS

NAME	TYPE	NOTES
<-chan []byte		
error	error	

EXAMPLE

```
ch, err := port.ListenAsync(ctx, 256)
for b := range ch {
    fmt.Printf("rx: %q", b)
}
```

I2C Manager

I²C bus discovery and data transfer.

Permission: I2cService/*

ListBuses Returns the indices of all available I²C buses.

SIGNATURE

```
func (m *I2CManager) ListBuses() ([]uint32, error)
```

RETURNS

NAME	TYPE	NOTES
[]uint32		
error	error	

EXAMPLE

```
buses, err := client.I2CManager.ListBuses()
```

Open Configures the bus and returns a handle for all subsequent operations.

SIGNATURE

```
func (m *I2CManager) Open(bus uint32, clockHz uint32, tenBitAddr bool, clockStretching bool) (*I2CBus, error)
```

PARAMETERS

NAME	TYPE	NOTES
bus	uint32	
clockHz	uint32	
tenBitAddr	bool	
clockStretching	bool	

RETURNS

NAME	TYPE	NOTES
*I2CBus	I2CBus	
error	error	

EXAMPLE

```
b, err := client.I2CManager.Open(1, 400000, false, false)
```

Scan Probes all 7-bit addresses (0x03–0x77) and returns those that respond.

SIGNATURE

```
func (b *I2CBus) Scan() ([]uint32, error)
```

RETURNS

NAME	TYPE	NOTES
[]uint32		responding addresses
error	error	

EXAMPLE

```
addrs, err := b.Scan()
```

GetConfig Returns the current configuration of the bus.

SIGNATURE

```
func (b *I2CBus) GetConfig() (I2CConfig, error)
```

RETURNS

NAME	TYPE	NOTES
cfg	I2CConfig	
error	error	

EXAMPLE

```
cfg, err := b.GetConfig()
```

Transfer Performs a write, read, or write-then-read transaction. Pass nil data for read-only; 0 readLen for write-only.

SIGNATURE

```
func (b *I2CBus) Transfer(addr uint32, data []byte, readLen uint32, flags uint32) ([]byte, error)
```

PARAMETERS

NAME	TYPE	NOTES
addr	uint32	7-bit device address (e.g. 0x48)
data	[]byte	bytes to write (nil for read-only)
readLen	uint32	bytes to read back (0 for write-only)
flags	uint32	i2c_msg flags (0 for most uses)

RETURNS

NAME	TYPE	NOTES
[]byte		received bytes (nil for write-only)
error	error	

EXAMPLE

```
data, err := b.Transfer(0x48, []byte{0x00}, 2, 0)
```

SPI Manager

SPI bus configuration and full-duplex transfer.

Permission: SpiService/*

ListDevices Returns available SPI device names (e.g. "spidev0.0").

SIGNATURE

```
func (m *SpiManager) ListDevices() ([]string, error)
```

RETURNS

NAME	TYPE	NOTES
[]string		
error	error	

EXAMPLE

```
devices, err := client.SpiManager.ListDevices()
```

Open Configures the SPI device at (bus, cs) and returns a handle for all subsequent operations.

SIGNATURE

```
func (m *SpiManager) Open(bus, cs uint32, maxSpeedHz, bitsPerWord uint32, mode int,
lsbFirst bool) (*SpiDevice, error)
```

PARAMETERS

NAME	TYPE	NOTES
bus	uint32	
cs	uint32	Chip select index
maxSpeedHz	uint32	
bitsPerWord	uint32	
mode	int	0–3 (CPOL/CPHA)
lsbFirst	bool	

RETURNS

NAME	TYPE	NOTES
*SpiDevice	SpiDevice	
error	error	

EXAMPLE

```
dev, err := client.SpiManager.Open(0, 0, 1000000, 8, 0, false)
```

GetConfig Returns the current configuration of the device.

SIGNATURE

```
func (d *SpiDevice) GetConfig() (*SpiConfig, error)
```

RETURNS

NAME	TYPE	NOTES
*SpiConfig	SpiConfig	
error	error	

EXAMPLE

```
cfg, err := dev.GetConfig()
```

Transfer

Performs a full-duplex SPI transfer. Pass readLength=0 for write-only; when smaller than len(dataOut), the last readLength bytes are returned.

SIGNATURE

```
func (d *SpiDevice) Transfer(dataOut []byte, readLength uint32) ([]byte, error)
```

PARAMETERS

NAME	TYPE	NOTES
dataOut	[]byte	
readLength	uint32	0 = write-only

RETURNS

NAME	TYPE	NOTES
[]byte		received MISO bytes
error	error	

EXAMPLE

```
rx, err := dev.Transfer(txData, uint32(len(txData)))
```

Camera Manager

V4L2 camera capture and streaming.

Permission: CameraService/*

ListDevices

Returns all V4L2 video devices available on the system.

SIGNATURE

```
func (m *CameraManager) ListDevices(ctx context.Context) ([]*CameraDeviceInfo, error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	

RETURNS

NAME	TYPE	NOTES
[]*CameraDeviceInfo		
error	error	

EXAMPLE

```
devices, err := client.CameraManager.ListDevices(ctx)
```

GO

GetDeviceInfo

Returns metadata (driver, card, formats, resolutions) for a specific camera device.

SIGNATURE

```
func (m *CameraManager) GetDeviceInfo(ctx context.Context, deviceID string) (*CameraDeviceInfo, error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
deviceID	string	e.g. "/dev/video0"

RETURNS

NAME	TYPE	NOTES
*CameraDeviceInfo		
error	error	

EXAMPLE

```
info, err := client.CameraManager.GetDeviceInfo(ctx, "/dev/video0")
```

LockCamera

Acquires an exclusive lock on a camera for a given client ID. Must be called before CaptureImage or StreamFrames.

SIGNATURE

```
func (m *CameraManager) LockCamera(ctx context.Context, deviceID, clientID string) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
deviceID	string	
clientID	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.CameraManager.LockCamera(ctx, "/dev/video0", "myapp")
```

UnlockCamera

 Releases the lock obtained with LockCamera.

SIGNATURE

```
func (m *CameraManager) UnlockCamera(ctx context.Context, deviceID, clientID string) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
deviceID	string	
clientID	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.CameraManager.UnlockCamera(ctx, "/dev/video0", "myapp")
```

CaptureImage Captures a single frame from a camera device. Requires LockCamera first.

SIGNATURE

```
func (m *CameraManager) CaptureImage(ctx context.Context, deviceID string, width, height
int32, format string) (*CaptureImageResult, error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
deviceID	string	
width	int32	
height	int32	
format	string	"mjpeg" or "yuyv"

RETURNS

NAME	TYPE	NOTES
*CaptureImageResult		
error	error	

EXAMPLE

```
img, err := client.CameraManager.CaptureImage(ctx, "/dev/video0", 1280, 720, "mjpeg")
```

StreamFrames

Opens a server-streaming frame stream at the requested FPS and resolution. Read from the stream until io.EOF or error.

SIGNATURE

```
func (m *CameraManager) StreamFrames(ctx context.Context, req *StreamFramesRequest)
(ServerStream[Frame], error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
req	*StreamFramesRequest	

RETURNS

NAME	TYPE	NOTES
ServerStream[Frame]		
error	error	

EXAMPLE

```
stream, err := client.CameraManager.StreamFrames(ctx, &StreamFramesRequest{
    DeviceId: "/dev/video0", Fps: 30, Width: 1280, Height: 720,
})
for {
    frame, err := stream.Recv()
    if err == io.EOF { break }
}
```

AI Manager

On-device AI model management and inference. LoadModel / UploadAndLoadModel return an AIModel handle — inference and lifecycle operations are called on the handle.

Permission: AiService/*

ListModels Returns metadata for all currently loaded models.

SIGNATURE

```
func (m *AIManager) ListModels() ([]*ModelInfo, error)
```

RETURNS

NAME	TYPE	NOTES
[]*ModelInfo		
error	error	

EXAMPLE

```
models, err := client.AIManager.ListModels()
```

LoadModel Loads a model already present on the device filesystem. Returns an AIModel handle.

SIGNATURE

```
func (m *AIManager) LoadModel(modelID, modelPath string, backend ModelBackend, execution
ExecutionMode) (*AIModel, error)
```

PARAMETERS

NAME	TYPE	NOTES
modelID	string	
modelPath	string	Absolute path on the device
backend	ModelBackend	
execution	ExecutionMode	

RETURNS

NAME	TYPE	NOTES
*AIModel		
error	error	

EXAMPLE

```
model, err := client.AIManager.LoadModel(
    "detector", "/models/yolo.onnx",
    ModelBackend_ONNX, ExecutionMode_EXEC_CPU,
)
```

UploadAndLoadModel Streams a local model file to the device in chunks then loads it. Returns an AIModel handle.

SIGNATURE

```
func (m *AIManager) UploadAndLoadModel(modelID, localPath string, backend ModelBackend,
execution ExecutionMode) (*AIModel, error)
```

PARAMETERS

NAME	TYPE	NOTES
modelID	string	
localPath	string	Path to the model file on the host
backend	ModelBackend	
execution	ExecutionMode	

RETURNS

NAME	TYPE	NOTES
*AIModel		
error	error	

EXAMPLE

```
model, err := client.AIManager.UploadAndLoadModel(
    "detector", "./yolo.onnx",
    ModelBackend_ONNX, ExecutionMode_EXEC_CPU,
)
```

Unload Frees the model from the inference backend.

SIGNATURE

```
func (m *AIModel) Unload() error
```

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := model.Unload()
```

IsLoaded Returns whether the model is loaded and its tensor schema.

SIGNATURE

```
func (m *AIModel) IsLoaded() (*IsModelLoadedResponse, error)
```

RETURNS

NAME	TYPE	NOTES
*IsModelLoadedResponse		
error	error	

EXAMPLE

```
r, err := model.IsLoaded()
```

Infer Runs a single synchronous forward pass. inputShape nil uses the schema from load.

SIGNATURE

```
func (m *AIModel) Infer(ctx context.Context, inputData []byte, inputShape []int32, dtype TensorDataType) (*InferResponse, error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
inputData	[]byte	Raw little-endian bytes
inputShape	[]int32	nil = use schema from LoadModel
dtype	TensorDataType	

RETURNS

NAME	TYPE	NOTES
*InferResponse		
error	error	

EXAMPLE

```
resp, err := model.Infer(ctx, inputData, []int32{1, 3, 640, 640},  
TensorDataType_FLOAT32)
```

StreamInfer Opens a bidirectional inference stream. Returns an AIInferStream handle.

SIGNATURE

```
func (m *AIModel) StreamInfer(ctx context.Context) (*AIInferStream, error)
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	

RETURNS

NAME	TYPE	NOTES
*AIInferStream		
error	error	

EXAMPLE

```
s, err := model.StreamInfer(ctx)
```

Send Submits an inference request on the stream.

SIGNATURE

```
func (s *AIInferStream) Send(inputData []byte, inputShape []int32, dtype TensorDataType) error
```

PARAMETERS

NAME	TYPE	NOTES
inputData	[]byte	
inputShape	[]int32	
dtype	TensorDataType	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := s.Send(inputData, nil, TensorDataType_FLOAT32)
```

Recv Reads the next inference result. Returns io.EOF when the server closes the stream.

SIGNATURE

```
func (s *AIInferStream) Recv() (*InferResponse, error)
```

RETURNS

NAME	TYPE	NOTES
*InferResponse		
error	error	

EXAMPLE

```
resp, err := s.Recv()
```

Close Signals the end of the send side of the stream.

SIGNATURE

```
func (s *AIInferStream) Close() error
```

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := s.Close()
```

Firewall Manager

Network firewall zone and rule management.

Permission: FirewallService/*

ListZones Returns all configured firewall zones.

SIGNATURE

```
func (f *FirewallManager) ListZones() ([]*ZoneRequest, error)
```

RETURNS

NAME	TYPE	NOTES
[]*ZoneRequest		
error	error	

EXAMPLE

```
zones, err := client.FirewallManager.ListZones()
```

AddZone Creates a new firewall zone.

SIGNATURE

```
func (f *FirewallManager) AddZone(name string, interfaces []string, inputPolicy,
outputPolicy ZonePolicy, masquerade bool) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
name	string	
interfaces	[]string	
inputPolicy	ZonePolicy	
outputPolicy	ZonePolicy	
masquerade	bool	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.FirewallManager.AddZone("lan", []string{"eth0"}, ZonePolicy_ACCEPT,
ZonePolicy_ACCEPT, false)
```

RemoveZone Deletes a firewall zone by name.

SIGNATURE

```
func (f *FirewallManager) RemoveZone(name string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
name	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.FirewallManager.RemoveZone("lan")
```

ListRules Returns all configured firewall rules.

SIGNATURE

```
func (f *FirewallManager) ListRules() ([]*FirewallRule, error)
```

RETURNS

NAME	TYPE	NOTES
[]*FirewallRule		
error	error	

EXAMPLE

```
rules, err := client.FirewallManager.ListRules()
```

AddRule Adds a traffic rule between two zones.

SIGNATURE

```
func (f *FirewallManager) AddRule(srcZone, dstZone string, protocol FirewallProtocol,
srcIP string, destPort int32, action ZonePolicy, comment string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
srcZone	string	
dstZone	string	
protocol	FirewallProtocol	PROTO_ANY PROTO_TCP PROTO_UDP PROTO_ICMP
srcIP	string	CIDR or empty for any
destPort	int32	0 for any
action	ZonePolicy	
comment	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.FirewallManager.AddRule("lan", "wan", FirewallProtocol_TCP, "", 443,
ZonePolicy_ACCEPT, "allow HTTPS")
```

RemoveRule Removes a traffic rule by its ID.

SIGNATURE

```
func (f *FirewallManager) RemoveRule(id string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
id	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.FirewallManager.RemoveRule(ruleID)
```

FlushRules Removes all traffic rules (zones are preserved).

SIGNATURE

```
func (f *FirewallManager) FlushRules() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.FirewallManager.FlushRules()
```

ApplyFirewall Commits all pending zone and rule changes to the active firewall.

SIGNATURE

```
func (f *FirewallManager) ApplyFirewall() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.FirewallManager.ApplyFirewall()
```

VPN Manager

VPN profile management and connection control.

Permission: VpnService/*

GetCapabilities Reports which VPN providers are available on the device.

SIGNATURE

```
func (v *VPNManager) GetCapabilities() (*VpnCapabilities, error)
```

RETURNS

NAME	TYPE	NOTES
*VpnCapabilities		
error	error	

EXAMPLE

```
caps, err := client.VPNManager.GetCapabilities()
```

GO

ListProfiles Returns all saved VPN profiles (config data not included).

SIGNATURE

```
func (v *VPNManager) ListProfiles() ([]*VpnProfile, error)
```

RETURNS

NAME	TYPE	NOTES
[]*VpnProfile		
error	error	

EXAMPLE

```
profiles, err := client.VPNManager.ListProfiles()
```

ApplyProfile Creates or updates a VPN profile. If ProfileId is empty the server assigns one.

SIGNATURE

```
func (v *VPNManager) ApplyProfile(profile *VpnProfile, connectAfterApply bool) (string, error)
```

PARAMETERS

NAME	TYPE	NOTES
profile	*VpnProfile	
connectAfterApply	bool	

RETURNS

NAME	TYPE	NOTES
profileID	string	
error	error	

EXAMPLE

```
id, err := client.VPNManager.ApplyProfile(profile, false)
```

ApplyWireGuard Convenience wrapper — creates or updates a WireGuard profile from a raw .conf file.

SIGNATURE

```
func (v *VPNManager) ApplyWireGuard(displayName string, configData []byte, autoConnect bool) (string, error)
```

PARAMETERS

NAME	TYPE	NOTES
displayName	string	
configData	[]byte	Raw WireGuard .conf content
autoConnect	bool	

RETURNS

NAME	TYPE	NOTES
profileID	string	
error	error	

EXAMPLE

```
id, err := client.VPNManager.ApplyWireGuard("office", confBytes, true)
```

ApplyOpenVPN Convenience wrapper — creates or updates an OpenVPN profile from a raw .ovpn file.

SIGNATURE

```
func (v *VPNManager) ApplyOpenVPN(displayName string, configData []byte, autoConnect bool) (string, error)
```

PARAMETERS

NAME	TYPE	NOTES
displayName	string	
configData	[]byte	Raw .ovpn file content
autoConnect	bool	

RETURNS

NAME	TYPE	NOTES
profileID	string	
error	error	

EXAMPLE

```
id, err := client.VPNManager.ApplyOpenVPN("corp", ovpnBytes, false)
```

RemoveProfile Deletes a VPN profile by ID (disconnects first if active).

SIGNATURE

```
func (v *VPNManager) RemoveProfile(profileID string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
profileID	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.VPNManager.RemoveProfile(profileID)
```

Connect

Activates a VPN profile and returns the session ID. The tunnel comes up asynchronously — use WatchEvents or GetStatus to track.

SIGNATURE

```
func (v *VPNManager) Connect(profileID string) (string, error)
```

PARAMETERS

NAME	TYPE	NOTES
profileID	string	

RETURNS

NAME	TYPE	NOTES
sessionID	string	
error	error	

EXAMPLE

```
sessionID, err := client.VPNManager.Connect(profileID)
```

Disconnect

Tears down the active tunnel for the given profile. Pass empty string to disconnect the currently active profile.

SIGNATURE

```
func (v *VPNManager) Disconnect(profileID string) (bool, error)
```

PARAMETERS

NAME	TYPE	NOTES
profileID	string	

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.VPNManager.Disconnect(profileID)
```

GetStatus Returns the current active session and provider details. Session is nil when no tunnel is active.

SIGNATURE

```
func (v *VPNManager) GetStatus() (*Session, string, error)
```

RETURNS

NAME	TYPE	NOTES
*Session		
providerDetails	string	
error	error	

EXAMPLE

```
sess, details, err := client.VPNManager.GetStatus()
```

ListSessions Returns all active sessions (at most one in the current implementation).

