Device Discovery

Published 2014-10-28 | (Compatible with SDK 4.5,5.0,5.1 and 2013,2014 models)

> This API allows developers to get a list of discovered AllShare devices with a specified ID, domain and NIC.

Contents

device discovery

Introduction

This API allows developers to get a list of discovered AllShare devices with a specified ID, domain and NIC.

It also provide to get device discovery update notification.

Summary of Interfaces and Methods

Interface	Method
DeviceDiscoveryCallback	void **ondeviceadded** (Device device) void **ondeviceremoved** (Device device)
Icon	
DeviceFinder	Device **getDevice** (DeviceType deviceType, DeviceId id) DeviceArray **getDeviceList** (deviceType deviceType) DeviceArray **getDeviceListByDomain** (DeviceType deviceType, DeviceDmain domain) DeviceArray **getDeviceListByNIC** (DeviceType deviceType, DOMString nic) void **refresh** () long **addDeviceDiscoveryListener** (DeviceDiscoveryCallback deviceDiscoveryCallback) void **removeDeviceDiscoveryListener** (long deviceDiscoveryListener)
Device	

Type Definition

DeviceId

The device identifier

typedef DOMString DeviceId;

DeviceArray

DeviceArray identifier

typedef sequence<Device> DeviceArray;

IconArray

IconArray identifier

typedef sequence<Icon> IconArray;

DeviceType

Specifies the device type which can be discovered.

```
enum DeviceType {
    "AVPLAYER",
    "IMAGEVIEWER",
    "FILERECEIVER",
    "MEDIAPROVIDER",
    "MEDIARECEIVER",
    "TVCONTROLLER",
    "VIEWCONTROLLER",
    "UNKNOWN"
};
```

AVPLAYER: Audio and video player
IMAGEVIEWER: Image viewer
FILERECEIVER: File receiver
MEDIAPROVIDER: Media provider
MEDIARECEIVER: Media receiver
TVCONTROLLER: TV controller
VIEWCONTROLLER: View controller
UNKNOWN: Unkwon device type

DeviceDomain

Specifies the network domain which the device is located.

```
enum DeviceDomain {
    "LOCAL_NETWORK",
    "MY_DEVICE",
    "REMOTE_NETWORK",
    "UNKNOWN"
};
```

LOCAL_NETWORK: Located in local network area
MY_DEVICE: Located in my device
REMOTE_NETWORK: Located in internet area
UNKNOWN: Located in unknown area

Interfaces

DeviceDiscoveryCallback

Provides a generic callback for notification about device discovery events

```
[Callback, NoInterfaceObject] interface DeviceDiscoveryCallback {
    void ondeviceadded(in Device device);
    void ondeviceremoved(in Device device);
};
```

METHODS

ondeviceadded()

Invoked when a new device is appeared in the network.

Signature:	

void ondeviceadded(in Device device);

Parameters:	

device

Optional: No.
Nullable: No.
Type: Device.
Description: A newly discovered device.

ondeviceremoved()

Invoked when a device is going to disappear in the network

Signature:	

void ondeviceremoved(in Device device);

Parameters:	

device

Optional: No.
Nullable: No.
Type: Device.
Description: Device which is going to be disappeared.

Icon

Represents to the icon information.

```
[NoInterfaceObject] interface Icon {
  readonly attribute long depth;
  readonly attribute long height;
  readonly attribute long width;
  readonly attribute DOMString mimeType;
  readonly attribute DOMString iconUri;
};
```

ATTRIBUTES

readonly long depth

Color depth of the icon by bits per pixel.

This attribute is readonly.

readonly long height

Height of the icon.

This attribute is readonly.

readonly long width

Width of the icon.

This attribute is readonly.

readonly DOMString mimeType

MIME type of the icon.

This attribute is readonly.

readonly DOMString iconUri

Uri of the icon.

This attribute is readonly.

DeviceFinder

Provides methods for finding AllShare Devices

```
[NoInterfaceObject] interface DeviceFinder {
    Device      getDevice(in DeviceType deviceType, in DeviceId id) raises(WebAPIException);
    DeviceArray getDeviceList(in DeviceType deviceType) raises(WebAPIException);
    DeviceArray getDeviceListByDomain(in DeviceType deviceType, in DeviceDomain domain) raise
s(WebAPIException);
    DeviceArray getDeviceListByNIC(in DeviceType deviceType, in DOMString nic) raises(WebAPIEx
ception);
    void        refresh() raises(WebAPIException);
    long        addDeviceDiscoveryListener(in DeviceDiscoveryCallback deviceDiscoveryCallback) rais
es(WebAPIException);
    void        removeDeviceDiscoveryListener(in long deviceDiscoveryListener) raises(WebAPIExcepti
on);
};
```

METHODS

getDevice()

Provides a device with a specified device ID and a specified device type.

