SES
Platform Services

Media Platform Solutions | AS-20502

Novaspread-S

Reference manual

Version:

0.53

Date issued:

21 August 2015

Important Notice

This document has been produced by SES Platform Services GmbH (SES PS). Certain product names or brand names may be trademarks or designations of their respective owners.

Liability/Copyright

SES Platform Services GmbH

Beta Straße 1-10

D-85774 Unterföhring

Germany

Cooperation

This document has been developed in cooperation with:

TARA Systems GmbH

Gmunder Str. 53

D-81379 München

Germany

SES
Platform Services

TABLE OF CONTENT

1. INTRODUCTION

1.1. Purpose of document

This document describes the required and provided interfaces of Novaspread-S in the scope of the Multiscreen product of SES Platform Services.

1.2. Document history

Version	Date	Author	Changes
0.53	2015-08-21	Manfred Schmidt Georg Kamjunke	Working draft

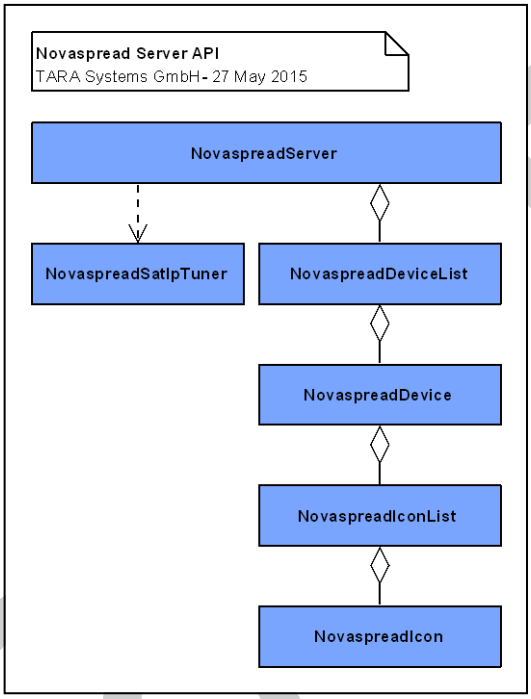
1.3. References

SPS; "AS-20001: Multiscreen"
SPS; "AS-20501: Novaspread-S"

2. PROVIDED API

The following section describes the Application Programming Interface (API) which is provided by Novaspread-S. To use the interface in an application include the NovaspreadServer.h.

The following diagram gives an overview of the classes provided by Novaspread-S.



2.1. Novaspread Basic Types

Novaspread uses the following basic types:

`NovaspreadTInt8` – A 8-bit signed integer
`NovaspreadTInt16` – A 16-bit signed integer
`NovaspreadTInt32` – A 32-bit signed integer
`NovaspreadTUInt8` – A 8-bit unsigned integer
`NovaspreadTUInt16` – A 16-bit unsigned integer
`NovaspreadTUInt32` – A 32-bit unsigned integer
`NovaspreadTBoolean` – A boolean type which can have the values `NOVASPREAD_TRUE` or `NOVASPREAD_FALSE`
`NOVASPREAD_NULL` – A null-reference

2.2. NovaspreadServer

A NovaspreadServer represents the Novaspread-S main class. Before the NovaspreadServer can be used it must be initialised with the function NovaspreadServerInit(). With the function NovaspreadServerDone() the NovaspreadServer is shutdown again.

With the function NovaspreadServerSetFriendlyName() the name of the SAT>IP server provided by the NovaspreadServer is defined.

After setting the host IP address by calling NovaspreadServerSetHostIpAddress(), the UPnP server and the UPnP device detection can be started by calling NovaspreadServerStart(). To access other (standard) SAT>IP

servers a device list of all SAT>IP servers can be retrieved with the function NovaspreadServerGetDeviceList(). One of these SAT>IP servers is selected with the function NovaspreadServerSelectDevice(). SAT>IP tuners are only accessed from the selected SAT>IP server. A SAT>IP tuner can be allocated with the function NovaspreadServerCreateSatIpTuner() and used by the host device to receive parts of a transport streams.

A ProcessRequestListener must be set at the NovapreadServer. Via this listener the NovaspreadServer informs the application that NovaspreadServerProcess() must be called form the main thread, so that NovaspreadServer can process internal data.

2.2.1. NovaspreadTServerInitParameters

This type defines initialization parameters for NovaspreadServer. It is recommended to initialize this struct with all 0 (see example below).

NovaspreadServer expects to find a logconfig.xml file within the DataPath. This file is used to configure logging functionality. If logconfig.xml is not found, NovaspreadServer is still initialized, but no loggings will be printed.

NovaspreadServer expects to find a deviceDescription.xml file within the DataPath. This file contains the device description that is used for UPnP advertisement. If this file is missing, NovaspreadServer can be initialized, but not be started. Don't add icons to the deviceDescription.xml. Use icons.xml instead.

NovaspreadServer expects to find a icons.xml file within the DataPath. This file contains the description of icons which will be used for UPnP advertisement.

SYNTAX

```
typedef struct
{
  const char *       DataPath;
  NovaspreadTUInt16  HttpPort;
  const char *       UniqueDeviceName;
  NovaspreadTBoolean EnableMemoryChecks;
  NovaspreadTBoolean EnableTestSupport;

} NovaspreadTServerInitParameters;
```

COMPONENTS

DataPath
Path to a directory within the local file system, were NovaspreadServer can store its configuration data.

HttpPort
The port that should be used for listening on incoming HTTP requests.

UniqueDeviceName
The unique device name to be used as identification of the UPnP server. The UDN must be supplied in the format 'uuid:[UUID]', whereas UUID is a Universally Unique Identifier string, like '550e8400-e29b-11d4-a716-446655440000'.

