provider

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

> This interface provides interfaces to browse and search media contents shared by DLNA devices and to download a content from remote DMS.

Contents

Introduction

This API allows developers to share media contents between DLNA devices.

It provides the operations as follows:

Browse or search contents which are provided from all available digital media servers (DMS)

Download a content from other DMS to my local device.

Summary of Interfaces and Methods

Interface	Method
MediaProviderSuccessCallback	void onsuccess (ItemList itemList, boolean endOfItems, DeviceId providerId)
MediaProviderErrorCallback	void onerror (WebAPIError error, DeviceId providerId)
MediaProvider	void browse (Item folderItem, unsigned long startIndex, unsigned long requestCount MediaProviderSuccessCallback browseCallback, MediaProviderErrorCallback errorCallback, AbstractFilter browseFilter, SortMode sortMode) void search (DOMString keyword, unsigned long startIndex, unsigned long requestCount, MediaProviderSuccessCallback successCallback, MediaProviderErrorCallback errorCallback, AbstractFilter searchFilter)

Type Definition

ItemList

Array of Items which are provided from a device on the network.

```
typedef sequence<Item> ItemList;
```

Interfaces

MediaProviderSuccessCallback

Callback invoked when the retrieving operations of a Provider object is successfully done.

```
[Callback=FunctionOnly, NoInterfaceObject] interface MediaProviderSuccessCallback {
    void onsuccess(in ItemList itemList, in boolean endOfItems, in DeviceId providerId);
};
```

METHODS

onsuccess()

Callback function when the retrieving request is successful.

Signature:	

```
void onsuccess(in ItemList itemList, in boolean endOfItems, in DeviceId providerId);
```

Parameters:	

itemList
- Optional: No.
- Nullable: No.
- Type: ItemList.
- Description: A list of items. If there is no item, then it will return an empty list.

endOfItems
- Optional: No.
- Nullable: No.
- Type: boolean.
- Description: The end of items flag.

providerId
- Optional: No.
- Nullable: No.
- Type: DeviceId.
- Description: The identifier of the provider device.

MediaProviderErrorCallback

Generic error callback for provider related operations.

```
[Callback=FunctionOnly, NoInterfaceObject] interface MediaProviderErrorCallback {
    void onerror(in WebAPIError error, in DeviceId providerId);
};
```

METHODS

onerror()

Callback function invoked when error occurs.

Signature:	

```
void onerror(in WebAPIError error, in DeviceId providerId);
```

Parameters:	

error
- Optional: No.
- Nullable: No.
- Type: WebAPIError.
- Description: WebAPIError object which indicates error type and message.

providerId
- Optional: No.
- Nullable: No.
- Type: DeviceId.
- Description: The identifier of the provider device.

MediaProvider

Interface provides a way to share media contents between devices. MediaProvider is a specific type of device. Developers can obtain the object by using method that retrieves devices.

```
[NoInterfaceObject] interface MediaProvider : Device {
  readonly attribute boolean isSearchable;
  readonly attribute Item rootFolder;
  void browse(in Item folderItem,
              in unsigned long startIndex,
              in unsigned long requestCount,
              in MediaProviderSuccessCallback browseCallback,
              in optional MediaProviderErrorCallback errorCallback,
              in optional AbstractFilter browseFilter,
              in optional SortMode sortMode) raises(WebAPIException);
  void search(in DOMString keyword,
              in unsigned long startIndex,
              in unsigned long requestCount,
              in MediaProviderSuccessCallback successCallback,
              in optional MediaProviderErrorCallback errorCallback,
              in optional AbstractFilter searchFilter) raises(WebAPIException);
};
```

ATTRIBUTES

readonly boolean isSearchable

Specifies whether the provider is supporting search capability.

This attribute is readonly.

readonly Item rootFolder

Object of the root folder.

This attribute is readonly.

METHODS

browse()

Get the item list in a folder by the specified starting index and count.

Signature:	

```
void browse(in Item folderItem,
            in unsigned long startIndex,
            in unsigned long requestCount,
            in MediaProviderSuccessCallback browseCallback,
            in optional MediaProviderErrorCallback errorCallback,
            in optional AbstractFilter browseFilter,
            in optional SortMode sortMode);
```

The error callback is launched with these error types:

NotConnectedServiceError: if the application doesn't connect to allshare framework.

NotFoundError: if the item does not exist.

InvalidStateError: if the device or state is invalid or unusable.

NetworkNotAvailableError: if the network is not available.

TimeOutError: If the request is timeout.

