Mobile Manager Reference Guide

Open Source GPRS/3G device management

Mobile Manager Reference

1 Mobile Manager description....2
2 Mobile Manager dbus service....3
2.1 Manager object interfaces....3
2.1.1 Controller Interface....3
2.1.1.1 Interface description....3
2.1.1.2 Methods 3
2.1.1.3 Signals....4
2.1.2 Dialer Interface....8
2.1.2.1 Interface description....8
2.1.2.2 Methods 8
2.1.2.3 Signals....8
2.2 Device object interfaces....9
2.2.1 DeviceAuth Interface....9
2.2.1.1 Interface description....9
2.2.1.2 Methods....9
2.2.2 DeviceInfo Interface....10
2.2.2.1 Interface description....10
2.2.2.2 Methods 11
2.2.3 DeviceState Interface....12
2.2.3.1 Interface description....12
2.2.3.2 Methods 12
2.2.4 DeviceXZone Interface....15
2.2.4.1 Methods 15

1 Mobile Manager description

Mobile Manager is an abstraction layer for GPRS and 3G devices. It launches, manages and takes statistics of GPRS and 3G connections.

Its main features are:

- Plug & Play device support. Your GPRS/3G devices are automatically detected and configured.
- PIN/PUK management. Mobile Manager will take care about authentication, code changes, card activations, etc.
- Device status control and card information. Is it attached? Is it on? Configured? Ready? You can also know about carrier selection, network information, signal strength...
- Connection establishment and control.

It is engineered as a Dbus service. It creates a Dbus object when a new device is connected, and automatically destroy it when the device is removed. It is also event based, so you can handle different signals and connect them to your apps.

2 Mobile Manager dbus service

2.1 Manager object interfaces

2.1.1 Controller Interface

2.1.1.1 Interface description

The controller detects any mobile device inserted in your system and configure it if it's supported.

There are two groups of signals, emitted by the active device and emitted by all devices. The controller

always has active device that it's the device by default. You can change it with the controller API.

2.1.1.2 Methods

GetAvailableDevices() -> (ao)

Return:

  (ao) -> an array of dbus objects representing all devices detected by Mobile Manager

FromDevIdGetObject(s: dev_id) -> (o)

dev_id : represent the id of the device

Return:

  (o) -> a dbus object representing the device referenced by dev_id

GetActiveDevice() -> (o)

Return:

  (o) -> a dbus object representing the active device

SetActiveDevice(s: device_obj_path) -> (b)

device_obj_path : The device object path will be the active device

Return :

  (b) -> True if the action has been successful, False if there was an error.

2.1.1.3 Signals

ActiveDevCardStatusChanged(i: status)

Desc : Emitted when the status of the active device has changed

~> status :

CARD_STATUS_ERROR = -1
CARD_STATUS_NO_DETECTED = 0
CARD_STATUS_DETECTED = 10
CARD_STATUS_CONFIGURED = 20
CARD_STATUS_NO_SIM = 25
CARD_STATUS_PIN_REQUIRED = 30
CARD_STATUS_PUK_REQUIRED = 40
CARD_STATUS_OFF = 50
CARD_STATUS_ATTACHING = 60
CARD_STATUS_READY = 70

ActiveDevTechStatusChanged(i: status)

Desc : Emitted when the card technology of the active device has changed

~> status :

CARD_TECH_GSM = 0
CARD_TECH_GSM_COMPACT = 1
CARD_TECH_UMTS = 2
CARD_TECH_HSPA = 3

ActiveDevModeStatusChanged(i: status)

Desc : Emitted when the card mode of the active device has changed

~> status :

CARD_TECH_SELECTION_GPRS = 0
CARD_TECH_SELECTION_UMTS = 1
CARD_TECH_SELECTION_GRPS_PREFERED = 2
CARD_TECH_SELECTION_UMTS_PREFERED = 3
CARD_TECH_SELECTION_NO_CHANGE = 4
CARD_TECH_SELECTION_AUTO = 5

ActiveDevDomainStatusChanged(i: status)

Desc : Emitted when the domain of the active device has changed

~> status :

CARD_DOMAIN_CS = 0

CARD_DOMAIN_PS = 1

CARD_DOMAIN_CS_PS = 2

CARD_DOMAIN_ANY = 4

ActiveDevSignalStatusChanged(i: status)

Desc : Emitted when the signal level of the active device has changed

~> status : signal level of the active device

ActiveDevPinActStatusChanged(b: status)

Desc : Emitted when the PIN activation status of the active device has changed

~> status : True if PIN is activate , False when it's deactivate

ActiveDevRoamingActStatusChanged(b: status)

Desc : Emitted when the active device is in roaming

~> status : True if device is in roaming , False it's not.