EnableMemoryChecks
NOVASPREAD_TRUE: memory checks are enabled. NOVASPREAD_FALSE: memory checks are disabled. It is recommended to disable memory checks on target platforms, because the checks themselves need an amount of memory.

EnableTestSupport
NOVASPREAD_TRUE: Test environment is enabled. The clock can be controlled in the test. NOVASPREAD_FALSE: Test environment is disabled.

EXAMPLE

```
NovaspreadTServerInitParameters initParameters;

 memset( &initParameters, 0, sizeof( initParameters ));
 initParameters.DataPath = "/data";
 NovaspreadServerInit( &initParameters );
```

2.2.2. NovaspreadTServerProcessRequestListener

A function of this type must be set at the NovaspreadServer by calling NovaspreadServerSetProcessRequestListener(). The listener is called every time NovaspreadServer must process internal data and therefore NovaspreadServerProcess() must be called. This listener can be called from various threads. So NovaspreadServerProcess() shall not be called from the listener. Instead the main thread shall be informed that a call to NovaspreadServerProcess() is needed. It is allowed to call NovaspreadServerProcess more often than the listener is called, but it is not required.

SYNTAX

```
typedef void
(* NovaspreadTServerProcessRequestListener ) (
  void * aContext );
```

PARAMETERS

aContext
The context is passed unchanged from NovaspreadServerSetProcessRequestListener().

SEE ALSO

NovaspreadServerSetProcessRequestListener()

2.2.3. NovaspreadServerGetVersion

This function returns a null-terminated ASCII string representing the version of the NovaspreadServer library. The returned version string is composed as follows:

```
<version>       ::= <major>.<minor>.<patch>[.<branchExt>][:<options>][-<revision>]
<major>         ::= <number>  (max 4 digits - no leading 0)
<minor>         ::= <number>  (2 digits - leading 0)
<patch>         ::= <number>  (2 digits - leading 0)
<revision>      ::= as returned by svnversion.exe
<options>       ::= <option>+
<option>        ::= <upperCaseChar>[<number>]
<branchExt>     ::= (<char>|<digit>|<underscore>)+
<number>        ::= <digit>+
<char>          ::= <upperCaseChar>|<lowerCaseChar>
<digit>         ::= 0..9
<upperCaseChar> ::= A..Z
<lowerCaseChar> ::= a..z
<underscore>    ::= _
```

SYNTAX

```
PUBLIC const char *
NovaspreadServerGetVersion(
  void );
```

RETURN VALUE

The NovaspreadServer version.

2.2.4. NovaspreadServerInit

This function initializes the NovaspreadServer. It must be called once, before any other NovaspreadServer function is called.

After initialization the NovaspreadServer is stopped. Call NovaspreadServerSetHostIpAddress() and NovaspreadServerStart() to start the NovaspreadServer.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerInit(
  NovaspreadTServerInitParameters * aInitParameters );
```

PARAMETERS

aInitParameters
The initialization parameter for NovaspreadServer.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

SEE ALSO

```
NovaspreadTServerInitParameters
NovaspreadServerDone()
NovaspreadServerSetHostIpAddress()
NovaspreadServerStart()
```

2.2.5. NovaspreadServerDone

This function shuts down the NovaspreadServer. After this function is called no other functions of the NovaspreadServer shall be called.

SYNTAX

```
PUBLIC void
NovaspreadServerDone(
  void );
```

SEE ALSO

```
NovaspreadServerInit()
```

2.2.6. NovaspreadServerProcess

This function must be called every time the ProcessRequestListener is called. It shall not be called from the ProcessRequestListener's context. Instead it shall be called from the main thread, like all other functions of NovaspreadServer.

NovaspreadServerProcess() performs internal processing, like handling of incoming RTPS requests.

SYNTAX

```
PUBLIC void
NovaspreadServerProcess(
  void );
```

SEE ALSO

```
NovaspreadTServerProcessRequestListener
NovaspreadServerSetProcessRequestListener()
```

2.2.7. NovaspreadServerSetProcessRequestListener

Sets a ProcessRequestListener. Only one ProcessRequestListener can be set. This function should be called only once after the initialization of the NovaspreadServer. The listener cannot be unset.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerSetProcessRequestListener(
  NovaspreadTServerProcessRequestListener aListener,
  void *                                  aContext );
```

PARAMETERS

aListener
The listener.

aContext
The context.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FASLE
otherwise.

SEE ALSO

NovaspreadTServerProcessRequestListener
NovaspreadServerProcess()

2.2.8. NovaspreadServerSetHostIpAddress

Sets the HostIpAddress.

This function can be called only if the NovaspreadServer is stopped.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerSetHostIpAddress(
  const char * aIpAddress );
```

PARAMETERS

aIpAddress
The IP address (in case of a multi-homed host) of the network adapter to be used for network communication. If INARIS_NULL or an empty string is passed, the first network adapter gets used.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadServerStart()
NovaspreadServerStop()

2.2.9. NovaspreadServerStart

Starts the NovaspreadServer. Starts the UPnP server and the Sat>IP server. If no host IP address was set before, the first network adapter is used.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerStart(
  void );
```

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE

otherwise

SEE ALSO

`NovaspreadServerStop()`

2.2.10. NovaspreadServerStop

Stops the NovaspreadClient. This function stops the player and the UPnP device detection. The server is unselected.