SIGNATURE

```
func (v *VPNManager) ListSessions() ([]*Session, error)
```

RETURNS

NAME	TYPE	NOTES
[]*Session		
error	error	

EXAMPLE

```
sessions, err := client.VPNManager.ListSessions()
```

IsConnected Reports whether there is an active tunnel in state UP.

SIGNATURE

```
func (v *VPNManager) IsConnected() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
bool		
error	error	

EXAMPLE

```
connected, err := client.VPNManager.IsConnected()
```

WatchEvents

Streams VPN tunnel events. Blocks until ctx is cancelled or the server closes the connection. Pass empty profileID to receive events from all profiles.

SIGNATURE

```
func (v *VPNManager) WatchEvents(ctx context.Context, profileID string, handler func(*VPNEvent)) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
profileID	string	empty = all profiles
handler	func(*VPNEvent)	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.VPNManager.WatchEvents(ctx, "", func(e *VPNEvent) {
    fmt.Println(e.GetState())
})
```

App Hub Manager

Service registration and HTTP routing via the Orbit app hub.

Permission: AppHubService/*

RegisterWebUI

Convenience wrapper — registers a local HTTP server with a TCP health-check. Prefer this for the common case.

SIGNATURE

```
func (m *AppHubManager) RegisterWebUI(addr, route string) error
```

PARAMETERS

NAME	TYPE	NOTES
addr	string	"host:port" or ":port"
route	string	Path prefix to proxy, e.g. "/myapp"

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AppHubManager.RegisterWebUI("127.0.0.1:9033", "/myapp")
```

RegisterService Announces the app's HTTP server to the portal with full control over routes and health check.

SIGNATURE

```
func (m *AppHubManager) RegisterService(req *RegisterServiceRequest) error
```

PARAMETERS

NAME	TYPE	NOTES
req	*RegisterServiceRequest	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AppHubManager.RegisterService(req)
```

UnregisterService Removes the app's registration from the portal. Safe to call on shutdown.

SIGNATURE

```
func (m *AppHubManager) UnregisterService() error
```

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AppHubManager.UnregisterService()
```

GetService Returns registration details for a given service ID.

SIGNATURE

```
func (m *AppHubManager) GetService(serviceID string) (*Service, error)
```

PARAMETERS

NAME	TYPE	NOTES
serviceID	string	

RETURNS

NAME	TYPE	NOTES
*Service		
error	error	

EXAMPLE

```
svc, err := client.AppHubManager.GetService("my-service-id")
```

ListServices Returns all services currently registered in the portal.

SIGNATURE

```
func (m *AppHubManager) ListServices() ([]*Service, error)
```

RETURNS

NAME	TYPE	NOTES
[]*Service		
error	error	

EXAMPLE

```
services, err := client.AppHubManager.ListServices()
```

AddRoute Adds an extra path prefix to the already-registered service.

SIGNATURE

```
func (m *AppHubManager) AddRoute(path string) error
```

PARAMETERS

NAME	TYPE	NOTES
path	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AppHubManager.AddRoute("/api/v2")
```

RemoveRoute Removes a path prefix from the registered service.

SIGNATURE

```
func (m *AppHubManager) RemoveRoute(path string) error
```

PARAMETERS

NAME	TYPE	NOTES
path	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AppHubManager.RemoveRoute("/api/v1")
```

GetRoutingTable Returns all current proxy routing entries.

SIGNATURE

```
func (m *AppHubManager) GetRoutingTable() ([]*RoutingEntry, error)
```

RETURNS

NAME	TYPE	NOTES
[]*RoutingEntry		
error	error	

EXAMPLE

```
entries, err := client.AppHubManager.GetRoutingTable()
```

WatchServices Streams service lifecycle events. Blocks until ctx is cancelled or the server closes.

SIGNATURE

```
func (m *AppHubManager) WatchServices(ctx context.Context, handler func(*ServiceEvent)) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
handler	func(*ServiceEvent)	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.AppHubManager.WatchServices(ctx, func(e *ServiceEvent) {  
    fmt.Println(e.GetType(), e.GetService().GetServiceId())  
})
```

Event Manager

System-wide event bus.

Permission: EventService/*

Subscribe Subscribes to system events. Calls handler for each event. Blocks until context is cancelled.

SIGNATURE

```
func (e *EventManager) Subscribe(ctx context.Context, handler func(*Event), types
...EventType) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
handler	func(*Event)	
types	...EventType	Omit to receive all events (EVENT_APP_INSTALLED, EVENT_APP_REMOVED, EVENT_APP_STARTED, etc.)

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.EventManager.Subscribe(ctx, func(e *Event) {
    fmt.Println(e.Type, e.Payload)
}, client.EVENT_APP_STARTED)
```

Development Manager

Live log streaming from running applications.

Permission: DevelopmentService/*

SubscribeLogs Streams log entries matching the given filter. Blocks until context is cancelled.

SIGNATURE

```
func (d *DevelopmentManager) SubscribeLogs(ctx context.Context, app string, tag string,
level LogLevel, onEntry func(LogEntry)) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
app	string	app name filter (empty = all)
tag	string	tag filter (empty = all)
level	LogLevel	minimum log level (LOG_LEVEL_DEBUG, LOG_LEVEL_INFO, LOG_LEVEL_WARNING, LOG_LEVEL_ERROR, LOG_LEVEL_FATAL)
onEntry	func(LogEntry)	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.DevelopmentManager.SubscribeLogs(ctx, "myapp", "", client.LOG_LEVEL_INFO,
func(e LogEntry) {
    fmt.Println(e.Timestamp, e.Message)
})
```

SubscribeLogsAsync Starts SubscribeLogs in a goroutine with automatic reconnection. Returns a channel.

SIGNATURE

```
func (d *DevelopmentManager) SubscribeLogsAsync(ctx context.Context, app string, tag string, level LogLevel) <-chan LogEntry
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
app	string	app name filter (empty = all)
tag	string	tag filter (empty = all)
level	LogLevel	minimum log level (LOG_LEVEL_DEBUG, LOG_LEVEL_INFO, LOG_LEVEL_WARNING, LOG_LEVEL_ERROR, LOG_LEVEL_FATAL)

RETURNS

NAME	TYPE	NOTES
<-chan LogEntry	chan	
error	error	

EXAMPLE

```
ch := client.DevelopmentManager.SubscribeLogsAsync(ctx, "myapp", "", client.LOG_LEVEL_INFO)
```

Update Manager

OTA firmware update and factory reset.

Permission: UpdateService/*

Update Sends a .orbit system image to the device to perform an OS update.

SIGNATURE

```
func (u *UpdateManager) Update(ctx context.Context, orbitPath string) error
```

PARAMETERS

NAME	TYPE	NOTES
ctx	Context	
orbitPath	string	

RETURNS

NAME	TYPE	NOTES
error	error	

EXAMPLE

```
err := client.UpdateManager.Update(ctx, "/path/to/update.orbit")
```

FactoryReset Performs a factory reset, erasing all user data.

SIGNATURE

```
func (u *UpdateManager) FactoryReset() (bool, error)
```

RETURNS

NAME	TYPE	NOTES
ok	bool	
error	error	

EXAMPLE

```
ok, err := client.UpdateManager.FactoryReset()
```

Power Manager

Device power control.

Permission: PowerService/*

Reboot

Reboots the device.

SIGNATURE

```
func (p *PowerManager) Reboot(force bool, reason string) (*PowerResult, error)
```

PARAMETERS

NAME	TYPE	NOTES
force	bool	
reason	string	

RETURNS

NAME	TYPE	NOTES
*PowerResult		{ Success, Message }
error	error	

EXAMPLE

```
res, err := client.PowerManager.Reboot(false, "update")
```

GO

Shutdown

Shuts down the device.

SIGNATURE

```
func (p *PowerManager) Shutdown(force bool, reason string) (*PowerResult, error)
```

PARAMETERS

NAME	TYPE	NOTES
force	bool	
reason	string	

RETURNS

NAME	TYPE	NOTES
*PowerResult		{ Success, Message }
error	error	

EXAMPLE

```
res, err := client.PowerManager.Shutdown(false, "maintenance")
```

Types

Complex objects returned or accepted by the SDK methods.

GpioPin

Identifies a GPIO line by chip index and line offset.

FIELDS		
NAME	TYPE	NOTES
Name	string	Human-readable pin name (e.g. "GPIO17", "PWM0_OUT")
Number	int32	Line offset within the chip
ChipNumber	int32	Chip index (0 = /dev/gpiochip0)

MetricsInfoResponse

Response returned by SystemManager.GetMetrics.

FIELDS		
NAME	TYPE	NOTES
Metrics	*MetricsInfo	System metrics snapshot
Error	*ErrorInfo	Non-nil when the request failed

MetricsInfo

Full snapshot of system metrics at the time of the call.

FIELDS		
NAME	TYPE	NOTES
SysUptime	*MetricsInfo_SysUptime	System uptime and load
CpuUsage	float64	Overall CPU usage (0–100 %)
CpuCorePercent	[]float64	Per-core CPU usage (0–100 %)
MemoryUsage	float64	Memory usage percentage
SocThermal	float64	SoC temperature (°C)
CpuStats	*MetricsInfo_CpuStats	CPU context-switch and interrupt counters
CpuFreq	*MetricsInfo_CpuFreq	CPU frequency info (MHz)
VirtualMemory	*MetricsInfo_VirtualMemory	Detailed memory breakdown
DiskUsage	map[string]*MetricsInfo_DiskUsage	Disk usage per mount point
DiskIoCounters	*MetricsInfo_DiskIoCounters	Disk I/O counters
Network	*MetricsInfo_Network	Network I/O counters
SensorsTemperatures	int32	Sensor temperature reading

MetricsInfo_SysUptime

System uptime and load average.

FIELDS		
NAME	TYPE	NOTES
Time	string	Current system time string
Uptime	int64	Uptime in seconds
Users	int32	Number of logged-in users
LoadAverage	[]float64	1-min, 5-min, 15-min load averages

MetricsInfo_CpuStats

CPU context-switch and interrupt counters since boot.

FIELDS			
NAME		TYPE	NOTES
CtxSwitches		int64	Total context switches
Interrupts		int64	Total hardware interrupts
SoftInterrupts		int64	Total software interrupts
Syscalls		int64	Total system calls

MetricsInfo_CpuFreq

CPU frequency information in MHz.

FIELDS		
NAME	TYPE	NOTES
Current	float64	Current CPU frequency (MHz)
Min	float64	Minimum CPU frequency (MHz)
Max	float64	Maximum CPU frequency (MHz)

MetricsInfo_VirtualMemory

Detailed virtual memory breakdown (all sizes in bytes).

FIELDS		
NAME	TYPE	NOTES
Total	int64	Total installed RAM (bytes)
Available	int64	Available memory (bytes)
Percent	float64	Usage percentage
Used	int64	Used memory (bytes)
Free	int64	Free memory (bytes)
Active	int64	Active memory (bytes)
Inactive	int64	Inactive memory (bytes)
Buffers	int64	Buffer cache (bytes)
Cached	int64	Page cache (bytes)
Shared	int64	Shared memory (bytes)
Slab	int64	Kernel slab cache (bytes)

MetricsInfo_DiskUsage

Disk usage for a single mount point (all sizes in bytes).

FIELDS		
NAME	TYPE	NOTES
Total	int64	Total disk size (bytes)
Used	int64	Used space (bytes)
Free	int64	Free space (bytes)
Percent	float64	Usage percentage

MetricsInfo_DiskIoCounters

Disk I/O counters since boot.

FIELDS		
NAME	TYPE	NOTES
ReadCount	int64	Number of reads
WriteCount	int64	Number of writes
ReadBytes	int64	Bytes read
WriteBytes	int64	Bytes written
ReadTime	int64	Time spent reading (ms)
WriteTime	int64	Time spent writing (ms)
ReadMergedCount	int64	Merged read operations
WriteMergedCount	int64	Merged write operations
BusyTime	int64	Time disk was busy (ms)

MetricsInfo_Network

Aggregated network I/O counters across all interfaces.

FIELDS		
NAME	TYPE	NOTES
BytesSent	int64	Total bytes sent
BytesRecv	int64	Total bytes received
PacketsSent	int64	Total packets sent
PacketsRecv	int64	Total packets received
Errin	int64	Total input errors
Errout	int64	Total output errors
Dropin	int64	Total inbound packets dropped
Dropout	int64	Total outbound packets dropped

EthernetLinkProperties

Link-layer and IP properties of an Ethernet interface.

FIELDS		
NAME	TYPE	NOTES
InterfaceName	string	Interface name (e.g. "eth0")
MacAddress	string	MAC address
State	EthernetState	Current link state
Mtu	int32	Maximum transmission unit
DhcpEnable	bool	True when DHCP is active

ZoneRequest

Firewall zone definition (as returned by ListZones).

FIELDS		
NAME	TYPE	NOTES
Name	string	Zone name (e.g. "lan", "wan")
Interfaces	[]string	Network interfaces assigned to this zone
InputPolicy	ZonePolicy	POLICY_ACCEPT POLICY_DROP POLICY_REJECT
OutputPolicy	ZonePolicy	POLICY_ACCEPT POLICY_DROP POLICY_REJECT
Masquerade	bool	Enable NAT masquerade (typically wan only)

FirewallRule

A firewall traffic rule as returned by ListRules.

FIELDS		
NAME	TYPE	NOTES
Id	string	Server-generated rule ID
SrcZone	string	Source zone (empty = any)
DstZone	string	Destination zone (empty = any)
Protocol	FirewallProtocol	PROTO_ANY PROTO_TCP PROTO_UDP PROTO_ICMP
SrcIp	string	Source IP/CIDR (empty = any)
DestPort	int32	Destination port (0 = any)
Action	ZonePolicy	POLICY_ACCEPT POLICY_DROP POLICY_REJECT
Comment	string	Optional description

SendOptions

Optional fields for `MobileNotificationManager.Send`. A nil `*SendOptions` is equivalent to an empty one.

FIELDS

NAME	TYPE	NOTES
<code>ImageUrl</code>	<code>string</code>	Image shown in the notification
<code>Data</code>	<code>map[string]string</code>	Delivered to the Android app for deep-linking, etc.
<code>CollapseKey</code>	<code>string</code>	Replaces any pending notification with the same key
<code>TtlSeconds</code>	<code>int32</code>	How long FCM holds the notification if the device is offline

Java API Reference

Orbit OS SDK · Version 26.0.1 · Generated 2026-09-18

CONTENTS

Auth Manager	2	Camera Manager	40
System Manager	3	Ai Manager	43
Package Manager	18	Firewall Manager	46
Ethernet Manager	20	Vpn Manager	49
WiFi Manager	23	App Hub Manager	54
GPIO Manager	27	Event Manager	58
PWM Manager	30	Development Manager	59
UART Manager	32	Update Manager	60
I2C Manager	35	Power Manager	61
SPI Manager	38	Types	62

Auth Manager

Local device authentication via Gravity.

login

Authenticates a user and returns a session token and its expiry.

SIGNATURE

`LoginResult login(String username, String password) throws AuthException`

PARAMETERS

NAME	TYPE	NOTES
username	String	
password	String	

RETURNS

NAME	TYPE	NOTES
token	String	
expiresAt	long	Unix timestamp

EXAMPLE

`var r = client.authManager().login("admin", "secret");
String token = r.token();`

JAVA

logout

Invalidates a session token.

SIGNATURE

`void logout(String token) throws AuthException`

PARAMETERS

NAME	TYPE	NOTES
token	String	

EXAMPLE

`client.authManager().logout(token);`

System Manager

Device identity, hardware info, OS metadata and live metrics.

Permission: `SystemService/*`

getApiVersion Returns the API version and revision running on the device.

SIGNATURE

`ApiVersion getApiVersion() throws SystemServiceException`

RETURNS

NAME	TYPE	NOTES
value	<code>ApiVersion</code>	

EXAMPLE

`var v = client.systemManager().getApiVersion();`

getApiVersionInfo Returns the API version as a formatted string (e.g. "26.0").

SIGNATURE

`String getApiVersionInfo() throws SystemServiceException`

RETURNS

NAME	TYPE	NOTES
value	<code>String</code>	

EXAMPLE

`String ver = client.systemManager().getApiVersionInfo();`

getDeviceName Returns the configured device name.

SIGNATURE

```
String getDeviceName() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getDeviceName();
```

getSocModel Returns the SoC model identifier.

SIGNATURE

```
String getSocModel() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getSocModel();
```

getSocVendor Returns the SoC vendor name.

SIGNATURE

```
String getSocVendor() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getSocVendor();
```

getBoardModel Returns the board model identifier.

SIGNATURE

```
String getBoardModel() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getBoardModel();
```

getBoardVendor Returns the board vendor name.

SIGNATURE

```
String getBoardVendor() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getBoardVendor();
```

getHardwareVersion Returns the hardware revision string.

SIGNATURE

```
String getHardwareVersion() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getHardwareVersion();
```

getHardwareModel Returns the hardware model identifier.

SIGNATURE

```
String getHardwareModel() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getHardwareModel();
```

getSystemUuid Returns the system UUID.

SIGNATURE

```
String getSystemUuid() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getSystemUuid();
```

getBoardSerial Returns the board serial number.

SIGNATURE

```
String getBoardSerial() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getBoardSerial();
```

getCpuSerial Returns the CPU serial number.

SIGNATURE

```
String getCpuSerial() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getCpuSerial();
```

getMachineId Returns the machine ID.

SIGNATURE

```
String getMachineId() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getMachineId();
```

getArchitecture Returns the CPU architecture (e.g. "arm64", "amd64").

SIGNATURE

```
String getArchitecture() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getArchitecture();
```

getTotalRAM Returns total RAM in bytes.

SIGNATURE

```
long getTotalRAM() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	long	

EXAMPLE

```
var value = client.systemManager().getTotalRAM();
```

getCpuModel Returns the CPU model string.