InvalidValuesError: If any input parameter contains invalid values.
UnknownError: In any other error case.

Parameters:	

folderItem
Optional: No.
Nullable: No.
Type: Item.
Description: The item which represents a folder to be retrieved.

startIndex
Optional: No.
Nullable: No.
Type: unsigned long.
Description: The starting zero-based offset to enumerate children under the specific item.

requestCount
Optional: No.
Nullable: No.
Type: unsigned long.
Description: The requested number of entries

browseCallback
Optional: No.
Nullable: No.
Type: MediaProviderSuccessCallback.
Description: Success callback invoked when the browse operation is successful, and it returns a set of items matching the filter

errorCallback
Optional: Yes.
Nullable: Yes.
Type: MediaProviderErrorCallback.
Description: Generic error callback for provider related operations.

browseFilter
Optional: Yes.
Nullable: Yes.
Type: AbstractFilter
Description: Filter applied for the browsing. An AttributeFilter object constructed with (itemType, null , ItemType[]) can be valid as a parameter.

sortMode
Optional: Yes.
Nullable: Yes.
Type: SortMode.
Description: Sorting mode which contents will be listed by a specific order after the browsing is finished. Sorting with a title, date, artist and albumTitle as the attribute can be supported on the tizen web platform only. Other web platform will ignore this parameter.

Exceptions:	

WebAPIException:
with error type NotSupportedError, if this feature is not supported.
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().
//Define browse callback
function browseCB(list, endOfItem, providerId) {
   // the item list can be retrieved here
}

function errorCB(error, deviceId) {
   console.log(error.message);
}

// Define a filter to browse a video type only
var filter = new webapis.AttributeFilter("itemType", null, {"VIDEO"});

// Define a sort mode
var mode = new webapis.SortMode("title", "ASC");

try {
   var providers = serviceProvider.getDeviceFinder().getDevices("MEDIAPROVIDER");
   if (providers.length > 0) {
      // retrieves first DMS from the root folder
      providers[0].browse(providers[0].rootFolder, 0, 10, browseCB, errorCB, filter, mode);
   }
} catch(e) {
   console.log(e.message);
}
```

search()

Search content items with a keyword.

Signature:	

```
void search(in DOMString keyword,
            in unsigned long startIndex,
            in unsigned long requestCount,
            in MediaProviderSuccessCallback successCallback,
            in optional MediaProviderErrorCallback errorCallback,
            in optional AbstractFilter searchFilter);
```

The error callback is launched with these error types:

NotConnectedServiceError: if the application doesn't connect to allshare framework.
NotFoundError: if the item does not exist.
InvalidStateError: if the device or state is invalid or unusable.
NetworkNotAvailableError: if the network is not available.
TimeOutError: If the request is timeout.
InvalidValuesError: If any input parameter contains invalid values.
UnknownError: In any other error case.

Parameters:	

keyword
Optional: No.

Nullable: No.
Type: DOMString.
Description: The keyword which will be used to search.

startIndex
Optional: No.
Nullable: No.
Type: unsigned long.
Description: The starting zero-based offset to enumerate children under the specific item.

requestCount
Optional: No.
Nullable: No.
Type: unsigned long.
Description: The requested number of entries

successCallback
Optional: No.
Nullable: No.
Type: MediaProviderSuccessCallback.
Description: Success callback invoked when the search operation is successful, and it returns a set of items matching the filter

errorCallback
Optional: Yes.
Nullable: Yes.
Type: MediaProviderErrorCallback.
Description: Generic error callback for provider related operations.

searchFilter
Optional: Yes.
Nullable: No.
Type: AbstractFilter
Description: Filter applied to the searched result. An AttributeFilter object constructed with (itemType, null, ItemType[]) can be valid as a parameter.

Exceptions:	

WebAPIException:
with error type NotSupportedError, if this feature is not supported.
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 keyword = "foo";

//Define browse callback
function searchCB(list, endOfItem, providerId) {
   // the item list can be retrieved here
}

function errorCB(error, device) {
   console.log(device + " raises " + error);
}

// Define a filter to browse a video type only
var filter = new webapis.AttributeFilter("itemType", null, {"VIDEO"});

try {
   var providers = serviceProvider.getDeviceFinder().getDevices("MEDIAPROVIDER");
   if (providers.length > 0) {
      // search the keyword on the first DMS.
      providers[0].search(keyword, 0, 40, searchCB, errorCB, filter);
   }
} catch(e) {
   console.log(e.message);
}
```