SYNTAX

```
PUBLIC void
NovaspreadServerStop(
  void );
```

SEE ALSO

`NovaspreadServerStart()`

2.2.11. NovaspreadServerFactoryReset

Resets the NovaspradServer to factory defaults.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerFactoryReset(
  void );
```

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.2.12. NovaspreadServerSetFriendlyName

Sets the friendly name used by the SAT>IP Server.

This function can be called only if the NovaspreadServer is stopped.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerSetFriendlyName(
  const char * aFriendlyName );
```

PARAMETERS

aFriendlyName
The friendly name as UTF-8 encoded string.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.2.13. NovaspreadServerGetDeviceList

Gets the currently available list of SAT>IP Servers. The DeviceList contains all (standard) SAT>IP server devices which have been detected in the local network at the moment when the function is called. The returned list is a copy and does not change, if e.g. a new SAT>IP server was found after getting the list. To get an updated list, the list must be released and retrieved again with this function. The SAT>IP Server provided by the NovaspreadServer is excluded from this list to avoid self-referencing.

When the DeviceList is no longer used, the function NovaspreadDeviceListRelease() must be called to release it.

SYNTAX

```
PUBLIC NovaspreadTDeviceList
NovaspreadServerGetDeviceList(
  void );
```

RETURN VALUE

A new DeviceList. NOVASPREAD_NULL if an error occurred.

SEE ALSO

NovaspreadTDeviceList
NovaspreadDeviceListRelease()

2.2.14. NovaspreadServerSelectDevice

With this function a SAT>IP server device is selected. SatIpTuners will only be used from this selected device.

If a call to this function changes the selected device, all currently connected SatIpTuners will be disconnected. If NOVASPREAD_NULL is passed as Device, SatIpTuners can no longer connect to a SAT>IP server.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadServerSelectDevice(
  NovaspreadTDevice aDevice );
```

PARAMETERS

aDevice
The SAT>IP server device to be selected.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

SEE ALSO

NovaspreadServerGetDeviceList()
NovaspreadServerCreateSatIpTuner()

2.2.15. NovaspreadServerGetSelectedDevice

Gets the currently selected SAT>IP server device.

SYNTAX

```
PUBLIC NovaspreadTDevice
NovaspreadServerGetSelectedDevice(
  void );
```

RETURN VALUE

The selected device. NOVASPREAD_NULL if no device has been selected.

SEE ALSO

NovaspreadServerSelectDevice()

2.2.16. NovaspreadServerCreateSatIpTuner

Creates a new SatIpTuner. A SatIpTuner can be used to receive transport stream data from a SAT>IP server, that is available in the local network. If the SatIpTuner is no longer used, call NovaspreadSatIpTunerRelease() to free it. The function call is non-blocking, i.e. it returns immediately. To actually allocate a SatIpTuner from the selected SAT>IP server device call the function NovaspreadSatIpTunerConnect().

SYNTAX

```
PUBLIC NovaspreadTSatIpTuner
NovaspreadServerCreateSatIpTuner(
  void );
```

RETURN VALUE

A new SatIpTuner. NOVASPREAD_NULL if an error occurred.

SEE ALSO

NovaspreadTSatIpTuner
NovaspreadSatIpTunerConnect()

2.3. NovaspreadDeviceList

A DeviceList holds Devices which have been found via UPnP device detection in the network. A DeviceList represents a snapshot of the Devices when the list is retrieved. The DeviceList is not changed afterwards even if new Devices are found or disappeared from the network. To update a list for the user interface simply retrieve the list again to get a current snapshot of this list.

2.3.1. NovaspreadTDeviceList

This type defines the **NovaspreadTDeviceList** handle.

SYNTAX

```
typedef struct NovaspreadTDeviceListStruct * NovaspreadTDeviceList;
```

2.3.2. NovaspreadDeviceListRelease

Releases this DeviceList. After calling this function the list shall no longer be accessed. Each DeviceList must be released with this function when it is no longer used.

SYNTAX

```
PUBLIC void
NovaspreadDeviceListRelease(
  NovaspreadTDeviceList This );
```

PARAMETERS

This
The DeviceList.

2.3.3. NovaspreadDeviceListGetLength

Gets the number of Devices stored in this list. If the list is empty 0 is returned.

SYNTAX

```
PUBLIC NovaspreadTUInt32
NovaspreadDeviceListGetLength(
  NovaspreadTDeviceList This );
```

PARAMETERS

This
The DeviceList.

RETURN VALUE

The number of Devices stored in this list.

2.3.4. NovaspreadDeviceListGetDevice

Gets the Device at the given index from the DeviceList. The returned Device must be released by calling NovaspreadDeviceRelease(), when it is no longer needed. The first Device in the list has the index 0.

SYNTAX

```
PUBLIC NovaspreadTDevice
NovaspreadDeviceListGetDevice(
  NovaspreadTDeviceList This,
  NovaspreadTUInt32     aIndex );
```

PARAMETERS

This
The DeviceList.

aIndex
The index of the Device to be returned.

RETURN VALUE

A Device. NOVASPREAD_NULL if the given index is invalid.

SEE ALSO

NovaspreadTDevice
NovaspreadDeviceListGetLength()

2.4. NovaspreadDevice

A Device represents a SAT>IP server which was found via UPnP in the local network. The Device provides information that are retrieved from the UPnP device description provided by the SAT>IP server.

To get access to the properties of a Device, use the function NovaspreadDeviceListGetDevice().

2.4.1. NovaspreadTDevice

This type defines the **NovaspreadTDevice** handle.

SYNTAX

```
typedef struct NovaspreadTDeviceStruct * NovaspreadTDevice;
```

2.4.2. NovaspreadTDeviceType

Defines various types of servers.

SYNTAX

```
typedef enum
{
  NOVASPREAD_DEVICE_TYPE_MULTISCREEN_SERVER,
  NOVASPREAD_DEVICE_TYPE_SAT_IP_SERVER,
  NOVASPREAD_DEVICE_TYPE_FTV_RC_SERVER