ActiveDevCarrierChanged(s: carrier_name)

Desc : Emitted when the carrier name of the active device has changed

~> carrier : Carrier name

ActiveDevCarrierSmStatusChanged(i: status)

ActiveDevXZoneChanged(s: xzone_name)

DevCardStatusChanged(s: device, i: status)

Desc : Emitted when the status any device has changed

~> device : the device id

~> status :

CARD_STATUS_ERROR = -1

CARD_STATUS_NO_DETECTED = 0

CARD_STATUS_DETECTED = 10

CARD_STATUS_CONFIGURED = 20

CARD_STATUS_NO_SIM = 25

CARD_STATUS_PIN_REQUIRED = 30

CARD_STATUS_PUK_REQUIRED = 40

CARD_STATUS_OFF = 50

CARD_STATUS_ATTACHING = 60

CARD_STATUS_READY = 70

DevTechStatusChanged(s: device, i: status)

Desc : Emitted when the card technology of any active device has changed

~> device : the device id

~> status :

CARD_TECH_GSM = 0

CARD_TECH_GSM_COMPACT = 1

CARD_TECH_UMTS = 2

CARD_TECH_HSPA = 3

DevModeStatusChanged(s: device, i: status)

Desc : Emitted when the card mode of any device has changed

~> device : the device id

~> status :

CARD_TECH_SELECTION_GPRS = 0

CARD_TECH_SELECTION_UMTS = 1

CARD_TECH_SELECTION_GRPS_PREFERED = 2

CARD_TECH_SELECTION_UMTS_PREFERED = 3

CARD_TECH_SELECTION_NO_CHANGE = 4

CARD_TECH_SELECTION_AUTO = 5

DevDomainStatusChanged(s: device, i: status)

Desc : Emitted when the domain of any device has changed

~> device : the device id

~> status :

CARD_DOMAIN_CS = 0

CARD_DOMAIN_PS = 1

CARD_DOMAIN_CS_PS = 2

CARD_DOMAIN_ANY = 4

DevSignalStatusChanged(s: device, i: status)

Desc : Emitted when the signal level of any device has changed

~> device : the device id

~> status : signal level of the active device

DevPinActStatusChanged(s: device, b: status)

Desc : Emitted when the PIN activation status of any device has changed

~> device : the device id

~> status : True if PIN is activate , False when it's deactivate

DevRoamingActStatusChanged(s: device, b: status)

Desc : Emitted when any device is in roaming

~> device : the device id

~> status : True if device is in roaming , False it's not.

DevCarrierChanged(s: device, s: carrier_name)

Desc : Emitted when the carrier name of any device has changed

~> device : the device id

~> carrier : Carrier name

DevCarrierSmStatusChanged(s: device, i: status)

~> device : The device id

~> status :

DevXZoneChanged(s: device, s: xzone_name)

~> device : The device id

~> xzone_name : The xzone name

ActiveDeviceChanged(s: device)

Desc : Emitted when any device has changed

~> device : The device id

AddedDevice(s: device)

Desc : Emitted when a device is added to mobile manager device list

~> device : The device id

RemovedDevice (s: device)

Desc : Emitted when a device is removed from mobile manager device list

~> device : The device id

SupportedDeviceDetected (s: device)

Desc : Emitted when a device is detected by mobile manager and it's supported.

~> device : The device id

2.1.2 Dialer Interface

2.1.2.1 Interface description

This interface is a wrapper of ppp management tasks. With this interface is possible start/stop a ppp connection using the active device of Mobile Manager by default and receive events about it status.

2.1.2.2 Methods

Start (s: username, s: password, s: apn, b: auto_dns, s: primary_dns, s: secundary_dns, s: dns_suffixes)

Desc : Start a ppp connection with params

username : username string. If there isn't username send ''

password: password string . If there isn't password send ''

auto_dns: True : use peer dns's , False: use parameters information

primary_dns: primary dns. If there isn't primary dns send ''

secundary_dns: primary dns. If there isn't secundary dns send ''

dns_suffixes: dns suffixes. If there isn't dns suffixes send ''

Stop ()

Desc : Stop the ppp connection

Status () -> (i)

Desc : Status of the ppp connection

Return :

  -> 0 Disconnected, 1 Connected, 2 Connecting, 3 Disconnecting

2.1.2.3 Signals

Connected ()

Connecting ()

Disconnected ()

Disconnecting ()

Stats (i: recv_bytes, i: sent_bytes, d: interval_time)
Desc : Report ppp connection stats
~> recv_bytes : Received bytes in "interval_time"
~> sent_bytes : Sent bytes in "interval_time"
~> interval_time : Interval time

2.2 Device object interfaces

2.2.1 DeviceAuth Interface

2.2.1.1 Interface description

This interface helps to the developer with the PIN/PUK management

2.2.1.2 Methods

SendPIN(s: pin) -> (b)
Desc : Send the PIN code to the device
pin : PIN code
Return :
(b) : True if the action has been successful, False if there was an error.

SetPIN(s: old_pin, s: new_pin) -> (b)
Desc : Set new PIN to the SIM card
old_pin : Old PIN code
new_pin : New PIN code

Return :
  (b) : True if the action has been successful, False if there was an error.

SetPINActive(s: pin, b: active) -> (b)
Desc : Active/Deactive PIN in the SIM card
pin :
active :
Return :
  (b) : True if the action has been successful, False if there was an error.