SIGNATURE

```
String getCpuModel() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getCpuModel();
```

getCpuCores Returns the number of physical CPU cores.

SIGNATURE

```
long getCpuCores() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	long	

EXAMPLE

```
var value = client.systemManager().getCpuCores();
```

getCpuThreads Returns the number of logical CPU threads.

SIGNATURE

```
long getCpuThreads() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	long	

EXAMPLE

```
var value = client.systemManager().getCpuThreads();
```

getCpuMinMhz Returns the minimum CPU frequency in MHz.

SIGNATURE

```
double getCpuMinMhz() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	double	

EXAMPLE

```
var value = client.systemManager().getCpuMinMhz();
```

getCpuMaxMhz Returns the maximum CPU frequency in MHz.

SIGNATURE

```
double getCpuMaxMhz() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	double	

EXAMPLE

```
var value = client.systemManager().getCpuMaxMhz();
```

getOsName Returns the OS name.

SIGNATURE

```
String getOsName() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getOsName();
```

getOsVersion Returns the OS version string.

SIGNATURE

```
String getOsVersion() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getOsVersion();
```

getKernelVersion Returns the kernel version string.

SIGNATURE

```
String getKernelVersion() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getKernelVersion();
```

getDistro Returns the Linux distribution name.

SIGNATURE

```
String getDistro() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getDistro();
```

getDistroVersion Returns the distribution version string.

SIGNATURE

```
String getDistroVersion() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getDistroVersion();
```

getRuntimeVersion Returns the Orbit runtime version.

SIGNATURE

```
String getRuntimeVersion() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getRuntimeVersion();
```

getOsRevision Returns the logical OS revision string.

SIGNATURE

```
String getOsRevision() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getOsRevision();
```

getBuildVersion Alias for getOsRevision.

SIGNATURE

```
String getBuildVersion() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getBuildVersion();
```

getRuntimeBuildDate Returns the runtime image build date.

SIGNATURE

```
String getRuntimeBuildDate() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getRuntimeBuildDate();
```

getBuildDate Alias for getRuntimeBuildDate.

SIGNATURE

```
String getBuildDate() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	String	

EXAMPLE

```
var value = client.systemManager().getBuildDate();
```

getMetrics Returns live CPU usage, RAM, storage and network stats.

SIGNATURE

```
MetricsInfoResponse getMetrics() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
value	MetricsInfoResponse	

EXAMPLE

```
var m = client.systemManager().getMetrics();
System.out.printf("CPU=%.1f%% MEM=%.1f%%\n", m.getCpuUsage(), m.getMemoryUsage());
```

attach Attaches the SDK client to the running Gravity system service.

SIGNATURE

```
boolean attach() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().attach();
```

enableDevMode Enables developer mode on the device.

SIGNATURE

```
boolean enableDevMode() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().enableDevMode();
```

disableDevMode Disables developer mode on the device.

SIGNATURE

```
boolean disableDevMode() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().disableDevMode();
```

isDevModeEnabled Reports whether developer mode is currently enabled.

SIGNATURE

```
boolean isDevModeEnabled() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().isDevModeEnabled();
```

enableSSHServer Enables the SSH server on the device.

SIGNATURE

```
boolean enableSSHServer() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().enableSSHServer();
```

disableSSHServer Disables the SSH server on the device.

SIGNATURE

```
boolean disableSSHServer() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().disableSSHServer();
```

isSSHServerEnabled Reports whether the SSH server is currently enabled.

SIGNATURE

```
boolean isSSHServerEnabled() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().isSSHServerEnabled();
```

enableRebootOnFailure Enables automatic reboot when a critical failure is detected.

SIGNATURE

```
boolean enableRebootOnFailure() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().enableRebootOnFailure();
```

disableRebootOnFailure Disables automatic reboot on failure.

SIGNATURE

```
boolean disableRebootOnFailure() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().disableRebootOnFailure();
```

isRebootOnFailureEnabled Reports whether automatic reboot on failure is enabled.

SIGNATURE

```
boolean isRebootOnFailureEnabled() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().isRebootOnFailureEnabled();
```

allowUntrustedApps Allows installation of apps from untrusted sources.

SIGNATURE

```
boolean allowUntrustedApps() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().allowUntrustedApps();
```

disallowUntrustedApps Blocks installation of apps from untrusted sources.

SIGNATURE

```
boolean disallowUntrustedApps() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().disallowUntrustedApps();
```

isUntrustedAppsAllowed Reports whether untrusted app installation is currently allowed.

SIGNATURE

```
boolean isUntrustedAppsAllowed() throws SystemServiceException
```

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.systemManager().isUntrustedAppsAllowed();
```

Package Manager

Application package (.orb) installation and management.

Permission: PackageService/*

getInstalledPackages Returns all installed packages.

SIGNATURE

`List<InstalledPackage> getInstalledPackages() throws PackageManagerException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<InstalledPackage></code>	

EXAMPLE

`var pkgs = client.packageManager().getInstalledPackages();`

installUpdatePackage Uploads and installs (or updates) an .orb package, streamed in chunks.

SIGNATURE

`boolean installUpdatePackage(String filePath, int chunkSize) throws PackageManagerException, IOException`

PARAMETERS

NAME	TYPE	NOTES
filePath	<code>String</code>	
chunkSize	<code>int</code>	Upload chunk size in bytes (e.g. 262144)

RETURNS

NAME	TYPE	NOTES
installed	<code>boolean</code>	

EXAMPLE

`boolean ok = client.packageManager().installUpdatePackage("/path/to/app.orb", 262144);`

removePackage Uninstalls a package by ID.

SIGNATURE

```
boolean removePackage(String packageId) throws PackageManagerException
```

PARAMETERS

NAME	TYPE	NOTES
packageId	String	

RETURNS

NAME	TYPE	NOTES
removed	boolean	

EXAMPLE

```
boolean ok = client.packageManager().removePackage("com.example.app");
```

Ethernet Manager

Ethernet interface configuration and status.

Permission: EthernetService/*

listInterfaces Returns all Ethernet interfaces on the device.

SIGNATURE

`List<EthernetInterfaceInfo> listInterfaces() throws EthernetException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<EthernetInterfaceInfo></code>	

EXAMPLE

`var ifaces = client.ethernetManager().listInterfaces();`

isConnected Reports whether the given interface has link.

SIGNATURE

`boolean isConnected(String ifname) throws EthernetException`

PARAMETERS

NAME	TYPE	NOTES
ifname	<code>String</code>	

RETURNS

NAME	TYPE	NOTES
connected	<code>boolean</code>	

EXAMPLE

`boolean up = client.ethernetManager().isConnected("eth0");`

getLinkProperties Returns link properties (MAC, state, MTU, IP config) for an interface.

SIGNATURE

```
EthernetInterfaceInfo getLinkProperties(String ifname) throws EthernetException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

RETURNS

NAME	TYPE	NOTES
value	EthernetInterfaceInfo	

EXAMPLE

```
var props = client.ethernetManager().getLinkProperties("eth0");
```

setStaticConfig Applies a static IPv4 configuration to an interface.

SIGNATURE

```
void setStaticConfig(String ifname, String ipv4Address, String ipv4Gateway, List<String> ipv4Dns) throws EthernetException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	
ipv4Address	String	CIDR notation, e.g. "192.168.1.10/24"
ipv4Gateway	String	
ipv4Dns	List<String>	

EXAMPLE

```
client.ethernetManager().setStaticConfig(
    "eth0", "192.168.1.10/24", "192.168.1.1", List.of("8.8.8.8"));
```

setDhcp Switches an interface to DHCP addressing.

SIGNATURE

```
void setDhcp(String ifname) throws EthernetException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

EXAMPLE

```
client.ethernetManager().setDhcp("eth0");
```

enable Brings an Ethernet interface up.

SIGNATURE

```
void enable(String ifname) throws EthernetException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

EXAMPLE

```
client.ethernetManager().enable("eth0");
```

disable Brings an Ethernet interface down.

SIGNATURE

```
void disable(String ifname) throws EthernetException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

EXAMPLE

```
client.ethernetManager().disable("eth0");
```

WiFi Manager

Wi-Fi interface management — client mode and access point.

Permission: `WifiService/*`

listInterfaces Returns all Wi-Fi interfaces on the device.

SIGNATURE

`List<WifiInterfaceInfo> listInterfaces() throws WifiException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<WifiInterfaceInfo></code>	

EXAMPLE

`var ifaces = client.wifiManager().listInterfaces();`

getLinkProperties Returns link-layer and mode properties for a Wi-Fi interface.

SIGNATURE

`WifiInterfaceInfo getLinkProperties(String ifname) throws WifiException`

PARAMETERS

NAME	TYPE	NOTES
ifname	<code>String</code>	

RETURNS

NAME	TYPE	NOTES
value	<code>WifiInterfaceInfo</code>	

EXAMPLE

`var props = client.wifiManager().getLinkProperties("wlan0");`

isConnected Returns true if the interface has an active Wi-Fi connection.

SIGNATURE

```
boolean isConnected(String ifname) throws WiFiException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

RETURNS

NAME	TYPE	NOTES
connected	boolean	

EXAMPLE

```
boolean ok = client.wifiManager().isConnected("wlan0");
```

connect Configures and connects to a Wi-Fi network using a client config.

SIGNATURE

```
void connect(WiFiClientConfig cfg) throws WiFiException
```

PARAMETERS

NAME	TYPE	NOTES
cfg	WiFiClientConfig	

EXAMPLE

```
client.wifiManager().connect(cfg); // cfg: WiFiManager.WiFiClientConfig
```

reconnect Re-establishes the connection on an interface using the stored config.

SIGNATURE

```
void reconnect(String ifname) throws WiFiException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

EXAMPLE

```
client.wifiManager().reconnect("wlan0");
```

disconnect Disconnects from the current Wi-Fi network.

SIGNATURE

```
void disconnect(String ifname) throws WiFiException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

EXAMPLE

```
client.wifiManager().disconnect("wlan0");
```

startHotspot Configures and starts an access point from an AP config.

SIGNATURE

```
void startHotspot(WiFiAPConfig cfg) throws WiFiException
```

PARAMETERS

NAME	TYPE	NOTES
cfg	WiFiAPConfig	

EXAMPLE

```
client.wifiManager().startHotspot(cfg); // cfg: WiFiManager.WiFiAPConfig
```

stopHotspot Stops the access point on the given interface.

SIGNATURE

```
void stopHotspot(String ifname) throws WiFiException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	

EXAMPLE

```
client.wifiManager().stopHotspot("wlan0");
```

scan Scans for nearby Wi-Fi networks. forceRescan=true triggers a new hardware scan (~3 s).

SIGNATURE

```
List<ScannedNetwork> scan(String ifname, boolean forceRescan) throws WifiException
```

PARAMETERS

NAME	TYPE	NOTES
ifname	String	
forceRescan	boolean	

RETURNS

NAME	TYPE	NOTES
value	List<ScannedNetwork>	

EXAMPLE

```
var nets = client.wifiManager().scan("wlan0", true);
```

GPIO Manager

General-purpose I/O pin control (and hardware PWM helpers).

Permission: GpioService/*

listPins Returns all GPIO pins available on the device.

SIGNATURE

`List<GpioPin> listPins() throws GpioException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<GpioPin></code>	

EXAMPLE

`var pins = client.gpioManager().listPins();`

getDirection Reads the current direction of a pin.

SIGNATURE

`Direction getDirection(GpioPin pin) throws GpioException`

PARAMETERS

NAME	TYPE	NOTES
pin	<code>GpioPin</code>	

RETURNS

NAME	TYPE	NOTES
value	<code>Direction</code>	

EXAMPLE

`var dir = client.gpioManager().getDirection(pin);`

setDirection Configures a pin as input or output.

SIGNATURE

```
void setDirection(GpioPin pin, Direction direction) throws GpioException
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	
direction	Direction	Direction.IN Direction.OUT

EXAMPLE

```
var pin = new GpioManager.GpioPin("GPIO17", 17, 0);
client.gpioManager().setDirection(pin, GpioManager.Direction.OUT);
```

getLevel Reads the current logic level of a pin.

SIGNATURE

```
Level getLevel(GpioPin pin) throws GpioException
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	

RETURNS

NAME	TYPE	NOTES
value	Level	

EXAMPLE

```
var level = client.gpioManager().getLevel(pin);
```

setLevel Sets the logic level of an output pin.

SIGNATURE

```
void setLevel(GpioPin pin, Level level) throws GpioException
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	
level	Level	Level.LOW Level.HIGH

EXAMPLE

```
client.gpioManager().setLevel(pin, GpioManager.Level.HIGH);
```

PWM Manager

PWM channel configuration and control.

Permission: PwmService/*

listChannels Returns all available PWM channels.

SIGNATURE

`List<PwmChannel> listChannels() throws PwmException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<PwmChannel></code>	

EXAMPLE

`var channels = client.pwmManager().listChannels();`

getProperties Reads the current configuration of a PWM channel.

SIGNATURE

`PwmProperties getProperties(PwmChannel channel) throws PwmException`

PARAMETERS

NAME	TYPE	NOTES
channel	<code>PwmChannel</code>	

RETURNS

NAME	TYPE	NOTES
value	<code>PwmProperties</code>	

EXAMPLE

`var props = client.pwmManager().getProperties(channel);`

setPwm Configures and starts a PWM output on the given channel.

SIGNATURE

```
void setPwm(PwmChannel channel, double dutyCycle, double frequencyHz) throws PwmException
```

PARAMETERS

NAME	TYPE	NOTES
channel	PwmChannel	
dutyCycle	double	0.0–1.0
frequencyHz	double	

EXAMPLE

```
client.pwmManager().setPwm(channel, 0.5, 1000.0); // 50% @ 1 kHz
```

stopPwm Disables PWM output on the given channel.

SIGNATURE

```
void stopPwm(PwmChannel channel) throws PwmException
```

PARAMETERS

NAME	TYPE	NOTES
channel	PwmChannel	

EXAMPLE

```
client.pwmManager().stopPwm(channel);
```

UART Manager

Serial port (UART) communication. Ports are addressed by name.

Permission: UartService/*

listPorts Returns the names of all available UART ports.

SIGNATURE

`List<String> listPorts() throws UartException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<String></code>	

EXAMPLE

`var ports = client.uartManager().listPorts();`

open Opens and configures a UART port.

SIGNATURE

`void open(UartConfig cfg) throws UartException`

PARAMETERS

NAME	TYPE	NOTES
cfg	<code>UartConfig</code>	

EXAMPLE

`client.uartManager().open(new UartManager.UartConfig(
 "ttyAMA0", 115200, 8,
 UartManager.Parity.NONE, UartManager.StopBits.ONE, UartManager.FlowControl.NONE));`

getConfig Returns the current configuration of an open port.

SIGNATURE

```
UartConfig getConfig(String port) throws UartException
```

PARAMETERS

NAME	TYPE	NOTES
port	String	

RETURNS

NAME	TYPE	NOTES
value	UartConfig	

EXAMPLE

```
var cfg = client.uartManager().getConfig("ttyAMA0");
```

write Writes raw bytes to a port. Returns the number of bytes written.

SIGNATURE

```
int write(String port, byte[] data) throws UartException
```

PARAMETERS

NAME	TYPE	NOTES
port	String	
data	byte[]	

RETURNS

NAME	TYPE	NOTES
bytesWritten	int	

EXAMPLE

```
int n = client.uartManager().write("ttyAMA0", "hello\r\n".getBytes());
```

listenAsync Streams incoming bytes via callbacks. Returns a StreamHandle whose cancel() stops the stream.

SIGNATURE

```
StreamHandle listenAsync(String port, int maxChunkSize, Consumer<byte[]> onData,
Consumer<Throwable> onError)
```

PARAMETERS

NAME	TYPE	NOTES
port	String	
maxChunkSize	int	
onData	Consumer<byte[]>	
onError	Consumer<Throwable>	

RETURNS

NAME	TYPE	NOTES
value	StreamHandle	

EXAMPLE

```
var handle = client.uartManager().listenAsync("ttyAMA0", 256,
data -> System.out.println("RX: " + new String(data)),
err -> System.err.println(err.getMessage()));
// later: handle.cancel();
```

close Closes a port.

SIGNATURE

```
void close(String port) throws UartException
```

PARAMETERS

NAME	TYPE	NOTES
port	String	

EXAMPLE

```
client.uartManager().close("ttyAMA0");
```

I2C Manager

I2C bus discovery and data transfer.

Permission: I2cService/*

listBuses Returns the indices of all available I2C buses.

SIGNATURE

```
List<Integer> listBuses() throws I2CException
```

RETURNS

NAME	TYPE	NOTES
value	List<Integer>	

EXAMPLE

```
var buses = client.i2cManager().listBuses();
```

scanBus Probes all 7-bit addresses on a bus and returns those that respond.

SIGNATURE

```
List<Integer> scanBus(int bus) throws I2CException
```

PARAMETERS

NAME	TYPE	NOTES
bus	int	

RETURNS

NAME	TYPE	NOTES
value	List<Integer>	responding addresses

EXAMPLE

```
var addrs = client.i2cManager().scanBus(1);
```

getConfig Returns the current configuration of a bus.

SIGNATURE

```
I2CConfig getConfig(int bus) throws I2CException
```

PARAMETERS

NAME	TYPE	NOTES
bus	int	

RETURNS

NAME	TYPE	NOTES
value	I2CConfig	

EXAMPLE

```
var cfg = client.i2cManager().getConfig(1);
```

setConfig Applies a configuration to a bus.

SIGNATURE

```
void setConfig(I2CConfig cfg) throws I2CException
```

PARAMETERS

NAME	TYPE	NOTES
cfg	I2CConfig	

EXAMPLE