} NovaspreadTDeviceType;
```

COMPONENTS

NOVASPREAD_DEVICE_TYPE_MULTISCREEN_SERVER
The server is a full-fledged Multiscreen-Server (including both a SAT>IP server and a FreeTV RC server).

NOVASPREAD_DEVICE_TYPE_SAT_IP_SERVER
The server is a stand-alone SAT>IP server (without Multiscreen).

NOVASPREAD_DEVICE_TYPE_FTV_RC_SERVER
The server is a stand-alone FreeTV RC server (without Multiscreen).

2.4.3. NovaspreadDeviceRelease

Release this Device. This function must be called for each Device retrieved with the functions NovaspreadDeviceListGetDevice() when the Device is no longer used.

SYNTAX

```
PUBLIC void
NovaspreadDeviceRelease(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

SEE ALSO

NovaspreadDeviceListGetDevice()

2.4.4. NovaspreadDeviceEquals

Returns whether two devices are equal. Do not use the == operator to compare NovaspreadDevices.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadDeviceEquals(
  NovaspreadTDevice aDevice1,
  NovaspreadTDevice aDevice2 );
```

PARAMETERS

aDevice1
The first device.

aDevice2
The second device.

RETURN VALUE

NOVASPREAD_TRUE
if the devices are equal.

NOVASPREAD_FALSE
otherwise

2.4.5. NovaspreadDeviceGetType

Gets the type of the device.

SYNTAX

```
PUBLIC NovaspreadTDeviceType
NovaspreadDeviceGetType(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The device type.

SEE ALSO

NovaspreadTDeviceType

2.4.6. NovaspreadDeviceGetIpAddress

Gets the IP address of this Device.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetIpAddress(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The IP address as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.7. NovaspreadDeviceGetFriendlyName

Gets the friendly name of this Device.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetFriendlyName(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The friendly name as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.8. NovaspreadDeviceGetManufacturer

Gets the manufacturer information of this Device.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetManufacturer(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The manufacturer as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.9. NovaspreadDeviceGetManufacturerUrl

Gets the manufacturer's URL.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetManufacturerUrl(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The manufacturer's URL as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.10. NovaspreadDeviceGetModelDescription

Gets the model description.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetModelDescription(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The model description as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.11. NovaspreadDeviceGetModelNumber

Gets the model number.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetModelNumber(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device

RETURN VALUE

The model number as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.12. NovaspreadDeviceGetModelUrl

Gets the model's URL.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetModelUrl(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The model URL as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.13. NovaspreadDeviceGetSerialNumber

Gets the serial number.

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetSerialNumber(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The serial number as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.14. NovaspreadDeviceGetUniqueDeviceName

Gets the unique device name (UDN).

SYNTAX

```
PUBLIC const char *
NovaspreadDeviceGetUniqueDeviceName(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The unique device name as UTF-8 encoded string. NOVASPREAD_NULL is returned if not defined or if an error occurred.

2.4.15. NovaspreadDeviceGetIconList

Gets the IconList defined for this Device. The returned IconList must be released with the function NovaspreadIconListRelease() if it is no longer used.

SYNTAX

```
PUBLIC NovaspreadTIconList
NovaspreadDeviceGetIconList(
  NovaspreadTDevice This );
```

PARAMETERS

This
The Device.

RETURN VALUE

The IconList. NOVASPREAD_NULL if an error occurred.

SEE ALSO

NovaspreadTIconList
NovaspreadIconListRelease()

2.5. NovaspreadIconList

For each Device a list of Icons can be retrieved. An Icon is used for a graphical user interface. The IconList class represents a list of Icons.

2.5.1. NovaspreadTIconList

This type defines the **NovaspreadTIconList** handle.

SYNTAX

```
typedef struct NovaspreadTIconListStruct * NovaspreadTIconList;
```

2.5.2. NovaspreadIconListRelease

Releases this IconList. After calling this function the list shall no longer be accessed. Each time an IconList is retrieved with the function NovaspreadDeviceGetIconList() this function must be called to release the IconList.

SYNTAX

```
PUBLIC void
NovaspreadIconListRelease(
  NovaspreadTIconList This );
```

PARAMETERS

This
The IconList

SEE ALSO

NovaspreadDeviceGetIconList()

2.5.3. NovaspreadIconListGetLength

Gets the number of Icons stored in this list. If the IconList is empty this function returns 0.

SYNTAX

```
PUBLIC NovaspreadTUInt32
NovaspreadIconListGetLength(
  NovaspreadTIconList This );
```

PARAMETERS

This
The IconList

RETURN VALUE

The number of Icons stored in this list. 0 if the IconList is empty.

SEE ALSO

NovaspreadIconListGetIcon()

2.5.4. NovaspreadIconListGetIcon

Gets the Icon at the given index from the IconList. The returned Icon must be released by calling NovaspreadIconRelease(), when it is no longer needed. The index starts with 0 for the first icon in the IconList.

SYNTAX

```
PUBLIC NovaspreadTIcon
NovaspreadIconListGetIcon(
  NovaspreadTIconList This,
  NovaspreadTUInt32   aIndex );
```

PARAMETERS

This
The IconList

aIndex
The index of the Icon to be returned.

RETURN VALUE

The Icon at the given index position. NOVASPREAD_NULL if the given index is invalid.

2.6. NovaspreadIcon

Defines an Icon used for a Device. Properties of an Icon are the width, height and color depth and the URL from where the icon image file can be loaded.

2.6.1. NovaspreadTIcon

This type defines the **NovaspreadTIcon** handle.

SYNTAX