IsPINActive() -> (b)
Desc : Report the PIN activation status
Return :
  (b) : True is active, False is deactive

PINStatus() -> (i)
Return : Report the PIN Status
  (i) :
    PIN_STATUS_WAITING_PIN = 1
    PIN_STATUS_WAITING_PUK = 2
    PIN_STATUS_READY = 3
    PIN_STATUS_NO_SIM = 4
    PIN_STATUS_SIM_FAILURE = 5

SendPUK(s: puk, s: pin) -> (b)
Desc : Send PUK code to the SIM card
puk : PUK code
pin : PIN code
Return :
  (b) : True if the action has been successful, False if there was an error.

2.2.2 DeviceInfo Interface

2.2.2.1 Interface description

This interface report information to the developer about the device.

2.2.2.2 Methods

GetCapabilities() -> (as)

Desc : Each device has its own capabilities. These capabilities determine the access to the interfaces .

Some devices, as bluetooth or serial port, hasn't Auth management interface or Xzone interface, for example.

Return :

(as) : Array with the capabilities list. Each capability is returned in dbus URI format.

HasCapability(s: capability) -> (b)

capability : Capability

Return :

(b) True if the device has this capability, False if hasn't it.

GetDataDevicePath() -> (s)

Return :

(s) : Return the data device port used for establish the ppp connection

GetConfDevicePath() -> (s)

Return :

(s) : Return the conf device port used for communicate with the device with AT commands

GetVelocity() -> (i)

Return :

(s) : Return the device velocity

SetVelocity(i: velocity)

velocity : Velocity

GetHardwareFlowControl() -> (b)

Return :

(b) : True/False if has/hasn't hardware flow control

SetHardwareFlowControl(b: value)

value : True/False if you want active/deactive hardware flow control

GetHardwareErrorControl() -> (b)

Return :

(b) : True/False if has/hasn't hardware error control

SetHardwareErrorControl(b: value)

value : True/False if you want active/deactive hardware error control

GetHardwareCompress() -> (b)

Return :

(b) : True/False if has/hasn't hardware compress

SetHardwareCompress(b: value)

value : True/False if you want active/deactive hardware compress

GetPrettyName() -> (s)

Return :

(s) : Return the device pretty name

GetPriority() -> (i)

Return :

(i) : Return the device priority

2.2.3 DeviceState Interface

2.2.3.1 Interface description

This interface report information about the state of the device.

2.2.3.2 Methods

EmitStatusSignals()

Desc : Mobile manager emit again all status signals. Useful when you star up your client and need

all status information.

GetSingal() -> (i)

Desc : Get the device signal level.

Return :

(i) : Return the signal level

GetCarrierList() -> (a(isssi)aiai)

Desc : (ASYNC method) GetCarrierList return all information about the carriers that your device

has detected.

Return :

a(isssi) : Array with carrier informations

ai : Supported modes

ai : Supported formats

GetCarrier() -> (s)

Return :

(s) : Return the carrier name

IsOn() -> (b)

Return :

(b) : True/False if the device is on/off.

TurnOn() -> (b)

Return :

(b) : True/False if the device is turn on/off.

TurnOff() -> (b)

Return :

(b) : True/False if the device is turn off/on.

GetNetInfo() -> (iiisi)

Desc : GetNetInfo return network informations

Return :

(i) : Tech in use

(i) : Card mode

(i) : Card domain

(s) : Carrier

(i) : Carrier mode

GetModeDomain() -> (ii)

Return :

(i) : Card mode

(i) : Card domain

SetModeDomain(i: mode, i: domain) -> (b)

mode : Card mode

domain : Card domain

Return :

(b) : True if the action has been successful

GetCardInfo() -> (as)

Desc : GetCardInfo return the information reported by the device about itself.

Return :

(as) : An array of strings return from ATI command

GetCardStatus() -> (i)

Desc : Return the device status

Return :

(i) : CARD_STATUS_ERROR = -1

CARD_STATUS_NO_DETECTED = 0

CARD_STATUS_DETECTED = 10

CARD_STATUS_CONFIGURED = 20

CARD_STATUS_NO_SIM = 25

CARD_STATUS_PIN_REQUIRED = 30

CARD_STATUS_PUK_REQUIRED = 40
CARD_STATUS_OFF = 50
CARD_STATUS_ATTACHING = 60
CARD_STATUS_READY = 70

SetCarrier(i: carrier_id, i: tech) -> (b)
carrier_id : carried id
tech : tech
Return:
(b) : True if the action has been successful

SetCarrierAutoSelection() -> (b)
Desc : Set the device in auto selection mode.
Return :
(b) : True if the action has been successful

IsCarrierAuto() -> (b)
Return:
(b) : True if the device is in auto selection mode

IsAttached() -> (b)
Return:
(b) : True if the device is attached to any network

GetAttachState() -> (i)
Return:
(i) :

IsRoaming() -> (b)
Return :
(b) : True if the device is roaming.

2.2.4 DeviceXZone Interface

2.2.4.1 Methods

GetXZone() -> (s)

Return :

  (s) : Zone name