```
client.i2cManager().setConfig(new I2CManager.I2CConfig(1, 400000, false, false));
```

transfer Performs a write, read, or write-then-read transaction. Null data = read-only; readLength 0 = write-only.

SIGNATURE

```
byte[] transfer(I2CTransferRequest req) throws I2CException
```

PARAMETERS

NAME	TYPE	NOTES
req	I2CTransferRequest	

RETURNS

NAME	TYPE	NOTES
value	byte[]	received bytes (null for write-only)

EXAMPLE

```
byte[] data = client.i2cManager().transfer(  
    new I2CManager.I2CTransferRequest(1, 0x48, new byte[]{0x00}, 2));
```

SPI Manager

SPI bus configuration and full-duplex transfer.

Permission: SpiService/*

listDevices Returns available SPI device names (e.g. "spidev0.0").

SIGNATURE

`List<String> listDevices() throws SpiException`

RETURNS

NAME	TYPE	NOTES
value	<code>List<String></code>	

EXAMPLE

`var devices = client.spiManager().listDevices();`

getConfig Returns the current configuration of a device.

SIGNATURE

`SpiConfig getConfig(int bus, int chipSelect) throws SpiException`

PARAMETERS

NAME	TYPE	NOTES
bus	<code>int</code>	
chipSelect	<code>int</code>	

RETURNS

NAME	TYPE	NOTES
value	<code>SpiConfig</code>	

EXAMPLE

`var cfg = client.spiManager().getConfig(0, 0);`

setConfig Applies a configuration to a device.

SIGNATURE

```
void setConfig(SpiConfig cfg) throws SpiException
```

PARAMETERS

NAME	TYPE	NOTES
cfg	SpiConfig	

EXAMPLE

```
client.spiManager().setConfig(cfg); // cfg: SpiManager.SpiConfig
```

transfer Performs a full-duplex transfer. readLength 0 = write-only.

SIGNATURE

```
byte[] transfer(int bus, int chipSelect, byte[] dataOut, int readLength) throws SpiException
```

PARAMETERS

NAME	TYPE	NOTES
bus	int	
chipSelect	int	
dataOut	byte[]	
readLength	int	

RETURNS

NAME	TYPE	NOTES
value	byte[]	received MISO bytes

EXAMPLE

```
byte[] rx = client.spiManager().transfer(0, 0, new byte[]{(byte) 0x9F}, 3);
```

Camera Manager

V4L2 camera capture and streaming. lock() returns a LockedCamera handle for capture/stream.

Permission: CameraService/*

listDevices Returns all V4L2 video devices available on the system.

SIGNATURE

```
List<DeviceInfo> listDevices() throws CameraException
```

RETURNS

NAME	TYPE	NOTES
value	List<DeviceInfo>	

EXAMPLE

```
var devices = client.cameraManager().listDevices();
```

getDeviceInfo Returns metadata for a specific camera device.

SIGNATURE

```
DeviceInfo getDeviceInfo(String deviceId) throws CameraException
```

PARAMETERS

NAME	TYPE	NOTES
deviceId	String	e.g. "/dev/video0"

RETURNS

NAME	TYPE	NOTES
value	DeviceInfo	

EXAMPLE

```
var info = client.cameraManager().getDeviceInfo("/dev/video0");
```

lock Acquires an exclusive lock on a camera and returns a LockedCamera handle (AutoCloseable).

SIGNATURE

LockedCamera lock(String deviceId, String clientId) throws CameraException

PARAMETERS

NAME	TYPE	NOTES
deviceId	String	
clientId	String	

RETURNS

NAME	TYPE	NOTES
value	LockedCamera	

EXAMPLE

```
try (var cam = client.cameraManager().lock("/dev/video0", "my-app")) {
    var img = cam.captureImage(1280, 720, "mjpeg");
}
```

captureImage Captures a single frame. Called on a LockedCamera handle.

SIGNATURE

CaptureResult captureImage(int width, int height, String format) throws CameraException

PARAMETERS

NAME	TYPE	NOTES
width	int	
height	int	
format	String	"mjpeg" or "yuyv"

RETURNS

NAME	TYPE	NOTES
value	CaptureResult	

EXAMPLE

```
var img = cam.captureImage(1280, 720, "mjpeg");
```

streamFramesAsync Streams frames via callbacks at the requested FPS and resolution. Returns a StreamHandle.

SIGNATURE

```
StreamHandle streamFramesAsync(int fps, int width, int height, Consumer<Frame> onFrame,  
Consumer<Throwable> onError)
```

PARAMETERS

NAME	TYPE	NOTES
fps	int	
width	int	
height	int	
onFrame	Consumer<Frame>	
onError	Consumer<Throwable>	

RETURNS

NAME	TYPE	NOTES
value	StreamHandle	

EXAMPLE

```
var handle = cam.streamFramesAsync(30, 1280, 720,  
    frame -> process(frame.data()),  
    err -> System.err.println(err.getMessage()));
```

Ai Manager

On-device AI model lifecycle and inference. loadModel returns an AiModel handle.

Permission: `AiService/*`

listModels Returns metadata for all currently loaded models.

SIGNATURE

```
List<ModelInfo> listModels()
```

RETURNS

NAME	TYPE	NOTES
value	<code>List<ModelInfo></code>	

EXAMPLE

```
var models = client.aiManager().listModels();
```

loadModel Loads a model already present on the device filesystem. Returns an AiModel handle.

SIGNATURE

```
AiModel loadModel(String modelId, String modelPath, ModelBackend backend, ExecutionMode execution) throws AiException
```

PARAMETERS

NAME	TYPE	NOTES
modelId	<code>String</code>	
modelPath	<code>String</code>	Absolute path on the device
backend	<code>ModelBackend</code>	
execution	<code>ExecutionMode</code>	

RETURNS

NAME	TYPE	NOTES
value	<code>AiModel</code>	

EXAMPLE

```
var model = client.aiManager().loadModel("detector", "/models/yolo.onnx", AiManager.ModelBackend.ONNX, AiManager.ExecutionMode.CPU);
```

uploadAndLoadModel Streams a local model file to the device in chunks then loads it. Returns an AiModel handle.

SIGNATURE

```
AiModel uploadAndLoadModel(String modelId, File localFile, ModelBackend backend,
    ExecutionMode execution) throws AiException, IOException
```

PARAMETERS

NAME	TYPE	NOTES
modelId	String	
localFile	File	
backend	ModelBackend	
execution	ExecutionMode	

RETURNS

NAME	TYPE	NOTES
value	AiModel	

EXAMPLE

```
var model = client.aiManager().uploadAndLoadModel(
    "detector", new File("./yolo.onnx"),
    AiManager.ModelBackend.ONNX, AiManager.ExecutionMode.CPU);
```

unload Frees the model from the inference backend. Called on an AiModel handle.

SIGNATURE

```
void unload() throws AiException
```

EXAMPLE

```
model.unload();
```

isLoading Returns whether the model is currently loaded.

SIGNATURE

```
boolean isLoading() throws AiException
```

RETURNS

NAME	TYPE	NOTES
value	boolean	

EXAMPLE

```
boolean loaded = model.isLoading();
```

infer Runs a single synchronous forward pass.

SIGNATURE

```
InferResult infer(byte[] inputData, int[] inputShape, TensorDataType dtype) throws AiException
```

PARAMETERS

NAME	TYPE	NOTES
inputData	byte[]	Raw little-endian bytes
inputShape	int[]	
dtype	TensorDataType	

RETURNS

NAME	TYPE	NOTES
value	InferResult	

EXAMPLE

```
var result = model.infer(inputData, new int[]{1, 3, 640, 640},  
    AiManager.TensorDataType.FLOAT32);
```

Firewall Manager

Firewall zone and traffic rule management.

Permission: FirewallService/*

listZones

Returns all firewall zones.

SIGNATURE

```
List<Zone> listZones() throws FirewallException
```

RETURNS

NAME	TYPE	NOTES
value	List<Zone>	

EXAMPLE

```
var zones = client.firewallManager().listZones();
```

addZone

Creates a firewall zone.

SIGNATURE

```
void addZone(String name, List<String> interfaces, Policy inputPolicy, Policy outputPolicy, boolean masquerade) throws FirewallException
```

PARAMETERS

NAME	TYPE	NOTES
name	String	
interfaces	List<String>	
inputPolicy	Policy	
outputPolicy	Policy	
masquerade	boolean	

EXAMPLE

```
client.firewallManager().addZone("lan", List.of("eth0"), FirewallManager.Policy.ACCEPT, FirewallManager.Policy.ACCEPT, false);
```

removeZone Removes a firewall zone by name.

SIGNATURE

```
void removeZone(String name) throws FirewallException
```

PARAMETERS

NAME	TYPE	NOTES
name	String	

EXAMPLE

```
client.firewallManager().removeZone("lan");
```

listRules Returns all firewall traffic rules.

SIGNATURE

```
List<FirewallRule> listRules() throws FirewallException
```

RETURNS

NAME	TYPE	NOTES
value	List<FirewallRule>	

EXAMPLE

```
var rules = client.firewallManager().listRules();
```

addRule Adds a firewall traffic rule.

SIGNATURE

```
void addRule(FirewallRule rule) throws FirewallException
```

PARAMETERS

NAME	TYPE	NOTES
rule	FirewallRule	

EXAMPLE

```
client.firewallManager().addRule(rule); // rule: FirewallManager.FirewallRule
```

removeRule Removes a firewall rule by ID.

SIGNATURE

```
void removeRule(String id) throws FirewallException
```

PARAMETERS

NAME	TYPE	NOTES
id	String	

EXAMPLE

```
client.firewallManager().removeRule(ruleId);
```

flushRules Removes all firewall traffic rules.

SIGNATURE

```
void flushRules() throws FirewallException
```

EXAMPLE

```
client.firewallManager().flushRules();
```

Vpn Manager

VPN tunnel management (WireGuard, OpenVPN, IPSec).

Permission: VPNService/*

getCapabilities Returns which VPN backends the device supports.

SIGNATURE

VpnCapabilities getCapabilities() throws VpnException

RETURNS

NAME	TYPE	NOTES
value	VpnCapabilities	

EXAMPLE

var caps = client.vpnManager().getCapabilities();

listProfiles Returns all stored VPN profiles.

SIGNATURE

List<VpnProfile> listProfiles() throws VpnException

RETURNS

NAME	TYPE	NOTES
value	List<VpnProfile>	

EXAMPLE

var profiles = client.vpnManager().listProfiles();

applyWireGuard Creates or updates a WireGuard profile from a .conf payload. Returns the profile ID.

SIGNATURE

```
String applyWireGuard(String displayName, byte[] configData, boolean autoConnect) throws VpnException
```

PARAMETERS

NAME	TYPE	NOTES
displayName	String	
configData	byte[]	
autoConnect	boolean	

RETURNS

NAME	TYPE	NOTES
profileId	String	

EXAMPLE

```
String id = client.vpnManager().applyWireGuard("office", confBytes, true);
```

applyOpenVpn Creates or updates an OpenVPN profile from an .ovpn payload. Returns the profile ID.

SIGNATURE

```
String applyOpenVpn(String displayName, byte[] configData, boolean autoConnect) throws VpnException
```

PARAMETERS

NAME	TYPE	NOTES
displayName	String	
configData	byte[]	
autoConnect	boolean	

RETURNS

NAME	TYPE	NOTES
profileId	String	

EXAMPLE

```
String id = client.vpnManager().applyOpenVpn("home", ovpnBytes, false);
```

removeProfile Removes a VPN profile by ID.

SIGNATURE

```
boolean removeProfile(String profileId) throws VpnException
```

PARAMETERS

NAME	TYPE	NOTES
profileId	String	

RETURNS

NAME	TYPE	NOTES
removed	boolean	

EXAMPLE

```
boolean ok = client.vpnManager().removeProfile(id);
```

connect Brings up the tunnel for a profile. Returns the session ID.

SIGNATURE

```
String connect(String profileId) throws VpnException
```

PARAMETERS

NAME	TYPE	NOTES
profileId	String	

RETURNS

NAME	TYPE	NOTES
sessionId	String	

EXAMPLE

```
String session = client.vpnManager().connect(id);
```

disconnect Tears down the tunnel for a profile.

SIGNATURE

```
boolean disconnect(String profileId) throws VpnException
```

PARAMETERS

NAME	TYPE	NOTES
profileId	String	

RETURNS

NAME	TYPE	NOTES
ok	boolean	

EXAMPLE

```
boolean ok = client.vpnManager().disconnect(id);
```

getStatus Returns the current tunnel session and provider details.

SIGNATURE

```
StatusResult getStatus() throws VpnException
```

RETURNS

NAME	TYPE	NOTES
value	StatusResult	

EXAMPLE

```
var status = client.vpnManager().getStatus();
```

isConnected Returns true if a VPN tunnel is currently active.

SIGNATURE

```
boolean isConnected() throws VpnException
```

RETURNS

NAME	TYPE	NOTES
connected	boolean	

EXAMPLE

```
boolean up = client.vpnManager().isConnected();
```

watchEvents Streams VPN events via callbacks. Returns a StreamHandle.

SIGNATURE

```
StreamHandle watchEvents(String profileId, Consumer<VPNEvent> onEvent,  
Consumer<Throwable> onError)
```

PARAMETERS

NAME	TYPE	NOTES
profileId	String	
onEvent	Consumer<VPNEvent>	
onError	Consumer<Throwable>	

RETURNS

NAME	TYPE	NOTES
value	StreamHandle	

EXAMPLE

```
var h = client.vpnManager().watchEvents(id, ev -> log(ev), err -> log(err));
```

App Hub Manager

Gravity portal WebUI registry and reverse-proxy routing.

Permission: AppHubService/*

registerWebUI Registers a web UI served on host:port under a route prefix.

SIGNATURE

`void registerWebUI(String host, int port, String route) throws AppHubException`

PARAMETERS

NAME	TYPE	NOTES
host	String	
port	int	
route	String	

EXAMPLE

`client.appHubManager().registerWebUI("127.0.0.1", 9033, "/myapp");`

unregisterService Removes this app's service registration.

SIGNATURE

`void unregisterService() throws AppHubException`

EXAMPLE

`client.appHubManager().unregisterService();`

getService Returns the registration details for a service by ID.

SIGNATURE

```
Service getService(String serviceId) throws AppHubException
```

PARAMETERS

NAME	TYPE	NOTES
serviceId	String	

RETURNS

NAME	TYPE	NOTES
value	Service	

EXAMPLE

```
var svc = client.appHubManager().getService(serviceId);
```

listServices Returns all registered services on the portal.

SIGNATURE

```
List<Service> listServices() throws AppHubException
```

RETURNS

NAME	TYPE	NOTES
value	List<Service>	

EXAMPLE

```
var services = client.appHubManager().listServices();
```

addRoute Adds a URL path prefix to this service's routing table.

SIGNATURE

```
void addRoute(String path) throws AppHubException
```

PARAMETERS

NAME	TYPE	NOTES
path	String	

EXAMPLE

```
client.appHubManager().addRoute("/api");
```

removeRoute Removes a URL path prefix from this service's routing table.

SIGNATURE

```
void removeRoute(String path) throws AppHubException
```

PARAMETERS

NAME	TYPE	NOTES
path	String	

EXAMPLE

```
client.appHubManager().removeRoute("/api");
```

getRoutingTable Returns the full portal routing table.

SIGNATURE

```
List<RoutingEntry> getRoutingTable() throws AppHubException
```

RETURNS

NAME	TYPE	NOTES
value	List<RoutingEntry>	

EXAMPLE

```
var routes = client.appHubManager().getRoutingTable();
```

watchServices Streams ServiceEvent messages via callbacks. Returns a StreamHandle.

SIGNATURE

```
StreamHandle watchServices(Consumer<ServiceEvent> onEvent, Consumer<Throwable> onError)
```

PARAMETERS

NAME	TYPE	NOTES
onEvent	Consumer<ServiceEvent>	
onError	Consumer<Throwable>	

RETURNS

NAME	TYPE	NOTES
value	StreamHandle	

EXAMPLE

```
var h = client.appHubManager().watchServices(ev -> log(ev), err -> log(err));
```

Event Manager

System lifecycle event subscriptions.

Permission: EventService/*

subscribeAsync

Subscribes to system events via callbacks. Returns a StreamHandle whose cancel() ends the subscription.

SIGNATURE

```
StreamHandle subscribeAsync(Consumer<Event> onEvent, Consumer<Throwable> onError)
```

PARAMETERS

NAME	TYPE	NOTES
onEvent	Consumer<Event>	
onError	Consumer<Throwable>	

RETURNS

NAME	TYPE	NOTES
value	StreamHandle	

EXAMPLE

```
var sub = client.eventManager().subscribeAsync(  
    e -> System.out.printf("event: %s payload=%s%n", e.type(), e.payload()),  
    err -> System.err.println(err.getMessage());  
// later: sub.cancel();
```

Development Manager

Real-time application log streaming.

Permission: DevelopmentService/*

subscribeLogsAsync

Streams application logs via callbacks, filtered by app, tag and minimum level. Returns a StreamHandle.

SIGNATURE

```
StreamHandle subscribeLogsAsync(String app, String tag, LogLevel minLevel,
    Consumer<LogEntry> onEntry, Consumer<Throwable> onError)
```

PARAMETERS

NAME	TYPE	NOTES
app	String	
tag	String	
minLevel	LogLevel	
onEntry	Consumer<LogEntry>	
onError	Consumer<Throwable>	

RETURNS

NAME	TYPE	NOTES
value	StreamHandle	

EXAMPLE

```
var logs = client.developmentManager().subscribeLogsAsync(
    "", "", DevelopmentManager.LogLevel.INFO,
    e -> System.out.printf("[%s] %s: %s%n", e.level(), e.tag(), e.message()),
    err -> System.err.println(err.getMessage()));
```

Update Manager

OTA firmware update and factory reset.

Permission: UpdateService/*

update Uploads a firmware archive (streamed in chunks) and applies it.

SIGNATURE