```
typedef struct NovaspreadTIconStruct * NovaspreadTIcon;
```

2.6.2. NovaspreadIconRelease

Releases the Icon. After calling this function, the Icon shall no longer be accessed. Each Icon that is retrieved with the function NovaspreadIconListGetIcon() must be released with this function.

SYNTAX

```
PUBLIC void
NovaspreadIconRelease(
  NovaspreadTIcon This );
```

PARAMETERS

This
The Icon.

SEE ALSO

NovaspreadIconListGetIcon()

2.6.3. NovaspreadIconGetMimeType

Gets the MIME type of the Icon. The MIME type is a null-terminated ASCII string that defines the format of the Icon, e.g. "image/png", "image/jpeg".

SYNTAX

```
PUBLIC const char *
NovaspreadIconGetMimeType(
  NovaspreadTIcon This );
```

PARAMETERS

This
The Icon.

RETURN VALUE

The MIME type as null-terminated ASCII string.

2.6.4. NovaspreadIconGetWidth

Gets the width of the Icon in pixels.

SYNTAX

```
PUBLIC NovaspreadTUInt16
NovaspreadIconGetWidth(
  NovaspreadTIcon This );
```

PARAMETERS

This
The Icon.

RETURN VALUE

The width of the Icon.

SEE ALSO

NovaspreadIconGetHeight()

2.6.5. NovaspreadIconGetHeight

Gets the height of the Icon in pixels.

SYNTAX

```
PUBLIC NovaspreadTUInt16
NovaspreadIconGetHeight(
  NovaspreadTIcon This );
```

PARAMETERS

This
The Icon.

RETURN VALUE

The height of the Icon.

SEE ALSO

`NovaspreadIconGetWidth()`

2.6.6. NovaspreadIconGetDepth

Gets the color depth of the Icon. The returned value indicates the number of colors of the Icon.

SYNTAX

```
PUBLIC NovaspreadTUInt32
NovaspreadIconGetDepth(
  NovaspreadTIcon This );
```

PARAMETERS

This
The Icon.

RETURN VALUE

The color depth of the Icon.

2.6.7. NovaspreadIconGetUrl

Gets the URL of the Icon. From this URL the Icon can be downloaded.

SYNTAX

```
PUBLIC const char *
NovaspreadIconGetUrl(
  NovaspreadTIcon This );
```

PARAMETERS

This
The Icon.

RETURN VALUE

The URL as UTF-8 encoded string.

2.7. NovaspreadDvbId

The NovaspreadDvbId is used to identify a DVB service.

2.7.1. NovaspreadTDvbId

This data structure defines the DVB triplet of a DVB service. The DVB triplet is used to identify a DVB service.

SYNTAX

```
typedef struct
{
  NovaspreadTUInt16 OriginalNetworkId;
```

```
  NovaspreadTUInt16 TransportStreamId;
  NovaspreadTUInt16 ServiceId;

} NovaspreadTDvbId;
```

COMPONENTS

OriginalNetworkId
The DVB original network ID.

TransportStreamId
The DVB transport stream ID.

ServiceId
The DVB service ID.

2.8. NovaspreadTunerParameters

TunerParameters define the types for tuning parameters.

2.8.1. NovaspreadTTunerType

NovaspreadTunerType defines the different types of tuners. Which tuners are actually supported depends on the target platform. Currently only DVB-S/S2 tuners are supported.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_TYPE_DVB_S

} NovaspreadTTunerType;
```

COMPONENTS

NOVASPREAD_TUNER_TYPE_DVB_S
A DVB-S tuner receives data from a satellite

2.8.2. NovaspreadTTunerPolarization

This enumeration type defines the supported polarization types for DVB-S tuners.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_POLARIZATION_UNKNOWN,
  NOVASPREAD_TUNER_POLARIZATION_H,
  NOVASPREAD_TUNER_POLARIZATION_V,
  NOVASPREAD_TUNER_POLARIZATION_CIRCULAR_LEFT,
  NOVASPREAD_TUNER_POLARIZATION_CIRCULAR_RIGHT

} NovaspreadTTunerPolarization;
```

COMPONENTS

NOVASPREAD_TUNER_POLARIZATION_UNKNOWN
For internal use only.

NOVASPREAD_TUNER_POLARIZATION_H
Horizontal polarization

NOVASPREAD_TUNER_POLARIZATION_V
Vertical polarization

NOVASPREAD_TUNER_POLARIZATION_CIRCULAR_LEFT

Circular left polarization

NOVASPREAD_TUNER_POLARIZATION_CIRCULAR_RIGHT
Circular right polarization

2.8.3. NovaspreadTTunerRollOff

This enumeration type defines the values for the roll-off factor used in DVB-S2. For DVB-S always NOVASPREAD_TUNER_ROLL_OFF_0_35 is used.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_ROLL_OFF_UNKNOWN,
  NOVASPREAD_TUNER_ROLL_OFF_0_20,
  NOVASPREAD_TUNER_ROLL_OFF_0_25,
  NOVASPREAD_TUNER_ROLL_OFF_0_35

} NovaspreadTTunerRollOff;
```

COMPONENTS

NOVASPREAD_TUNER_ROLL_OFF_UNKNOWN
The roll-off factor is unknown.

NOVASPREAD_TUNER_ROLL_OFF_0_20
Roll-off is 0.20.

NOVASPREAD_TUNER_ROLL_OFF_0_25
Roll-off is 0.25.

NOVASPREAD_TUNER_ROLL_OFF_0_35
Roll-off is 0.35.