Signature:	

Device getDevice(in DeviceType deviceType, in DeviceId id);

If no device has been discovered, then null will be returned.

Parameters:	

deviceType

Optional: No.
Nullable: No.
Type: DeviceType.
Description: Specifies type of device

id

Optional: No.
Nullable: No.
Type: DeviceId.
Description: Specifies device id

Return value:	

Specified device object

Exceptions:	

WebAPIException:

with error type “UnknownError”, In any other error case.

with error type “SecurityError”, if this functionality is not allowed.

with error type “TypeMismatchError”, if any input parameter is not compatible with the expected type for that parameter.

Code example

```
var serviceProvider; // it is assumed that you obtained serviceProvider. For further details,
                     // see the creatServiceProvider(..).
var deviceId;        // it is assumed that media provider's device Id is already distingushed

try {
   var deviceFinder = serviceProvider.getDeviceFinder();
   var device = deviceFinder.getDevice("MEDIAPROVIDER", deviceId);
                     // Print out the device's id
   console.log(device.id);
} catch(e) {
   console.log(e.name);
}
```

getDeviceList()

Provides a list of discovered devices with a specified device type.

Signature:	

DeviceArray getDeviceList(in DeviceType deviceType);

If no devices of given device type have been discovered, then the empty list will be returned.

Parameters:	

deviceType

Optional: No.

Nullable: No.

Type: DeviceType.

Description: Specifies devices type

Return value:	

DeviceArray Specified devices array

Exceptions:	

WebAPIException:

with error type UnknownError, In any other error case.

with error type SecurityError, if this functionality is not allowed.

with error type TypeMismatchError, if any input parameter is not compatible with the expected type for that parameter.

Code example

```
var serviceProvider; // it is assumed that you obtained serviceProvider.
                    // For further details, see the createServiceProvider() or getServiceProvider().
try {
    var deviceFinder = serviceProvider.getDeviceFinder();
    var devices = deviceFinder.getDevices("MEDIAPROVIDER");
                    // Print out how many media provider devices are available
    console.log(devices.length);
} catch(e) {
    console.log(e.name);
}
```

getDeviceListByDomain()

Provides a list of discovered devices with specified device type within specified network domain.

Signature:	

DeviceArray getDeviceListByDomain(in DeviceType deviceType, in DeviceDomain domain);

If no devices of given device type and given domain have been discovered, then an empty list will be returned.

Parameters:	

deviceType

Optional: No.
Nullable: No.
Type: DeviceType.
Description: Provides type of devices specified by their domain.

domain

Optional: No.
Nullable: No.
Type: DeviceDomain.
Description: Specifies devices domain

Return value:	

DeviceArray Specified devices array

Exceptions:	

WebAPIException:

with error type UnknownError, In any other error case.
with error type SecurityError, if this functionality is not allowed.
with error type TypeMismatchError, if any input parameter is not compatible with the expected type for that parameter.

Code example

```
    var serviceProvider; // it is assumed that you has obtained a serviceProvider object.
                         // For further details, see the createServiceProvider() or getServiceProvider().
    try {
       var deviceFinder = serviceProvider.getDeviceFinder();
       var devices = deviceFinder.getDevicesByDomain("MEDIAPROVIDER", "LOCAL_NETWORK");
                         // Print out how many media provider devices are available on local network
       console.log(devices.length);
    } catch(e) {
       console.log(e.name);
    }
getDeviceListByNIC()
```

Provides a list of discovered devices with a specified type within a specified network interface controller (NIC).

Signature:	

DeviceArray getDeviceListByNIC(in DeviceType deviceType, in DOMString nic);

If no devices of given device type and given NIC have been discovered, then an empty list will be returned.

Parameters:	

deviceType

Optional: No.
Nullable: No.
Type: DeviceType.
Description: Specifies devices type

nic

Optional: No.
Nullable: No.
Type: DOMString.
Description: The name of NIC communicating with the device

Return value:	

DeviceArray Specified devices array

Exceptions:	

WebAPIException:

with error type UnknownError, in any other error case.
with error type SecurityError, if this functionality is not allowed.
with error type TypeMismatchError, if any input parameter is not compatible with the expected type for that parameter.

Code example

```
var serviceProvider; // it is assumed that you has obtained a serviceProvider object.
                     // For further details, see the createServiceProvider() or getServiceProvider().
try {
   var deviceFinder = serviceProvider.getDeviceFinder();
   var devices = deviceFinder.getDevicesByNIC("MEDIAPROVIDER", "Wireless LAN");
                     // Print out how many media provider devices are available on wireless LAN network
   console.log(devices.length);
} catch(e) {
   console.log(e.name);
}
```

refresh()

Refreshes the list of devices on the network.