```
void update(String filePath) throws UpdateException, IOException
```

PARAMETERS

NAME	TYPE	NOTES
filePath	String	

EXAMPLE

```
client.updateManager().update("/path/to/firmware.orbit");
```

factoryReset Restores the device to factory defaults. Erases all user data.

SIGNATURE

```
void factoryReset() throws UpdateException
```

EXAMPLE

```
client.updateManager().factoryReset();
```

Power Manager

Device power control.

Permission: `PowerService/*`

reboot Reboots the device.

SIGNATURE

`void reboot(boolean force, String reason) throws PowerException`

PARAMETERS

NAME	TYPE	NOTES
force	boolean	
reason	String	

EXAMPLE

`client.powerManager().reboot(false, "scheduled maintenance");`

shutdown Powers off the device.

SIGNATURE

`void shutdown(boolean force, String reason) throws PowerException`

PARAMETERS

NAME	TYPE	NOTES
force	boolean	
reason	String	

EXAMPLE

`client.powerManager().shutdown(false, "");`

Types

Complex objects returned or accepted by the SDK methods.

GpioPin

Identifies a GPIO line by chip index and line offset (record).

FIELDS

NAME	TYPE	NOTES
name	String	Human-readable pin name (e.g. "GPIO17")
line	int	Line offset within the chip
gpiochip	int	Chip index (0 = /dev/gpiochip0)

Direction

GPIO pin direction enum.

FIELDS

NAME	TYPE	NOTES
IN	enum	Input
OUT	enum	Output

Level

GPIO pin logic level enum.

FIELDS

NAME	TYPE	NOTES
LOW	enum	
HIGH	enum	

UartConfig

Configuration used to open a UART port (record).

FIELDS		
NAME	TYPE	NOTES
port	String	Port name (e.g. "ttyAMA0")
baudrate	int	Baud rate (e.g. 115200)
dataBits	int	5, 6, 7 or 8
parity	Parity	Parity mode
stopBits	StopBits	Stop bits
flowControl	FlowControl	Flow control mode

Parity

UART parity mode enum.

FIELDS		
NAME	TYPE	NOTES
NONE	enum	
EVEN	enum	
ODD	enum	

StopBits

UART stop bits enum.

FIELDS		
NAME	TYPE	NOTES
ONE	enum	
TWO	enum	

FlowControl

UART flow control enum.

FIELDS		
NAME	TYPE	NOTES
NONE	enum	
HARDWARE	enum	RTS/CTS
SOFTWARE	enum	XON/XOFF

I2CConfig

I²C bus configuration (record).

FIELDS

NAME	TYPE	NOTES
bus	<code>int</code>	Bus number
clockHz	<code>int</code>	Clock frequency in Hz
tenBitAddr	<code>boolean</code>	10-bit addressing
clockStretching	<code>boolean</code>	Allow clock stretching

I2CTransferRequest

Generic I²C transfer (record). data only = write, readLength only = read, both = write-then-read.

FIELDS

NAME	TYPE	NOTES
bus	<code>int</code>	Bus number
address	<code>int</code>	7-bit device address (e.g. 0x48)
data	<code>byte[]</code>	Bytes to write (null for read-only)
readLength	<code>int</code>	Bytes to read (0 for write-only)

SpiConfig

SPI device configuration (record).

FIELDS

NAME	TYPE	NOTES
bus	<code>int</code>	Bus index
chipSelect	<code>int</code>	Chip select
maxSpeedHz	<code>int</code>	Max clock speed in Hz
bitsPerWord	<code>int</code>	Bits per word (typically 8)
mode	<code>SpiMode</code>	Clock polarity/phase
lsbFirst	<code>boolean</code>	LSB-first bit order

SpiMode

SPI mode (clock polarity + phase) enum.

FIELDS

NAME	TYPE	NOTES
MODE0	enum	
MODE1	enum	
MODE2	enum	
MODE3	enum	

PwmChannel

Identifies a PWM hardware channel (record).

FIELDS

NAME	TYPE	NOTES
channel	int	Channel index
name	String	Channel name

PwmProperties

Current PWM channel state (record).

FIELDS

NAME	TYPE	NOTES
channel	PwmChannel	
enabled	boolean	
dutyCycle	double	0.0–1.0
frequencyHz	double	

WiFiMode

Wi-Fi operating mode enum.

FIELDS

NAME	TYPE	NOTES
UNKNOWN	enum	
AP	enum	
CLIENT	enum	
AP_CLIENT	enum	
DISABLED	enum	

FirewallRule

A firewall traffic rule (record).

FIELDS

NAME	TYPE	NOTES
id	String	Server-generated rule ID
srcZone	String	
dstZone	String	
protocol	Protocol	ANY TCP UDP ICMP
srcIp	String	
destPort	int	
action	Policy	ACCEPT DROP REJECT
comment	String	

Zone

Firewall zone definition (record).

FIELDS

NAME	TYPE	NOTES
name	String	
interfaces	List<String>	
inputPolicy	Policy	
outputPolicy	Policy	
masquerade	boolean	

Protocol

Firewall protocol filter enum.

FIELDS

NAME	TYPE	NOTES
ANY	enum	
TCP	enum	
UDP	enum	
ICMP	enum	

Policy

Firewall zone policy enum.

FIELDS

NAME	TYPE	NOTES
ACCEPT	enum	
DROP	enum	
REJECT	enum	

ModelInfo

Metadata for a loaded model (record).

FIELDS

NAME	TYPE	NOTES
modelId	String	
version	String	
backend	ModelBackend	
loadedAtUnix	long	

TensorInfo

Describes a single input or output tensor (record).

FIELDS

NAME	TYPE	NOTES
name	String	
dtype	TensorDataType	
shape	List<Integer>	

InferResult

Result of a synchronous inference call (record).

FIELDS

NAME	TYPE	NOTES
outputs	List<byte[]>	Raw output tensors
tensors	List<TensorInfo>	Output tensor schema

ModelBackend

AI model inference backend enum.

FIELDS

NAME	TYPE	NOTES
ONNX	enum	
TFLITE	enum	

ExecutionMode

Compute path used during inference.

FIELDS

NAME	TYPE	NOTES
CPU	enum	
GPU	enum	
NPU	enum	

TensorDataType

Raw byte encoding of tensor elements.

FIELDS

NAME	TYPE	NOTES
FLOAT32	enum	
INT32	enum	
INT64	enum	
UINT8	enum	

Frame

A single video frame from a camera stream (record).

FIELDS

NAME	TYPE	NOTES
data	byte[]	
timestamp	long	
width	int	
height	int	
format	String	

Python API Reference

Orbit OS SDK · Version 26.0.1 · Generated 2026-09-18

CONTENTS

Auth Manager	2	Camera Manager	47
System Manager	3	AI Manager	51
Package Manager	18	Firewall Manager	56
Ethernet Manager	21	VPN Manager	60
WiFi Manager	24	App Hub Manager	66
GPIO Manager	31	Event Manager	70
PWM Manager	34	Development Manager	71
UART Manager	36	Update Manager	73
I2C Manager	39	Power Manager	74
SPI Manager	43	Types	75

Auth Manager

Local device authentication via Gravity.

login Authenticate with username and password. Returns (session_token, expires_at).

SIGNATURE

```
def login(self, username: str, password: str) -> tuple[str, int]:
```

PARAMETERS

NAME	TYPE	NOTES
username	str	
password	str	

RETURNS

NAME	TYPE	NOTES
token	str	
expires_at	int	

EXAMPLE

```
token, expires_at = c.auth_manager.login("admin", "secret")
```

logout Invalidates a session token.

SIGNATURE

```
def logout(self, token: str) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
token	str	

EXAMPLE

```
c.auth_manager.logout(token)
```

System Manager

Device identity, hardware info, OS metadata, and live metrics.

get_api_version Returns the API version and revision running on the device.

SIGNATURE

```
def get_api_version(self) -> tuple[int, int]:
```

RETURNS

NAME	TYPE	NOTES
version	int	
revision	int	

EXAMPLE

```
version, revision = c.system_manager.get_api_version()
```

get_device_name Returns the configured device name.

SIGNATURE

```
def get_device_name(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
name = c.system_manager.get_device_name()
```

get_soc_model Returns the SoC model identifier.

SIGNATURE

```
def get_soc_model(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
model = c.system_manager.get_soc_model()
```

get_soc_vendor Returns the SoC vendor name.

SIGNATURE

```
def get_soc_vendor(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
vendor = c.system_manager.get_soc_vendor()
```

get_board_model Returns the board model identifier.

SIGNATURE

```
def get_board_model(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
model = c.system_manager.get_board_model()
```

get_board_vendor Returns the board vendor name.

SIGNATURE

```
def get_board_vendor(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
vendor = c.system_manager.get_board_vendor()
```

get_hardware_version Returns the hardware revision string.

SIGNATURE

```
def get_hardware_version(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
ver = c.system_manager.get_hardware_version()
```

get_hardware_model Returns the hardware model identifier.

SIGNATURE

```
def get_hardware_model(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
model = c.system_manager.get_hardware_model()
```

get_system_uuid Returns the system UUID.

SIGNATURE

```
def get_system_uuid(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
uuid = c.system_manager.get_system_uuid()
```

get_board_serial Returns the board serial number.

SIGNATURE

```
def get_board_serial(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
serial = c.system_manager.get_board_serial()
```

get_cpu_serial Returns the CPU serial number.

SIGNATURE

```
def get_cpu_serial(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
serial = c.system_manager.get_cpu_serial()
```

get_machine_id Returns the machine ID.

SIGNATURE

```
def get_machine_id(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
mid = c.system_manager.get_machine_id()
```

get_architecture Returns the CPU architecture (e.g. "arm64", "amd64").

SIGNATURE

```
def get_architecture(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
arch = c.system_manager.get_architecture()
```

get_cpu_model Returns the CPU model string.

SIGNATURE

```
def get_cpu_model(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
model = c.system_manager.get_cpu_model()
```

get_total_ram Returns total RAM in bytes.

SIGNATURE

```
def get_total_ram(self) -> int:
```

RETURNS

NAME	TYPE	NOTES
value	int	bytes

EXAMPLE

```
ram = c.system_manager.get_total_ram()
```

get_cpu_cores Returns the number of physical CPU cores.

SIGNATURE

```
def get_cpu_cores(self) -> int:
```

RETURNS

NAME	TYPE	NOTES
value	int	

EXAMPLE

```
cores = c.system_manager.get_cpu_cores()
```

get_cpu_threads Returns the number of logical CPU threads.

SIGNATURE

```
def get_cpu_threads(self) -> int:
```

RETURNS

NAME	TYPE	NOTES
value	int	

EXAMPLE

```
threads = c.system_manager.get_cpu_threads()
```

get_cpu_min_mhz Returns the minimum CPU frequency in MHz.

SIGNATURE

```
def get_cpu_min_mhz(self) -> float:
```

RETURNS

NAME	TYPE	NOTES
value	float	

EXAMPLE

```
min_mhz = c.system_manager.get_cpu_min_mhz()
```

get_cpu_max_mhz Returns the maximum CPU frequency in MHz.

SIGNATURE

```
def get_cpu_max_mhz(self) -> float:
```

RETURNS

NAME	TYPE	NOTES
value	float	

EXAMPLE

```
max_mhz = c.system_manager.get_cpu_max_mhz()
```

get_os_name Returns the OS name.

SIGNATURE

```
def get_os_name(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
name = c.system_manager.get_os_name()
```

get_os_version Returns the OS version string.

SIGNATURE

```
def get_os_version(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
ver = c.system_manager.get_os_version()
```

get_kernel_version Returns the kernel version string.

SIGNATURE

```
def get_kernel_version(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
ver = c.system_manager.get_kernel_version()
```

get_distro Returns the Linux distribution name.

SIGNATURE

```
def get_distro(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
distro = c.system_manager.get_distro()
```

get_distro_version Returns the distribution version string.

SIGNATURE

```
def get_distro_version(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
ver = c.system_manager.get_distro_version()
```

get_runtime_version Returns the Orbit runtime version.

SIGNATURE

```
def get_runtime_version(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
ver = c.system_manager.get_runtime_version()
```

get_os_revision Returns the logical OS revision string.

SIGNATURE

```
def get_os_revision(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
rev = c.system_manager.get_os_revision()
```

get_metrics Returns live CPU usage, RAM and storage stats.

SIGNATURE

```
def get_metrics(self) -> MetricsInfoResponse:
```

RETURNS

NAME	TYPE	NOTES
MetricsInfoResponse		proto with cpu_usage, ram_used, disk_used

EXAMPLE

```
metrics = c.system_manager.get_metrics()  
# metrics.cpu_usage, metrics.ram_used, metrics.disk_used
```

enable_dev_mode Enables developer mode on the device.

SIGNATURE

```
def enable_dev_mode(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.enable_dev_mode()
```

enable_ssh_server Starts the SSH server on the device.

SIGNATURE

```
def enable_ssh_server(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.enable_ssh_server()
```

disable_ssh_server Stops the SSH server on the device.

SIGNATURE

```
def disable_ssh_server(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.disable_ssh_server()
```

is_ssh_server_enabled Returns True if the SSH server is currently running.

SIGNATURE

```
def is_ssh_server_enabled(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	

EXAMPLE

```
enabled = c.system_manager.is_ssh_server_enabled()
```

disable_dev_mode Disables developer mode on the device.

SIGNATURE

```
def disable_dev_mode(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.disable_dev_mode()
```

is_dev_mode_enabled Returns True if developer mode is currently active.

SIGNATURE

```
def is_dev_mode_enabled(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	

EXAMPLE

```
enabled = c.system_manager.is_dev_mode_enabled()
```

enable_reboot_on_failure Enables automatic reboot when the app crashes.

SIGNATURE

```
def enable_reboot_on_failure(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.enable_reboot_on_failure()
```

disable_reboot_on_failure Disables automatic reboot on crash.

SIGNATURE

```
def disable_reboot_on_failure(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.disable_reboot_on_failure()
```

is_reboot_on_failure_enabled Returns True if automatic reboot on crash is enabled.

SIGNATURE

```
def is_reboot_on_failure_enabled(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	

EXAMPLE

```
enabled = c.system_manager.is_reboot_on_failure_enabled()
```

allow_untrusted_apps Permits installation of apps not signed by Orbit.

SIGNATURE

```
def allow_untrusted_apps(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.allow_untrusted_apps()
```

disallow_untrusted_apps Revokes permission for unsigned app installation.

SIGNATURE

```
def disallow_untrusted_apps(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.disallow_untrusted_apps()
```

is_untrusted_apps_allowed Returns True if unsigned app installation is permitted.

SIGNATURE

```
def is_untrusted_apps_allowed(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
enabled	bool	

EXAMPLE

```
ok = c.system_manager.is_untrusted_apps_allowed()
```

get_api_version_info Returns the API version and revision as a formatted string.

SIGNATURE

```
def get_api_version_info(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	e.g. "v26.0000.0000"

EXAMPLE

```
info = c.system_manager.get_api_version_info()
```

get_runtime_build_date Returns the build date of the Orbit runtime.

SIGNATURE

```
def get_runtime_build_date(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
date = c.system_manager.get_runtime_build_date()
```

get_build_date Alias for GetRuntimeBuildDate.

SIGNATURE

```
def get_build_date(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
date = c.system_manager.get_build_date()
```

get_build_version Alias for GetOsRevision — returns the OS revision string.

SIGNATURE

```
def get_build_version(self) -> str:
```

RETURNS

NAME	TYPE	NOTES
value	str	

EXAMPLE

```
ver = c.system_manager.get_build_version()
```

attach Attaches the client session to the runtime.

SIGNATURE

```
def attach(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.system_manager.attach()
```

Package Manager

Install, query, and remove ORB application packages.

get_installed_packages Returns all packages currently installed on the device.

SIGNATURE

```
def get_installed_packages(self) -> list[InstalledPackage]:
```

RETURNS

NAME	TYPE	NOTES
packages	list[InstalledPackage]	

EXAMPLE

```
pkgs = c.package_manager.get_installed_packages()
```

remove_package Uninstalls a package by its package ID.

SIGNATURE

```
def remove_package(self, package_id: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
package_id	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.package_manager.remove_package("com.example.myapp")
```

install_update_package Returns a client-streaming call for uploading an ORB archive.

SIGNATURE

```
def install_update_package(self, timeout: float = 60.0):
```

PARAMETERS

NAME	TYPE	NOTES
timeout	float	default 60.0 s

RETURNS

NAME	TYPE	NOTES
stream		gRPC client-streaming stub — call send(PackageChunk) then close_and_recv()

EXAMPLE

```
stream = c.package_manager.install_update_package()
with open("app.orb", "rb") as f:
    while chunk := f.read(256 * 1024):
        stream.send(PackageChunk(data=chunk))
resp = stream.close_and_recv()
```

list_installed_packages

Returns all packages currently installed on the device (canonical name; GetInstalledPackages is an alias).

SIGNATURE

```
def list_installed_packages(self) -> list[InstalledPackage]:
```

RETURNS

NAME	TYPE	NOTES
packages	list[InstalledPackage]	

EXAMPLE

```
pkgs = c.package_manager.list_installed_packages()
```

install_package Installs an ORB package from a local file path (streams the file, then applies it).

SIGNATURE

```
def install_package(self, local_path: str, chunk_size: int = 262144, timeout: float = 120.0) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
local_path	str	Local path to the .orb file
chunk_size	int	default 256 KiB
timeout	float	default 120.0 s

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.package_manager.install_package("./myapp.orb")
```

Ethernet Manager

Ethernet interface configuration and link properties.

list_ethernet_interfaces Returns all discovered Ethernet interfaces.

SIGNATURE

```
def list_ethernet_interfaces(self) -> list[EthernetInterface]:
```

RETURNS

NAME	TYPE	NOTES
interfaces	list[EthernetInterface]	

EXAMPLE

```
ifaces = c.ethernet_manager.list_ethernet_interfaces()
```

is_ethernet_connected Returns True if the interface has an active link.

SIGNATURE

```
def is_ethernet_connected(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
connected	bool	

EXAMPLE

```
up = c.ethernet_manager.is_ethernet_connected("eth0")
```

get_ethernet_link_properties Returns full link properties, or None if unavailable.