2.8.4. NovaspreadTTunerPilotTones

This enumeration type defines the two values for the pilot tone (on or off) for DVB-S2 signals.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_PILOT_TONES_UNKNOWN,
  NOVASPREAD_TUNER_PILOT_TONES_ON,
  NOVASPREAD_TUNER_PILOT_TONES_OFF

} NovaspreadTTunerPilotTones;
```

COMPONENTS

NOVASPREAD_TUNER_PILOT_TONES_UNKNOWN
It is unkown if pilot tones are enabled in the transmission or not.

NOVASPREAD_TUNER_PILOT_TONES_ON
Pilot tones are available in the transmission.

NOVASPREAD_TUNER_PILOT_TONES_OFF
Pilot tones are not used in the transmission.

2.8.5. NovaspreadTTunerModulationSystem

This type represents the supported modulation systems which are necessary for DVB-S2 tuners.

SYNTAX

```
typedef enum
```

```
{
  NOVASPREAD_TUNER_MODULATION_SYSTEM_UNKNOWN,
  NOVASPREAD_TUNER_MODULATION_SYSTEM_DVB_S,
  NOVASPREAD_TUNER_MODULATION_SYSTEM_DVB_S2

} NovaspreadTTunerModulationSystem;
```

COMPONENTS

NOVASPREAD_TUNER_MODULATION_SYSTEM_UNKNOWN
For internal use only.

NOVASPREAD_TUNER_MODULATION_SYSTEM_DVB_S
The modulation system 'DVB-S'.

NOVASPREAD_TUNER_MODULATION_SYSTEM_DVB_S2
The modulation system 'DVB-S2'.

2.8.6. NovaspreadTTunerModulation

This enumeration type defines the supported modulation types for DVB tuners. For each tuner type (e.g. DVB-S) only a selection of the listed modulation types can be used.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_MODULATION_UNKNOWN,
  NOVASPREAD_TUNER_MODULATION_AUTO,
  NOVASPREAD_TUNER_MODULATION_QPSK,
  NOVASPREAD_TUNER_MODULATION_8PSK,
  NOVASPREAD_TUNER_MODULATION_QAM_16,
  NOVASPREAD_TUNER_MODULATION_QAM_32,
  NOVASPREAD_TUNER_MODULATION_QAM_64,
  NOVASPREAD_TUNER_MODULATION_QAM_128,
  NOVASPREAD_TUNER_MODULATION_QAM_256,
  NOVASPREAD_TUNER_MODULATION_LAST

} NovaspreadTTunerModulation;
```

COMPONENTS

NOVASPREAD_TUNER_MODULATION_UNKNOWN
The modulation is unknown. This value shall not be used for setting the tuner parameters.

NOVASPREAD_TUNER_MODULATION_AUTO
If this modulation is used, the tuner tries to find out the correct modulation automatically.

NOVASPREAD_TUNER_MODULATION_QPSK
Represents a QPSK modulation.

NOVASPREAD_TUNER_MODULATION_8PSK
Represents a 8PSK modulation.

NOVASPREAD_TUNER_MODULATION_QAM_16
Represents a 16-QAM modulation.

NOVASPREAD_TUNER_MODULATION_QAM_32
Represents a 32-QAM modulation.

NOVASPREAD_TUNER_MODULATION_QAM_64
Represents a 64-QAM modulation.

NOVASPREAD_TUNER_MODULATION_QAM_128
Represents a 128-QAM modulation.

NOVASPREAD_TUNER_MODULATION_QAM_256
Represents a 256-QAM modulation.

NOVASPREAD_TUNER_MODULATION_LAST
The last modulation parameter. For internal use only.

2.8.7. NovaspreadTTunerCodeRate

This type defines the code rates.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_CODE_RATE_UNKNOWN,
  NOVASPREAD_TUNER_CODE_RATE_AUTO,
  NOVASPREAD_TUNER_CODE_RATE_1_2,
  NOVASPREAD_TUNER_CODE_RATE_1_3,
  NOVASPREAD_TUNER_CODE_RATE_1_4,
  NOVASPREAD_TUNER_CODE_RATE_2_3,
  NOVASPREAD_TUNER_CODE_RATE_2_5,
  NOVASPREAD_TUNER_CODE_RATE_3_4,
  NOVASPREAD_TUNER_CODE_RATE_3_5,
  NOVASPREAD_TUNER_CODE_RATE_4_5,
  NOVASPREAD_TUNER_CODE_RATE_5_6,
  NOVASPREAD_TUNER_CODE_RATE_6_7,
  NOVASPREAD_TUNER_CODE_RATE_7_8,
  NOVASPREAD_TUNER_CODE_RATE_8_9,
  NOVASPREAD_TUNER_CODE_RATE_9_10,
  NOVASPREAD_TUNER_CODE_RATE_LAST