Signature:	

```
void refresh();
```

Exceptions:	

WebAPIException:
with error type UnknownError, In any other error case.
with error type SecurityError, if this functionality is not allowed.

Code example

```
var serviceProvider; // it is assumed that you obtained serviceProvider.
                     // For further details, see the createServiceProvider() or getServiceProvider().
try {
   var deviceFinder = serviceProvider.getDeviceFinder();
   var providers = deviceFinder.getDevices("MEDIAPROVIDER");
   if (providers.length == 0) {
                     // Refreshs the device list and retreives the provider list again.
      deviceFinder.refresh();
      providers = deviceFinder.getDevices("MEDIAPROVIDER");
   }
} catch(e) {
   console.log(e.name);
}
```

addDeviceDiscoveryListener()

Registers device discovery event listener.

Signature:	

```
long addDeviceDiscoveryListener(in DeviceDiscoveryCallback deviceDiscoveryCallback);
```

Parameters:	

deviceDiscoveryCallback
Optional: No.
Nullable: No.
Type: DeviceDiscoveryCallback.
Description: Callback function called if the event occurred.

Return value:	

long Event listener id

Exceptions:	

WebAPIException:
with error type UnknownError, in any other error case.
with error type SecurityError, if this functionality is not allowed.
with error type TypeMismatchError, if any input parameter is not compatible with the expected type for that parameter.

Code example

```
var serviceProvider; // it is assumed that you has obtained a serviceProvider object.
                     // For further details, see the createServiceProvider() or getServiceProvider().
var monitoringCB = {
   ondeviceadded: function (device) {
      console.log("new device is appeared :" + device.name);
   },
   ondeviceremoved: function (device) {
      console.log("a device is disappeared :" + device.name);
   }
}

try {
   var listenerId = serviceProvider.getDeviceFinder().addDeviceDiscoveryListener(monitoringCB);
} catch(e) {
   console.log(e.name);
}
```

removeDeviceDiscoveryListener()

Clears device discovery event listener

Signature:	

void removeDeviceDiscoveryListener(in long deviceDiscoveryListener);

Parameters:	

deviceDiscoveryListener
Optional: No.
Nullable: No.
Type: long.
Description: Event listener id

Exceptions:	

WebAPIException:
with error type UnknownError, In any other error case.
with error type InvalidValuesError, if argument is invalid.
with error type SecurityError, if this functionality is not allowed.
with error type TypeMismatchError, if any input parameter is not compatible with the expected type for that parameter.

Code example

```
var serviceProvider; // it is assumed that you has obtained a serviceProvider object.
                     // For further details, see the createServiceProvider() or getServiceProvider().
var listenerId;      // it is assumed that his listener ID is returned from addDeviceDiscoveryListener().
var monitoringCB = {
   ondeviceadded: function (device) {
      console.log("new device is appeared :" + device.name);
   },
   ondeviceremoved: function (device) {
      console.log("a device is disappeared :" + device.name);
   }
}

try {
   var listenerId = serviceProvider.getDeviceFinder().addDeviceDiscoveryListener(monitoringCB);
} catch(e) {
   console.log(e.name);
}
```

Device

Provides the information about a device.

```
[NoInterfaceObject] interface Device {
   readonly attribute DeviceId id;
   readonly attribute DeviceDomain deviceDomain;
   readonly attribute DeviceType deviceType;
   readonly attribute IconArray iconArray;
   readonly attribute DOMString ipAddress;
   readonly attribute DOMString modelName;
   readonly attribute DOMString name;
   readonly attribute DOMString nic;
};
```

For retrieving a device object, see the getDevice(), getDevices(), getDeviceByDomain() or getDeviceByNIC().

ATTRIBUTES

readonly DeviceId id

> Specifies the unique ID of the device.
>
> This attribute is readonly.

readonly DeviceDomain deviceDomain

> Specifies the device domain.
>
> This attribute is readonly.

readonly DeviceType deviceType

> Specifies the device type.
>
> This attribute is readonly.

readonly IconArray iconArray

Specifies icons associated with the device.

This attribute is readonly.

readonly DOMString ipAddress

Specifies the device IPv4 adress.

This attribute is readonly.

readonly DOMString? modelName

Specifies the device model name.

This attribute is readonly.

readonly DOMString name

Specifies the device name.

This attribute is readonly.

readonly DOMString nic

Specified NIC (Network Interface Controller) name of the device.

This attribute is readonly.