SIGNATURE

```
def get_ethernet_link_properties(self, interface_name: str) -> EthernetLinkProperties | None:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
properties	EthernetLinkProperties None	

EXAMPLE

```
props = c.ethernet_manager.get_ethernet_link_properties("eth0")
```

set_ethernet_config Applies static or DHCP IP configuration to an interface.

SIGNATURE

```
def set_ethernet_config(self, interface_name: str, dhcp_enable: bool, ...) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	
dhcp_enable	bool	
ipv4_address	str	
ipv4_gateway	str	
ipv4_dns	list[str]	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.ethernet_manager.set_ethernet_config("eth0", dhcp_enable=True)
```

enable_ethernet Brings up an Ethernet interface.

SIGNATURE

```
def enable_ethernet(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.ethernet_manager.enable_ethernet("eth0")
```

disable_ethernet Shuts down an Ethernet interface.

SIGNATURE

```
def disable_ethernet(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.ethernet_manager.disable_ethernet("eth0")
```

WiFi Manager

Wi-Fi client and access-point configuration.

list_wifi_interfaces Returns all discovered Wi-Fi interfaces.

SIGNATURE

```
def list_wifi_interfaces(self) -> list[WiFiInterface]:
```

RETURNS

NAME	TYPE	NOTES
interfaces	list[WiFiInterface]	

EXAMPLE

```
ifaces = c.wifi_manager.list_wifi_interfaces()
```

scan_wifi Performs a passive scan and returns visible networks.

SIGNATURE

```
def scan_wifi(self, interface_name: str) -> list[WiFiNetwork]:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
networks	list[WiFiNetwork]	

EXAMPLE

```
nets = c.wifi_manager.scan_wifi("wlan0")
```

set_client_config Configures the Wi-Fi client association and IP settings.

SIGNATURE

```
def set_client_config(self, interface_name: str, ssid: str, ...) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	
ssid	str	
password	str	
security	str	
dhcp_enable	bool	
ipv4_address	str	
ipv4_gateway	str	
ipv4_dns	list[str]	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.wifi_manager.set_client_config("wlan0", ssid="MyNet", password="secret",  
security="WPA2", dhcp_enable=True, ipv4_address="", ipv4_gateway="", ipv4_dns=[])
```

connect Initiates a connection to the configured SSID.

SIGNATURE

```
def connect(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.wifi_manager.connect("wlan0")
```

disconnect Disconnects from the current SSID.

SIGNATURE

```
def disconnect(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.wifi_manager.disconnect("wlan0")
```

set_ap_config Configures the software access point.

SIGNATURE

```
def set_ap_config(self, interface_name: str, ssid: str, ...) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	
ssid	str	
password	str	
security	str	
ipv4_address	str	
channel	int	
band	str	
hidden	bool	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.wifi_manager.set_ap_config("wlan0", ssid="HotSpot", password="pass",  
security="WPA2", ipv4_address="192.168.4.1", channel=6, band="2.4GHz", hidden=False)
```

start_ap Starts the software access point.

SIGNATURE

```
def start_ap(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.wifi_manager.start_ap("wlan0")
```

stop_ap Stops the software access point.

SIGNATURE

```
def stop_ap(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.wifi_manager.stop_ap("wlan0")
```

is_wifi_connected Returns True if the interface is associated with an SSID.

SIGNATURE

```
def is_wifi_connected(self, interface_name: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
connected	bool	

EXAMPLE

```
up = c.wifi_manager.is_wifi_connected("wlan0")
```

get_wifi_link_properties Returns full link properties for an interface, or None.

SIGNATURE

```
def get_wifi_link_properties(self, interface_name: str) -> WiFiLinkProperties | None:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
properties	WiFiLinkProperties None	

EXAMPLE

```
props = c.wifi_manager.get_wifi_link_properties("wlan0")
```

set_wifi_mode Sets the operating mode of a Wi-Fi interface.

SIGNATURE

```
def set_wifi_mode(self, interface_name: str, mode: WiFiMode) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	
mode	WiFiMode	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
from client.wifi_manager import WIFI_MODE_CLIENT
ok = c.wifi_manager.set_wifi_mode("wlan0", WIFI_MODE_CLIENT)
```

get_wifi_mode Returns the current operating mode of a Wi-Fi interface.

SIGNATURE

```
def get_wifi_mode(self, interface_name: str) -> WiFiMode:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
mode	WiFiMode	

EXAMPLE

```
mode = c.wifi_manager.get_wifi_mode("wlan0")
```

get_ap_properties Returns the current access-point configuration, or None.

SIGNATURE

```
def get_ap_properties(self, interface_name: str) -> APProperties | None:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
properties	APProperties None	

EXAMPLE

```
props = c.wifi_manager.get_ap_properties("wlan0")
```

get_client_properties Returns the current client association properties, or None.

SIGNATURE

```
def get_client_properties(self, interface_name: str) -> ClientProperties | None:
```

PARAMETERS

NAME	TYPE	NOTES
interface_name	str	

RETURNS

NAME	TYPE	NOTES
properties	ClientProperties None	

EXAMPLE

```
props = c.wifi_manager.get_client_properties("wlan0")
```

GPIO Manager

GPIO digital I/O — list lines, read/drive levels, and set direction.

list_pins Returns all GPIO lines exported by the device.

SIGNATURE

```
def list_pins(self) -> list[GpioPin]:
```

RETURNS

NAME	TYPE	NOTES
pins	list[GpioPin]	

EXAMPLE

```
pins = c.gpio_manager.list_pins()
```

get_level Reads the logic level of a GPIO line.

SIGNATURE

```
def get_level(self, pin: GpioPin) -> GpioLevel:
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	

RETURNS

NAME	TYPE	NOTES
level	GpioLevel	

EXAMPLE

```
level = c.gpio_manager.get_level(GpioPin(number=26, chip_number=0))
```

set_level Drives a GPIO output HIGH or LOW.

SIGNATURE

```
def set_level(self, pin: GpioPin, level: GpioLevel) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	
level	GpioLevel	

RETURNS

NAME	TYPE	NOTES
	None	

EXAMPLE

```
from client.gpio_manager import GPIO_LEVEL_HIGH
c.gpio_manager.set_level(GpioPin(number=26, chip_number=0), GPIO_LEVEL_HIGH)
```

set_direction Sets the line direction.

SIGNATURE

```
def set_direction(self, pin: GpioPin, direction: GpioDirection) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	
direction	GpioDirection	

RETURNS

NAME	TYPE	NOTES
	None	

EXAMPLE

```
from client.gpio_manager import GPIO_DIR_OUT
pin = GpioPin(number=26, chip_number=0)
c.gpio_manager.set_direction(pin, GPIO_DIR_OUT)
```

get_direction Returns the direction of a GPIO line.

SIGNATURE

```
def get_direction(self, pin: GpioPin) -> GpioDirection:
```

PARAMETERS

NAME	TYPE	NOTES
pin	GpioPin	

RETURNS

NAME	TYPE	NOTES
direction	GpioDirection	

EXAMPLE

```
direction = c.gpio_manager.get_direction(GpioPin(number=26, chip_number=0))
```

PWM Manager

Hardware PWM channel control.

list_channels Returns all PWM channels available on the device.

SIGNATURE

```
def list_channels(self) -> list[PwmChannel]:
```

RETURNS

NAME	TYPE	NOTES
channels	list[PwmChannel]	

EXAMPLE

```
channels = c.pwm_manager.list_channels()
```

get_properties Reads the current configuration of a PWM channel.

SIGNATURE

```
def get_properties(self, ch: PwmChannel) -> PwmProperties:
```

PARAMETERS

NAME	TYPE	NOTES
ch	PwmChannel	

RETURNS

NAME	TYPE	NOTES
props	PwmProperties	

EXAMPLE

```
props = c.pwm_manager.get_properties(ch)
```

set_pwm Configures and starts a PWM channel.

SIGNATURE

```
def set_pwm(self, ch: PwmChannel, duty_cycle: float, frequency_hz: float) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
ch	PwmChannel	
duty_cycle	float	0.0–1.0
frequency_hz	float	

EXAMPLE

```
c_pwm_manager.set_pwm(ch, duty_cycle=0.5, frequency_hz=1000.0)
```

stop_pwm Disables PWM output on a channel.

SIGNATURE

```
def stop_pwm(self, ch: PwmChannel) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
ch	PwmChannel	

EXAMPLE

```
c_pwm_manager.stop_pwm(ch)
```

UART Manager

Serial port (UART) communication. Open returns a UartPort handle — all port operations are called on the handle.

list_ports Returns device paths for all available UART ports.

SIGNATURE

```
def list_ports(self) -> list[str]:
```

RETURNS

NAME	TYPE	NOTES
ports	list[str]	

EXAMPLE

```
ports = c.uart_manager.list_ports()
```

open Opens and configures a UART port. Returns a UartPort handle for all subsequent operations.

SIGNATURE

```
def open(self, cfg: UartConfig) -> UartPort:
```

PARAMETERS

NAME	TYPE	NOTES
cfg	UartConfig	

RETURNS

NAME	TYPE	NOTES
port	UartPort	

EXAMPLE

```
port = c.uart_manager.open(UartConfig(port="/dev/ttyS0", baudrate=115200, data_bits=8))
```

close Closes the port.

SIGNATURE

```
def close(self) -> None:
```

RETURNS

NAME	TYPE	NOTES
	None	

EXAMPLE

```
port.close()
```

get_config Returns the current configuration of the port.

SIGNATURE

```
def get_config(self) -> UartConfig:
```

RETURNS

NAME	TYPE	NOTES
config	UartConfig	

EXAMPLE

```
cfg = port.get_config()
```

write Writes bytes to the port. Returns the number of bytes written.

SIGNATURE

```
def write(self, data: bytes) -> int:
```

PARAMETERS

NAME	TYPE	NOTES
data	bytes	

RETURNS

NAME	TYPE	NOTES
bytes_written	int	

EXAMPLE

```
n = port.write(b"AT\r\n")
```

listen Streams received bytes from the port.

SIGNATURE

```
def listen(self, max_chunk_size: int = 256) -> Iterator[bytes]:
```

PARAMETERS

NAME	TYPE	NOTES
max_chunk_size	int	default 256

RETURNS

NAME	TYPE	NOTES
	Iterator[bytes]	

EXAMPLE

```
for chunk in port.listen():  
    print(chunk)
```

listen_async

Opens the stream immediately (raises on failure), then starts a background thread that pushes received bytes into a queue. Returns (queue, stop_event).

SIGNATURE

```
def listen_async(self, max_chunk_size: int = 256, max_queue: int = 256) -> tuple[Queue,  
Event]:
```

PARAMETERS

NAME	TYPE	NOTES
max_chunk_size	int	default 256
max_queue	int	buffer size, default 256

RETURNS

NAME	TYPE	NOTES
	tuple[queue.Queue, threading.Event]	

EXAMPLE

```
q, stop = port.listen_async()  
while (chunk := q.get()) is not None:  
    process(chunk)
```

I2C Manager

I2C bus discovery and data transfer. Open returns an I2Cbus handle — all bus operations are called on the handle.

list_buses Returns available I2C bus numbers.

SIGNATURE

```
def list_buses(self) -> list[int]:
```

RETURNS

NAME	TYPE	NOTES
buses	list[int]	

EXAMPLE

```
buses = c.i2c_manager.list_buses()
```

open Configures a bus and returns an I2Cbus handle for all subsequent operations.

SIGNATURE

```
def open(self, bus: int, clock_hz: int, ...) -> I2Cbus:
```

PARAMETERS

NAME	TYPE	NOTES
bus	int	
clock_hz	int	
ten_bit_addr	bool	default False
clock_stretching	bool	default True

RETURNS

NAME	TYPE	NOTES
bus	I2Cbus	

EXAMPLE

```
bus = c.i2c_manager.open(bus=1, clock_hz=400_000)
data = bus.read(addr=0x48, n=2)
```

scan Probes all 7-bit addresses on the bus. Returns responding addresses.

SIGNATURE

```
def scan(self) -> list[int]:
```

RETURNS

NAME	TYPE	NOTES
addresses	<code>list[int]</code>	

EXAMPLE

```
addrs = bus.scan()
```

get_config Returns the current clock and addressing configuration for the bus.

SIGNATURE

```
def get_config(self) -> I2CConfig:
```

RETURNS

NAME	TYPE	NOTES
config	<code>I2CConfig</code>	

EXAMPLE

```
cfg = bus.get_config()
```

transfer Performs a raw I2C transaction (write, read, or write-then-read).

SIGNATURE

```
def transfer(self, address: int, data: bytes = b"", read_length: int = 0, ...) -> bytes:
```

PARAMETERS

NAME	TYPE	NOTES
address	int	
data	bytes	default b""
read_length	int	default 0
flags	int	default 0

RETURNS

NAME	TYPE	NOTES
data	bytes	

EXAMPLE

```
rx = bus.transfer(address=0x48, data=b"\\x00", read_length=2)
```

write Writes bytes to an I2C device.

SIGNATURE

```
def write(self, addr: int, data: bytes) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
addr	int	
data	bytes	

RETURNS

NAME	TYPE	NOTES
	None	

EXAMPLE

```
bus.write(addr=0x3C, data=b"\\x00\\x01")
```

read Reads n bytes from an I2C device.

SIGNATURE

```
def read(self, addr: int, n: int) -> bytes:
```

PARAMETERS

NAME	TYPE	NOTES
addr	int	
n	int	

RETURNS

NAME	TYPE	NOTES
data	bytes	

EXAMPLE

```
data = bus.read(addr=0x48, n=2)
```

write_read Writes bytes then reads back a response (combined transaction).

SIGNATURE

```
def write_read(self, addr: int, write_data: bytes, read_len: int) -> bytes:
```

PARAMETERS

NAME	TYPE	NOTES
addr	int	
write_data	bytes	
read_len	int	

RETURNS

NAME	TYPE	NOTES
data	bytes	

EXAMPLE

```
result = bus.write_read(addr=0x48, write_data=b"\\x00", read_len=2)
```

SPI Manager

SPI bus configuration and full-duplex transfer. Open returns a SpiDevice handle — all device operations are called on the handle.

list_devices Returns device paths for all available SPI devices.

SIGNATURE

```
def list_devices(self) -> list[str]:
```

RETURNS

NAME	TYPE	NOTES
devices	list[str]	

EXAMPLE

```
devs = c.spi_manager.list_devices()
```

open Configures a SPI device and returns a SpiDevice handle for all subsequent operations.

SIGNATURE

```
def open(self, bus: int, chip_select: int, max_speed_hz: int, ...) -> SpiDevice:
```

PARAMETERS

NAME	TYPE	NOTES
bus	int	
chip_select	int	
max_speed_hz	int	
bits_per_word	int	
mode	int	default 0 (SPI_MODE_0)
lsb_first	bool	default False

RETURNS

NAME	TYPE	NOTES
device	SpiDevice	

EXAMPLE

```
dev = c.spi_manager.open(bus=0, chip_select=0, max_speed_hz=1_000_000, bits_per_word=8)
rx = dev.transfer(data_out=b"\xAB", read_length=2)
```

get_config Returns the current configuration for this device.

SIGNATURE

```
def get_config(self) -> SpiConfig:
```

RETURNS

NAME	TYPE	NOTES
config	SpiConfig	

EXAMPLE

```
cfg = dev.get_config()
```

transfer Performs a full-duplex SPI transfer.

SIGNATURE

```
def transfer(self, data_out: bytes, read_length: int) -> bytes:
```

PARAMETERS

NAME	TYPE	NOTES
data_out	bytes	bytes to clock out (MOSI)
read_length	int	bytes to clock in (MISO)

RETURNS

NAME	TYPE	NOTES
data_in	bytes	

EXAMPLE

```
rx = dev.transfer(data_out=b"\xAB\xCD", read_length=2)
```

write Clocks bytes out on MOSI without reading a response.

SIGNATURE

```
def write(self, data: bytes) -> None:
```

PARAMETERS

NAME	TYPE	NOTES
data	bytes	

RETURNS

NAME	TYPE	NOTES
	None	

EXAMPLE

```
dev.write(b"\x01\x02")
```

read Clocks in n bytes from MISO (clocks out zeros on MOSI).

SIGNATURE

```
def read(self, n: int) -> bytes:
```

PARAMETERS

NAME	TYPE	NOTES
n	int	

RETURNS

NAME	TYPE	NOTES
data	bytes	

EXAMPLE

```
data = dev.read(n=4)
```

write_read Full-duplex transfer where read length matches write length.

SIGNATURE

```
def write_read(self, data_out: bytes) -> bytes:
```

PARAMETERS

NAME	TYPE	NOTES
data_out	bytes	

RETURNS

NAME	TYPE	NOTES
data_in	bytes	

EXAMPLE

```
rx = dev.write_read(b"\\xAB\\xCD")
```

Camera Manager

Capture images and stream video frames from V4L2 camera devices.

capture_image Captures a single frame.

SIGNATURE

```
def capture_image(self, device_id: str, width: int = 0, height: int = 0, format: str = "mjpeg") -> CapturedImage:
```

PARAMETERS

NAME	TYPE	NOTES
device_id	str	V4L2 device path e.g. "/dev/video0"
width	int	0 = device default
height	int	0 = device default
format	str	"mjpeg" or "yuv420"

RETURNS

NAME	TYPE	NOTES
image	CapturedImage	CapturedImage(success, image_data, format, timestamp)

EXAMPLE

```
img = c.camera_manager.capture_image("/dev/video0", width=1280, height=720)
open("frame.jpg", "wb").write(img.image_data)
```

stream_frames Streams Frame proto messages from a camera device.