} NovaspreadTTunerCodeRate;
```

COMPONENTS

NOVASPREAD_TUNER_CODE_RATE_UNKNOWN
The code rate is unknown. This value shall not be used for setting the tuner parameters.

NOVASPREAD_TUNER_CODE_RATE_AUTO
If this code rate is used, the tuner tries to find out the correct code rate automatically.

NOVASPREAD_TUNER_CODE_RATE_1_2
Represents a code rate of 1/2

NOVASPREAD_TUNER_CODE_RATE_1_3
Represents a code rate of 1/3

NOVASPREAD_TUNER_CODE_RATE_1_4
Represents a code rate of 1/4

NOVASPREAD_TUNER_CODE_RATE_2_3
Represents a code rate of 2/3

NOVASPREAD_TUNER_CODE_RATE_2_5
Represents a code rate of 2/5

NOVASPREAD_TUNER_CODE_RATE_3_4
Represents a code rate of 3/4

NOVASPREAD_TUNER_CODE_RATE_3_5
Represents a code rate of 3/5

NOVASPREAD_TUNER_CODE_RATE_4_5
Represents a code rate of 4/5

NOVASPREAD_TUNER_CODE_RATE_5_6
Represents a code rate of 5/6

NOVASPREAD_TUNER_CODE_RATE_6_7
Represents a code rate of 6/7

NOVASPREAD_TUNER_CODE_RATE_7_8
Represents a code rate of 7/8

NOVASPREAD_TUNER_CODE_RATE_8_9
Represents a code rate of 8/9

NOVASPREAD_TUNER_CODE_RATE_9_10
Represents a code rate of 9/10

NOVASPREAD_TUNER_CODE_RATE_LAST
The last code rate parameter. For internal use only.

SEE ALSO

NovaspreadTTunerParameters

2.8.8. NovaspreadTTunerParamDvbS

This structure contains the tuning parameters of a DVB-S/S2 tuner.

SYNTAX

```
typedef struct
{
  NovaspreadTUInt16                FrontendId;
  NovaspreadTUInt8                 SignalSource;
  NovaspreadTInt16                 OrbitalPosition;
  NovaspreadTUInt32                Frequency;
  NovaspreadTTunerPolarization     Polarization;
  NovaspreadTTunerRollOff          RollOff;
  NovaspreadTTunerModulationSystem ModulationSystem;
  NovaspreadTTunerModulation       Modulation;
  NovaspreadTTunerPilotTones       PilotTones;
  NovaspreadTUInt32                SymbolRate;
  NovaspreadTTunerCodeRate         CodeRate;

} NovaspreadTTunerParamDvbS;
```

COMPONENTS

FrontendId
The Frontend identifier. 0 means not used.

SignalSource
This SignalSource is passed to the SAT>IP server. Set to 0 if not used.

OrbitalPosition
In 1/10 degrees. e.g. Astra 19.2E = 192

Frequency
The transponder frequency in KHz to tune to.

Polarization
The polarization of the transponder.

RollOff
The RollOff parameter. For DVB-S set to 0_35.

ModulationSystem
The used modulation system. This is only necessary for DVB-S2 tuner. DVB-S tuner will ignore it.

Modulation
The modulation of the transponder.

PilotTones
The pilot tones. For DVB-S set to OFF.

SymbolRate
Kilo-symbols per second.

CodeRate
The code rate of the transponder.

2.8.9. NovaspreadTTunerParamValue

This union contains the parameters for the different types of tuners. Currently only DVB-S/S2 is supported.

SYNTAX

```
typedef union
{
  NovaspreadTTunerParamDvbS DvbS;

} NovaspreadTTunerParamValue;
```

COMPONENTS

DvbS
The tuning parameters specific for DVB-S/S2 reception.

2.8.10. NovaspreadTTunerParameters

This data structure defines the tuning parameters to be set at a tuner. It defines the type of tuner (currently only DVB-S) for which the parameters are to be set. Depending on this type the Value is interpreted.

SYNTAX

```
typedef struct
{
  NovaspreadTTunerType       Type;
  NovaspreadTTunerParamValue Value;

} NovaspreadTTunerParameters;
```

COMPONENTS

Type
The type of the tuner.

Value
Structure containing the tuning parameters specific for a type of tuner.

2.8.11. NovaspreadTTunerSignalInfo

This type defines the SignalInfo of a tuner. The SignalInfo contains the Level and the Quality of the signal received by the Tuner. For a specification of the SignalInfo see SAT>IP Protocol Specification V1.2.2.

SYNTAX

```
typedef struct
{
  NovaspreadTUInt8 Level;
  NovaspreadTUInt8 Quality;

} NovaspreadTTunerSignalInfo;
```

COMPONENTS

Level
Numerical value between 0 and 255. An incoming L-band satellite signal of -25dBm corresponds to 224, -65dBm corresponds to 32 and no signal corresponds to 0.

Quality
Numerical value between 0 and 15. Lower values indicate to higher error rates. The value 15 indicates a BER lower than 2.0E-4 after Viterbi for DVB-S, a BER lower than 10.0E-7 for DVB-S2.

SEE ALSO

```
NovaspreadTunerGetSignalInfo()
NovaspreadSatIpTunerGetSignalInfo()
```

2.9. NovaspreadTranscoding

NovaspreadTranscoding defines all types which are used for transcoding. Transcoding is controlled with the Tuner by the function NovaspreadTunerSetTranscoding(). The NovaspreadTTranscoding structure defined in this section contains all necessary parameters.

2.9.1. NovaspreadTVideoCodec

This enumeration type defines all available video codecs used for transcoding. The list of supported video codecs depends on the platform.

SYNTAX

```
typedef enum
{
  NOVASPREAD_VIDEO_CODEC_UNKNOWN,
  NOVASPREAD_VIDEO_CODEC_MPEG_2,
  NOVASPREAD_VIDEO_CODEC_AVC,
  NOVASPREAD_VIDEO_CODEC_HEVC,
  NOVASPREAD_VIDEO_CODEC_LAST

} NovaspreadTVideoCodec;
```

COMPONENTS

NOVASPREAD_VIDEO_CODEC_UNKNOWN
The codec is unknown. For internal use only.

NOVASPREAD_VIDEO_CODEC_MPEG_2
MPEG-2 video

NOVASPREAD_VIDEO_CODEC_AVC
H.264 video (MPEG-4 AVC). The maximum profile shall be HP@L4.

NOVASPREAD_VIDEO_CODEC_HEVC
High Efficiency Video Coding (HEVC). The maximum profile shall be MP@L4.1 Main Tier.

NOVASPREAD_VIDEO_CODEC_LAST
For internal use only.

2.9.2. NovaspreadTVideoResolution

This enumeration type defines the possible video resolutions that can be used by the Transcoder for the video output resolution. The supported video resolutions depend on the platform.

SYNTAX