SIGNATURE

```
def stream_frames(self, device_id: str, fps: int = 30, ...) -> Generator:
```

PARAMETERS

NAME	TYPE	NOTES
device_id	str	
fps	int	default 30
width	int	0 = device default
height	int	0 = device default
timeout	float None	

RETURNS

NAME	TYPE	NOTES
frames	Generator[Frame]	

EXAMPLE

```
for frame in c.camera_manager.stream_frames("/dev/video0", fps=30):  
    process(frame.data, frame.width, frame.height)  
    if done: break
```

lock_camera Acquires exclusive access to a camera device.

SIGNATURE

```
def lock_camera(self, device_id: str, client_id: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
device_id	str	
client_id	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.camera_manager.lock_camera("/dev/video0", "myapp")
```

get_device_info Returns driver and format capabilities for a camera device.

SIGNATURE

```
def get_device_info(self, device_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
device_id	str	

RETURNS

NAME	TYPE	NOTES
response		DeviceInfoResponse proto

EXAMPLE

```
info = c.camera_manager.get_device_info("/dev/video0")
```

list_devices Returns all camera devices discovered on the system.

SIGNATURE

```
def list_devices(self) -> list[CameraDeviceInfo]:
```

RETURNS

NAME	TYPE	NOTES
devices	list[CameraDeviceInfo]	

EXAMPLE

```
devs = c.camera_manager.list_devices()
```

unlock_camera Releases exclusive access to a camera device.

SIGNATURE

```
def unlock_camera(self, device_id: str, client_id: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
device_id	str	
client_id	str	

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.camera_manager.unlock_camera("/dev/video0", "myapp")
```

AI Manager

Load ONNX/TFLite models and run inference on the device NPU or CPU.

load_model Loads a model already present on the device filesystem.

SIGNATURE

```
def load_model(self, model_id: str, model_path: str, backend: int = ONNX, ...):
```

PARAMETERS

NAME	TYPE	NOTES
model_id	str	
model_path	str	Absolute path on device
backend	int	ONNX or TFLITE
execution	int	EXEC_CPU, EXEC_GPU, or EXEC_HIGH_THREADS
version	str	optional label

RETURNS

NAME	TYPE	NOTES
response		LoadModelResponse proto

EXAMPLE

```
c.ai_manager.load_model("face", "/data/face.onnx", backend=ONNX, execution=EXEC_CPU)
```

upload_and_load_model Streams a local model file to the device and loads it.

SIGNATURE

```
def upload_and_load_model(self, model_id: str, local_path: str, ...) -> LoadModelResponse:
```

PARAMETERS

NAME	TYPE	NOTES
model_id	str	
local_path	str	Path on the local machine
backend	int	ONNX or TFLITE
execution	int	
chunk_size	int	default 256 KiB

RETURNS

NAME	TYPE	NOTES
response		LoadModelResponse proto

EXAMPLE

```
c.ai_manager.upload_and_load_model("face", "/local/face.onnx", backend=ONNX)
```

infer Runs a single inference pass.

SIGNATURE

```
def infer(self, model_id: str, input_data: bytes, input_shape: list[int] | None = None, ...):
```

PARAMETERS

NAME	TYPE	NOTES
model_id	str	
input_data	bytes	Raw tensor bytes
input_shape	list[int] None	optional — uses model default
input_dtype	int	TENSOR_FLOAT32 or TENSOR_UINT8

RETURNS

NAME	TYPE	NOTES
response		InferResponse proto

EXAMPLE

```
resp = c.ai_manager.infer("face", input_data=tensor_bytes, input_shape=[1, 3, 224, 224])
```

list_models Returns all currently loaded models.

SIGNATURE

```
def list_models(self) -> list[ModelInfo]:
```

RETURNS

NAME	TYPE	NOTES
models	list[ModelInfo]	

EXAMPLE

```
models = c.ai_manager.list_models()
```

is_model_loaded Returns True if a model is currently loaded.

SIGNATURE

```
def is_model_loaded(self, model_id: str) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
model_id	str	

RETURNS

NAME	TYPE	NOTES
loaded	bool	

EXAMPLE

```
if c.ai_manager.is_model_loaded("face"):  
    print("ready")
```

unload_model Unloads a model and frees its resources.

SIGNATURE

```
def unload_model(self, model_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
model_id	str	

EXAMPLE

```
c.ai_manager.unload_model("face")
```

stream_infer Bidirectional streaming inference. Yields InferResponse messages.

SIGNATURE

```
def stream_infer(self, requests: Iterator, timeout: float | None = None) -> Generator:
```

PARAMETERS

NAME	TYPE	NOTES
requests		Iterator of InferRequest protos
timeout	float None	

RETURNS

NAME	TYPE	NOTES
responses	Generator[InferResponse]	

EXAMPLE

```
for resp in c.ai_manager.stream_infer(request_iter):
    process(resp.output_data)
```

send

Go-only: submits an inference request on a stream handle. Python equivalent: use StreamInfer(requests) with a generator or queue.

SIGNATURE

```
# Python equivalent: StreamInfer(requests) – a single bidirectional generator
```

EXAMPLE

```
import queue
q = queue.queue()
def reqs():
    while True: yield q.get()
for resp in c.ai_manager.stream_infer(reqs()):
    print(resp)
```

recv

Go-only: reads the next inference result from a stream handle. Python equivalent: `StreamInfer()` yields results directly.

SIGNATURE

```
# Python equivalent: each iteration of StreamInfer() yields the next result
```

EXAMPLE

```
# See StreamInfer for the full example
```

close

Go-only: signals end of the send side of a stream handle. Python equivalent: `StreamInfer()` closes automatically when the request iterator is exhausted.

SIGNATURE

```
# Python equivalent: StreamInfer() closes automatically when request iterator is exhausted
```

EXAMPLE

```
# See StreamInfer for the full example
```

Firewall Manager

Firewall zone and rule management (iptables-backed).

list_zones Returns all configured firewall zones.

SIGNATURE

```
def list_zones(self):
```

RETURNS

NAME	TYPE	NOTES
response		ListZonesResponse proto

EXAMPLE

```
zones = c.firewall_manager.list_zones()
```

add_zone Creates a new firewall zone.

SIGNATURE

```
def add_zone(self, name: str, interfaces: list[str], ...) :
```

PARAMETERS

NAME	TYPE	NOTES
name	str	
interfaces	list[str]	
input_policy	ZonePolicy	default POLICY_ACCEPT
output_policy	ZonePolicy	default POLICY_ACCEPT
masquerade	bool	

RETURNS

NAME	TYPE	NOTES
response		AddZoneResponse proto

EXAMPLE

```
c.firewall_manager.add_zone("lan", interfaces=["eth0"], masquerade=True)
```

remove_zone Deletes a firewall zone by name.

SIGNATURE

```
def remove_zone(self, name: str):
```

PARAMETERS

NAME	TYPE	NOTES
name	str	

RETURNS

NAME	TYPE	NOTES
response		RemoveZoneResponse proto

EXAMPLE

```
c.firewall_manager.remove_zone("lan")
```

list_rules Returns all configured forwarding rules.

SIGNATURE

```
def list_rules(self):
```

RETURNS

NAME	TYPE	NOTES
response		ListRulesResponse proto

EXAMPLE

```
rules = c.firewall_manager.list_rules()
```

flush_rules Removes all forwarding rules.

SIGNATURE

```
def flush_rules(self):
```

RETURNS

NAME	TYPE	NOTES
response		FlushRulesResponse proto

EXAMPLE

```
c.firewall_manager.flush_rules()
```

add_rule Adds a forwarding rule between two zones.

SIGNATURE

```
def add_rule(self, src_zone: str, dst_zone: str, protocol: FirewallProtocol, src_ip: str, dest_port: int, action: ZonePolicy, comment: str = "") -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
src_zone	str	Source zone name
dst_zone	str	Destination zone name
protocol	FirewallProtocol	
src_ip	str	Source IP filter (empty = any)
dest_port	int	Destination port (0 = any)
action	ZonePolicy	ACCEPT/DROP/REJECT
comment	str	Optional human-readable note

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
from client.firewall_manager import PROTO_TCP, POLICY_ACCEPT
ok = c.firewall_manager.add_rule("lan", "wan", PROTO_TCP, src_ip="", dest_port=80,
action=POLICY_ACCEPT)
```

remove_rule Removes a forwarding rule by its ID.

SIGNATURE

```
def remove_rule(self, rule_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
rule_id	str	Rule ID returned by AddRule

EXAMPLE

```
c.firewall_manager.remove_rule("rule-1")
```

apply_firewall Commits all pending zone and rule changes to the kernel firewall.

SIGNATURE

```
def apply_firewall(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.firewall_manager.apply_firewall()
```

VPN Manager

WireGuard and OpenVPN profile management and tunnel control.

apply_profile Creates or updates a VPN profile from raw config data.

SIGNATURE

```
def apply_profile(self, profile_id: str, display_name: str, provider: int, config_data: bytes, ...) :
```

PARAMETERS

NAME	TYPE	NOTES
profile_id	str	
display_name	str	
provider	int	VPN_PROVIDER_WIREGUARD or VPN_PROVIDER_OPENVPN
config_data	bytes	raw .conf or .ovpn bytes
auto_connect	bool	reconnect after reboots
connect_after_apply	bool	start tunnel immediately

RETURNS

NAME	TYPE	NOTES
response		ApplyProfileResponse proto

EXAMPLE

```
from client.vpn_manager import VPN_PROVIDER_WIREGUARD
c.vpn_manager.apply_profile(
    "wg0", "Office VPN", VPN_PROVIDER_WIREGUARD,
    open("wg0.conf", "rb").read(), connect_after_apply=True)
```

connect Brings up the tunnel for a profile.

SIGNATURE

```
def connect(self, profile_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
profile_id	str	

RETURNS

NAME	TYPE	NOTES
response		ConnectResponse proto

EXAMPLE

```
c.vpn_manager.connect("wg0")
```

disconnect Tears down the tunnel for a profile.

SIGNATURE

```
def disconnect(self, profile_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
profile_id	str	

EXAMPLE

```
c.vpn_manager.disconnect("wg0")
```

list_profiles Returns all configured VPN profiles.

SIGNATURE

```
def list_profiles(self):
```

RETURNS

NAME	TYPE	NOTES
response		ListProfilesResponse proto

EXAMPLE

```
profiles = c.vpn_manager.list_profiles()
```

get_status Returns the current tunnel session status.

SIGNATURE

```
def get_status(self):
```

RETURNS

NAME	TYPE	NOTES
response		VPNStatusResponse proto

EXAMPLE

```
status = c.vpn_manager.get_status()
```

watch_events Streams VPNEvent messages for a profile's tunnel state changes.

SIGNATURE

```
def watch_events(self, profile_id: str = "", timeout: float | None = None) -> Generator:
```

PARAMETERS

NAME	TYPE	NOTES
profile_id	str	empty = all profiles
timeout	float None	

RETURNS

NAME	TYPE	NOTES
events	Generator[VPNEvent]	

EXAMPLE

```
from client.vpn_manager import TUNNEL_STATE_UP
for ev in c.vpn_manager.watch_events(profile_id="wg0"):
    if ev.state == TUNNEL_STATE_UP:
        print("VPN connected")
```

get_capabilities Returns the VPN providers supported by this device.

SIGNATURE

```
def get_capabilities(self):
```

RETURNS

NAME	TYPE	NOTES
response		GetCapabilitiesResponse proto

EXAMPLE

```
caps = c.vpn_manager.get_capabilities()
```

remove_profile Deletes a VPN profile.

SIGNATURE

```
def remove_profile(self, profile_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
profile_id	str	

EXAMPLE

```
c.vpn_manager.remove_profile("wg0")
```

list_sessions Returns all active VPN sessions.

SIGNATURE

```
def list_sessions(self):
```

RETURNS

NAME	TYPE	NOTES
response		ListSessionsResponse proto

EXAMPLE

```
sessions = c.vpn_manager.list_sessions()
```

apply_wireguard Shorthand to create or update a WireGuard profile from a .conf file.

SIGNATURE

```
def apply_wireguard(self, display_name: str, config_data: bytes, auto_connect: bool = False):
```

PARAMETERS

NAME	TYPE	NOTES
display_name	str	
config_data	bytes	WireGuard .conf file contents
auto_connect	bool	default False

RETURNS

NAME	TYPE	NOTES
response		ApplyProfileResponse proto

EXAMPLE

```
c.vpn_manager.apply_wireguard("Office VPN", open("wg0.conf", "rb").read())
```

apply_openvpn Shorthand to create or update an OpenVPN profile from an .ovpn file.

SIGNATURE

```
def apply_openvpn(self, display_name: str, config_data: bytes, auto_connect: bool = False):
```

PARAMETERS

NAME	TYPE	NOTES
display_name	str	
config_data	bytes	OpenVPN .ovpn file contents
auto_connect	bool	default False

RETURNS

NAME	TYPE	NOTES
response		ApplyProfileResponse proto

EXAMPLE

```
c.vpn_manager.apply_openvpn("Corp VPN", open("corp.ovpn", "rb").read())
```

is_connected Returns True if a VPN tunnel is currently active.

SIGNATURE

```
def is_connected(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
connected	bool	

EXAMPLE

```
if c.vpn_manager.is_connected():  
    print("VPN is up")
```

App Hub Manager

Register an app's HTTP service on the device web portal.

register_service Registers this app's HTTP service with the portal.

SIGNATURE

```
def register_service(self, host: str, port: int, routes: list[str] | None = None, ...):
```

PARAMETERS

NAME	TYPE	NOTES
host	str	Listen address (usually "127.0.0.1")
port	int	
routes	list[str] None	URL path prefixes
health_endpoint	str	default "/health"
exposure_mode	int	

RETURNS

NAME	TYPE	NOTES
response		RegisterServiceResponse proto

EXAMPLE

```
c.app_hub_manager.register_service("127.0.0.1", port=8080, routes=["/api", "/ui"])
```

unregister_service Removes this app's service registration.

SIGNATURE

```
def unregister_service(self):
```

EXAMPLE

```
c.app_hub_manager.unregister_service()
```

list_services Returns all registered services on the portal.

SIGNATURE

```
def list_services(self):
```

RETURNS

NAME	TYPE	NOTES
services		ListServicesResponse proto

EXAMPLE

```
services = c.app_hub_manager.list_services()
```

watch_services Streams ServiceEvent messages as services register or deregister.

SIGNATURE

```
def watch_services(self, timeout: float | None = None) -> Generator:
```

PARAMETERS

NAME	TYPE	NOTES
timeout	float None	

RETURNS

NAME	TYPE	NOTES
events	Generator[ServiceEvent]	

EXAMPLE

```
for ev in c.app_hub_manager.watch_services():
    print(ev.service_id, ev.type)
```

get_service Returns the registration details for a service by ID.

SIGNATURE

```
def get_service(self, service_id: str):
```

PARAMETERS

NAME	TYPE	NOTES
service_id	str	

RETURNS

NAME	TYPE	NOTES
response		ServiceRegistration proto

EXAMPLE

```
svc = c.app_hub_manager.get_service("myapp")
```

add_route Adds a URL path prefix to this service's routing table.

SIGNATURE

```
def add_route(self, path: str):
```

PARAMETERS

NAME	TYPE	NOTES
path	str	URL path prefix (e.g. "/api")

EXAMPLE

```
c.app_hub_manager.add_route("/api/v2")
```

remove_route Removes a URL path prefix from this service's routing table.

SIGNATURE

```
def remove_route(self, path: str):
```

PARAMETERS

NAME	TYPE	NOTES
path	str	

EXAMPLE

```
c.app_hub_manager.remove_route("/api/v2")
```

get_routing_table Returns the full portal routing table.

SIGNATURE

```
def get_routing_table(self):
```

RETURNS

NAME	TYPE	NOTES
response		RoutingTableResponse proto

EXAMPLE

```
table = c.app_hub_manager.get_routing_table()
```

register_web_ui Shorthand to register a web UI from an "host:port" address string.

SIGNATURE

```
def register_web_ui(self, addr: str, route: str):
```

PARAMETERS

NAME	TYPE	NOTES
addr	str	"host:port" of the service
route	str	URL path prefix to proxy (e.g. "/ui")

RETURNS

NAME	TYPE	NOTES
response		RegisterServiceResponse proto

EXAMPLE

```
c.app_hub_manager.register_web_ui("127.0.0.1:8080", "/ui")
```

Event Manager

Subscribe to system-wide events emitted by the Gravity runtime.

subscribe Opens a server-side event stream and yields Event proto messages.

SIGNATURE

```
def subscribe(self, *event_types: EventType.ValueType, timeout: float | None = None) -> Generator:
```

PARAMETERS

NAME	TYPE	NOTES
event_types	EventType.ValueType	EVENT_ constants — omit to receive all types
timeout	float None	stream idle timeout

RETURNS

NAME	TYPE	NOTES
events	Generator[Event]	

EXAMPLE

```
# All events
for ev in c.event_manager.subscribe():
    print(ev.type, ev.payload)
    if done: break

# Filtered
from client.event_manager import EVENT_APP_CRASHED, EVENT_NET_DOWN
for ev in c.event_manager.subscribe(EVENT_APP_CRASHED, EVENT_NET_DOWN):
    alert(ev)
```

Development Manager

Remote log streaming from the device log daemon.

subscribe_logs Opens a server-side log stream yielding LogMessage protos.

SIGNATURE

```
def subscribe_logs(self, app: str = "", tag: str = "", level: LogLevel = LogLevel.DEBUG, ...):
```

PARAMETERS

NAME	TYPE	NOTES
app	str	Filter to app name (empty = all)
tag	str	Filter to tag (empty = all)
level	LogLevel	minimum log level (default LogLevel.DEBUG)
timeout	float	default 60.0 s

RETURNS

NAME	TYPE	NOTES
stream		gRPC server-side stream of LogMessage

EXAMPLE

```
from client.development_manager import LogLevel
for entry in c.development_manager.subscribe_logs(app="myapp", level=LogLevel.INFO):
    print(entry.timestamp, entry.message)
```

subscribe_logs_async

Non-blocking variant of SubscribeLogs — returns a Queue that receives LogMessage entries.

SIGNATURE

```
def subscribe_logs_async(self, app: str = "", tag: str = "", level: LogLevel =
LogLevel.DEBUG, max_queue: int = 1000) -> queue.Queue:
```

PARAMETERS

NAME	TYPE	NOTES
app	str	Filter to app name (empty = all)
tag	str	Filter to tag (empty = all)
level	LogLevel	minimum log level (default LogLevel.DEBUG)
max_queue	int	default 1000

RETURNS

NAME	TYPE	NOTES
queue	queue.Queue	yields LogMessage items

EXAMPLE

```
from client.development_manager import LogLevel
q = c.development_manager.subscribe_logs_async(app="myapp", level=LogLevel.INFO)
while True:
    entry = q.get()
    print(entry.message)
```

Update Manager

Apply OTA firmware updates and perform factory resets.

update

Uploads a firmware archive and applies it.

SIGNATURE

```
def update(self, local_path: str, chunk_size: int = 262144, timeout: float = 120.0) -> bool:
```

PARAMETERS

NAME	TYPE	NOTES
local_path	str	Local path to the .ota archive
chunk_size	int	default 256 KiB
timeout	float	default 120.0 s

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.update_manager.update("/tmp/firmware.ota")
```

factory_reset

Restores the device to factory defaults. Erases all user data.

SIGNATURE

```
def factory_reset(self) -> bool:
```

RETURNS

NAME	TYPE	NOTES
success	bool	

EXAMPLE

```
ok = c.update_manager.factory_reset()
```

Power Manager

Device reboot and power-off.

reboot Reboots the device.

SIGNATURE

```
def reboot(self, force: bool = False, reason: str = "") -> PowerResult:
```

PARAMETERS

NAME	TYPE	NOTES
force	bool	Skip graceful app shutdown
reason	str	Logged before reboot

RETURNS

NAME	TYPE	NOTES
result	PowerResult	

EXAMPLE

```
result = c.power_manager.reboot(reason="update")
```

shutdown Powers off the device.

SIGNATURE

```
def shutdown(self, force: bool = False, reason: str = "") -> PowerResult:
```

PARAMETERS

NAME	TYPE	NOTES
force	bool	Skip graceful app shutdown
reason	str	

RETURNS

NAME	TYPE	NOTES
result	PowerResult	

EXAMPLE

```
result = c.power_manager.shutdown()
```

Types

Complex objects returned or accepted by the SDK methods.

gpio_pin

Identifies a GPIO line by chip index and line offset.

FIELDS		
NAME	TYPE	NOTES
gpiochip	int	Chip index (0 = /dev/gpiochip0)
line	int	Line offset within the chip
name	str	Human-readable pin name (e.g. "GPIO17", "PWM0_OUT")

gpio_direction

GPIO pin direction enum.

FIELDS		
NAME	TYPE	NOTES
GPIO_DIR_OUT	0	Output
GPIO_DIR_IN	1	Input

gpio_level

GPIO pin logic level enum.

FIELDS		
NAME	TYPE	NOTES
GPIO_LEVEL_LOW	0	
GPIO_LEVEL_HIGH	1	

uart_config

Configuration used to open a UART port.

FIELDS

NAME	TYPE	NOTES
port	str	Port name (e.g. "ttyS0", "ttyAMA0")
baudrate	int	Baud rate (e.g. 9600, 115200)
data_bits	int	5, 6, 7 or 8
parity	UartParity	Parity mode
stop_bits	UartStopBits	Stop bits
flow_control	UartFlowControl	Flow control mode

uart_parity

UART parity mode enum.

FIELDS

NAME	TYPE	NOTES
UART_PARITY_NONE	0	No parity
UART_PARITY_EVEN	1	Even parity
UART_PARITY_ODD	2	Odd parity

uart_stop_bits

UART stop bits enum.

FIELDS

NAME	TYPE	NOTES
UART_STOPBITS_1	0	1 stop bit
UART_STOPBITS_2	1	2 stop bits

uart_flow_control

UART flow control enum.

FIELDS

NAME	TYPE	NOTES
UART_FLOW_NONE	0	No flow control
UART_FLOW_HARDWARE	1	Hardware (RTS/CTS)
UART_FLOW_SOFTWARE	2	Software (XON/XOFF)

i2c_config

I2C bus configuration.

FIELDS

NAME	TYPE	NOTES
bus	int	Bus number (e.g. 1 for /dev/i2c-1)
clock_hz	int	Clock frequency in Hz
ten_bit_addr	bool	True = 10-bit addressing, False = 7-bit (default)
clock_stretching	bool	Allow slave to hold SCL low to pause transfer

i2c_transfer_request

Generic I2C transfer. Operation inferred from fields: data only = write, read_length only = read, both = write-then-read.

FIELDS

NAME	TYPE	NOTES
bus	int	Bus number
address	int	7-bit device address (e.g. 0x48)
data	bytes	Bytes to write
read_length	int	Number of bytes to read
flags	int	Optional Linux i2c_msg flags

spi_config

SPI device configuration.

FIELDS

NAME	TYPE	NOTES
bus	int	Bus index (e.g. 0 for /dev/spidev0.*)
chip_select	int	Chip select (e.g. 0 for /dev/spidev0.0)
max_speed_hz	int	Max clock speed in Hz (e.g. 1000000)
bits_per_word	int	Bits per word (typically 8)
lsb_first	bool	True = LSB first, False = MSB first (default)
mode	SpiMode	SPI clock polarity and phase

spi_bus_request

Identifies a SPI device by bus + chip select.

FIELDS

NAME	TYPE	NOTES
bus	int	Bus index
chip_select	int	Chip select

spi_mode

SPI mode (clock polarity + phase) enum.

FIELDS

NAME	TYPE	NOTES
SPI_MODE_0	0	CPOL=0, CPHA=0 — most common
SPI_MODE_1	1	CPOL=0, CPHA=1
SPI_MODE_2	2	CPOL=1, CPHA=0
SPI_MODE_3	3	CPOL=1, CPHA=1

ethernet_link_properties

Ethernet interface state and configuration.

FIELDS

NAME	TYPE	NOTES
interface_name	str	Interface name (e.g. "eth0")
mac_address	str	MAC address
state	EthernetState	Connection state
mtu	int	MTU in bytes
dhcp_enable	bool	True if DHCP is enabled
ipv4_address	str	IP address in CIDR format
ipv4_gateway	str	Gateway IP
ipv4_dns	list[str]	DNS servers

ethernet_state

Ethernet connection state enum.

FIELDS

NAME	TYPE	NOTES
ETH_UNKNOWN	0	Unknown
ETH_UNMANAGED	10	Not managed
ETH_UNAVAILABLE	20	No carrier
ETH_DISCONNECTED	30	Disconnected
ETH_PREPARE	40	Preparing
ETH_CONFIG	50	Configuring
ETH_NEED_AUTH	60	Auth required
ETH_IP_CONFIG	70	Configuring IP
ETH_IP_CHECK	80	Verifying IP
ETH_SECONDARIES	90	Secondary connections
ETH_CONNECTED	100	Connected
ETH_DEACTIVATING	110	Deactivating
ETH_FAILED	120	Failed

wifi_link_properties

WiFi interface state and configuration.

FIELDS

NAME	TYPE	NOTES
interface_name	str	Interface name (e.g. "wlan0")
mac_address	str	MAC address
mode	WiFiMode	Operating mode
state	WiFiState	Connection state
mtu	int	MTU in bytes
ap_properties	APProperties	Set when mode is AP or AP_CLIENT
client_properties	ClientProperties	Set when mode is CLIENT or AP_CLIENT

ap_properties

Access Point configuration and status.

FIELDS

NAME	TYPE	NOTES
interface_name	str	Interface name
ssid	str	Network name
security	str	Security type: "none", "wpa2", "wpa3", "wpa2-wpa3"
ipv4_address	str	IP in CIDR format
channel	int	1-14 for 2.4 GHz, 36-165 for 5 GHz
band	str	"2.4GHz" or "5GHz"
hidden	bool	Hidden SSID
active	bool	AP is currently broadcasting
connected_clients	int	Number of connected clients

client_properties

WiFi client (station) connection state.

FIELDS

NAME	TYPE	NOTES
interface_name	str	Interface name
mac_address	str	MAC address
state	WiFiState	Connection state
mtu	int	MTU in bytes
ssid	str	Connected network name
bssid	str	MAC address of the AP
security	str	Security type
signal_strength	int	RSSI in dBm
channel	int	Channel number
band	str	"2.4GHz" or "5GHz"
dhcp_enable	bool	True if DHCP is enabled
ipv4_address	str	IP in CIDR format
ipv4_gateway	str	Gateway IP
ipv4_dns	list[str]	DNS servers

wifi_mode

WiFi operating mode enum.

FIELDS

NAME	TYPE	NOTES
WIFI_MODE_UNKNOWN	0	Unknown
WIFI_MODE_AP	1	Access Point only
WIFI_MODE_CLIENT	2	Client (station) only
WIFI_MODE_AP_CLIENT	3	AP + Client simultaneously
WIFI_MODE_DISABLED	4	Disabled

wifi_state

WiFi connection state enum.

FIELDS

NAME	TYPE	NOTES
WIFI_UNKNOWN	0	Unknown
WIFI_UNMANAGED	10	Not managed by NetworkManager
WIFI_UNAVAILABLE	20	Managed but no carrier
WIFI_DISCONNECTED	30	Disconnected
WIFI_PREPARE	40	Preparing connection
WIFI_CONFIG	50	Configuring
WIFI_NEED_AUTH	60	Authentication required
WIFI_IP_CONFIG	70	Configuring IP
WIFI_IP_CHECK	80	Verifying IP
WIFI_SECONDARIES	90	Secondary connections
WIFI_ACTIVATED	100	Connected
WIFI_DEACTIVATING	110	Deactivating
WIFI_FAILED	120	Failed

pwm_channel

Identifies a PWM hardware channel.

FIELDS

NAME	TYPE	NOTES
channel	int	Hardware channel index
name	str	Human-readable name (e.g. "pwm0")

pwm_properties

Current PWM channel state.

FIELDS		
NAME	TYPE	NOTES
channel	PwmChannel	Channel identification
enabled	bool	True if the channel is active
duty_cycle	float	Duty cycle 0.0-1.0
frequency_hz	float	Frequency in Hz (e.g. 1000)

vpn_profile

Persisted VPN configuration.

FIELDS		
NAME	TYPE	NOTES
profile_id	str	Unique profile identifier
display_name	str	Human-readable name
provider	VpnProvider	VPN backend
config_data	bytes	Raw provider config file (.conf, .ovpn, etc.)
auto_connect	bool	Connect automatically on boot
secret_ref	str	Optional reference to secrets store

vpn_session

Active VPN session (runtime state).

FIELDS		
NAME	TYPE	NOTES
session_id	str	Session identifier
profile_id	str	Associated profile
provider	VpnProvider	VPN backend
state	TunnelState	Current tunnel state
interface_name	str	Tunnel interface (e.g. "wg0")
rx_bytes	int	Bytes received
tx_bytes	int	Bytes sent
started_at_unix_ms	int	Start time (Unix ms UTC)

vpn_provider

VPN backend enum.

FIELDS

NAME	TYPE	NOTES
VPN_PROVIDER_UNSPECIFIED	0	
VPN_PROVIDER_WIREGUARD	1	WireGuard
VPN_PROVIDER_OPENVPN	2	OpenVPN
VPN_PROVIDER_IPSEC	3	IPSec
VPN_PROVIDER_CUSTOM	15	Custom provider

tunnel_state

VPN tunnel state enum.

FIELDS

NAME	TYPE	NOTES
TUNNEL_STATE_UNSPECIFIED	0	
TUNNEL_STATE_DOWN	1	Tunnel is down
TUNNEL_STATE_CONNECTING	2	Establishing connection
TUNNEL_STATE_UP	3	Tunnel is up
TUNNEL_STATE_DEGRADED	4	Partially operational
TUNNEL_STATE_ERROR	5	Error

firewall_rule

A firewall rule (includes server-generated id).

FIELDS

NAME	TYPE	NOTES
id	str	Server-generated rule ID
src_zone	str	Source zone (empty = any)
dst_zone	str	Destination zone (empty = any)
protocol	FirewallProtocol	Protocol filter
src_ip	str	CIDR source IP (empty = any)
dest_port	int	Destination port (0 = any)
action	ZonePolicy	ACCEPT, DROP or REJECT
comment	str	Optional description

zone_request

Firewall zone definition.

FIELDS

NAME	TYPE	NOTES
name	str	Zone name (e.g. "lan", "wan")
interfaces	list[str]	Bound interfaces (e.g. ["eth0"])
input_policy	ZonePolicy	Default inbound policy
output_policy	ZonePolicy	Default outbound policy
masquerade	bool	Enable NAT (typically only on WAN)

firewall_protocol

Firewall protocol filter enum.

FIELDS

NAME	TYPE	NOTES
PROTO_ANY	0	Any protocol
PROTO_TCP	1	TCP
PROTO_UDP	2	UDP
PROTO_ICMP	3	ICMP

zone_policy

Firewall zone policy enum.

FIELDS

NAME	TYPE	NOTES
POLICY_ACCEPT	0	Allow traffic
POLICY_DROP	1	Silently discard
POLICY_REJECT	2	Reject with ICMP error

tensor_info

Describes a single input or output tensor of a loaded model.

FIELDS

NAME	TYPE	NOTES
name	str	Tensor name
dtype	TensorDataType	Element data type
shape	list[int]	Tensor shape (e.g. [1, 640, 640, 3])

infer_response

Result of a synchronous inference call.

FIELDS		
NAME	TYPE	NOTES
success	bool	True if inference succeeded
output_data	bytes	First output tensor as raw bytes
output_shape	list[int]	Shape of output_data
named_outputs	dict[str, bytes]	All outputs keyed by tensor name
latency_us	int	Inference wall time in microseconds

model_info

Metadata for a loaded model.

FIELDS		
NAME	TYPE	NOTES
model_id	str	Model identifier
version	str	Version label
backend	ModelBackend	Inference backend
loaded_at_unix	int	Unix timestamp when loaded

model_backend

AI model inference backend enum.

FIELDS		
NAME	TYPE	NOTES
MODEL_BACKEND_UNSPECIFIED	0	
ONNX	1	ONNX Runtime
TFLITE	2	TensorFlow Lite

execution_mode

Compute path used during inference.

FIELDS		
NAME	TYPE	NOTES
EXEC_CPU	0	Single/multi-core CPU (default)
EXEC_GPU	1	GPU / NPU delegate
EXEC_HIGH_THREADS	2	CPU with maximum thread count

tensor_data_type

Raw byte encoding of tensor elements.

FIELDS		
NAME	TYPE	NOTES
TENSOR_FLOAT32	0	4-byte little-endian IEEE 754, normalised [0, 1]
TENSOR_UINT8	1	1-byte unsigned integer [0, 255]
TENSOR_INT32	2	4-byte little-endian signed integer
TENSOR_INT64	3	8-byte little-endian signed integer

camera_device_info

V4L2 camera device capabilities.

FIELDS		
NAME	TYPE	NOTES
device_id	str	Device path (e.g. "/dev/video0")
driver	str	Kernel driver name
card	str	Card/device name
supported_formats	list[str]	e.g. ["mjpeg", "yuyv"]
resolutions	list[str]	e.g. ["1920x1080", "640x480"]

frame

A single video frame from a camera stream.

FIELDS		
NAME	TYPE	NOTES
data	bytes	Raw frame data
timestamp	int	Capture timestamp
width	int	Frame width in pixels
height	int	Frame height in pixels
format	str	Pixel format (e.g. "mjpeg", "yuyv")

metrics_info

System-wide runtime metrics snapshot.

FIELDS		
NAME	TYPE	NOTES
sys_uptime	<code>MetricsInfo.SysUptime</code>	Uptime and load averages
cpu_stats	<code>MetricsInfo.CpuStats</code>	CPU software counters
cpu_freq	<code>MetricsInfo.CpuFreq</code>	CPU frequency
virtual_memory	<code>MetricsInfo.VirtualMemory</code>	RAM usage
disk_usage	<code>MetricsInfo.DiskUsage</code>	Disk space
disk_io_counters	<code>MetricsInfo.DiskIOCounters</code>	Disk I/O stats
network	<code>MetricsInfo.Network</code>	Network I/O stats

MetricsInfo.SysUptime

System uptime and load averages.

FIELDS		
NAME	TYPE	NOTES
time	<code>str</code>	Human-readable time string
uptime	<code>int</code>	Uptime in seconds
users	<code>int</code>	Number of logged-in users
load_average	<code>list[float]</code>	1 min, 5 min, 15 min load averages

MetricsInfo.CpuStats

CPU software counters.

FIELDS		
NAME	TYPE	NOTES
ctx_switches	<code>int</code>	Context switches
interrupts	<code>int</code>	Hardware interrupts
soft_interrupts	<code>int</code>	Software interrupts
syscalls	<code>int</code>	System calls

MetricsInfo.CpuFreq

CPU frequency (MHz).

FIELDS

NAME	TYPE	NOTES
current	float	Current frequency in MHz
min	float	Minimum frequency in MHz
max	float	Maximum frequency in MHz