```
typedef enum
{
  NOVASPREAD_VIDEO_RESOLUTION_UNKNOWN,
  NOVASPREAD_VIDEO_RESOLUTION_144P,
  NOVASPREAD_VIDEO_RESOLUTION_288P,
  NOVASPREAD_VIDEO_RESOLUTION_576P,
  NOVASPREAD_VIDEO_RESOLUTION_576I,
  NOVASPREAD_VIDEO_RESOLUTION_720P,
  NOVASPREAD_VIDEO_RESOLUTION_1080P,
  NOVASPREAD_VIDEO_RESOLUTION_1080I,
  NOVASPREAD_VIDEO_RESOLUTION_LAST

} NovaspreadTVideoResolution;
```

COMPONENTS

NOVASPREAD_VIDEO_RESOLUTION_UNKNOWN
The resolution is unknown. For internal use only.

NOVASPREAD_VIDEO_RESOLUTION_144P
176x144 progressive

NOVASPREAD_VIDEO_RESOLUTION_288P

352x288 progressive

NOVASPREAD_VIDEO_RESOLUTION_576P
720x576 progressive

NOVASPREAD_VIDEO_RESOLUTION_576I
720x576 interlaced

NOVASPREAD_VIDEO_RESOLUTION_720P
1280x720 progressive

NOVASPREAD_VIDEO_RESOLUTION_1080P
1920x1080 progressive

NOVASPREAD_VIDEO_RESOLUTION_1080I
1920x1080 interlaced

NOVASPREAD_VIDEO_RESOLUTION_LAST
For internal use only

2.9.3. NovaspreadTAudioCodec

This enumeration type defines the different audio codecs used for transcoding. The list of supported audio codecs depends on the platform.

SYNTAX

```
typedef enum
{
  NOVASPREAD_AUDIO_CODEC_UNKNOWN,
  NOVASPREAD_AUDIO_CODEC_MP2,
  NOVASPREAD_AUDIO_CODEC_AC3,
  NOVASPREAD_AUDIO_CODEC_LC_AAC,
  NOVASPREAD_AUDIO_CODEC_HE_AAC,
  NOVASPREAD_AUDIO_CODEC_LAST

} NovaspreadTAudioCodec;
```

COMPONENTS

NOVASPREAD_AUDIO_CODEC_UNKNOWN
The codec is unknown. For internal use only.

NOVASPREAD_AUDIO_CODEC_MP2
MPEG-1 Audio Layer II

NOVASPREAD_AUDIO_CODEC_AC3
Dolby Digital

NOVASPREAD_AUDIO_CODEC_LC_AAC
Low-Complexity Advanced Audio Coding (LC-AAC)

NOVASPREAD_AUDIO_CODEC_HE_AAC
High-Efficiency Advanced Audio Coding (HE-AAC). The following profile shall be used: HE-AAC v1.

NOVASPREAD_AUDIO_CODEC_LAST
For internal use only

2.9.4. NovaspreadTTranscoding

The Transcoding type defines the output properties of the transcoded stream.

SYNTAX

```
typedef struct
{
  NovaspreadTVideoCodec      VideoCodec;
  NovaspreadTUInt32          VideoBitrate;
  NovaspreadTVideoResolution VideoResolution;
```

```
  NovaspreadTAudioCodec      AudioCodec;
  NovaspreadTUInt32          AudioBitrate;

} NovaspreadTTranscoding;
```

COMPONENTS

VideoCodec
The VideoCodec of the transcoded video stream. See NovaspreadTVideoCodec for a list of possible values.

VideoBitrate
The maximum bit rate of the transcoded video stream in kbits/sec.

VideoResolution
The resolution of the transcoded video stream.

AudioCodec
The AudioCodec of the transcoded audio stream. See NovaspreadTAudioCodec for a list of possible values.

AudioBitrate
The bit rate of the transcoded audio stream in kbits/sec.

SEE ALSO

NovaspreadTAudioCodec
NovaspreadTVideoCodec
NovaspreadTVideoResolution

2.10. NovaspreadSatIpTuner

A SatIpTuner can be used to receive transport stream data from a SAT>IP server which is available in the local network.

To create SatIpTuner call the function NovaspreadSeverCreateSatIpTuner(). Only SatIpTuners from the selected SAT>IP server device are used.

After creation a SatIpTuner is not connected to a SAT>IP server. To connect to a SAT>IP server, tuner parameters must be set at the SatIpTuner with the function NovaspreadSatIpTunerSetParameters() and then NovaspreadSatIpTunerConnect() must be called. By calling NovaspreadSatIpTunerSetPids(), the pids that shall be received from the SAT>IP server are defined.

As soon as the SatIpTuner has changed its ConnectionStatus to CONNECTED, NovaspreadSatIpTunerReadData() will provide transport stream data.

2.10.1. NovaspreadTSatIpTuner

This type defines the **NovaspreadTSatIpTuner** handle.

SYNTAX

```
typedef struct NovaspreadTSatIpTunerStruct * NovaspreadTSatIpTuner;
```

2.10.2. NovaspreadTSatIpTunerState

A SatIpTuner is in one of the following States.

SYNTAX

```
typedef enum
{
  NOVASPREAD_SAT_IP_TUNER_STATE_DISCONNECTED,
  NOVASPREAD_SAT_IP_TUNER_STATE_CONNECTED,
  NOVASPREAD_SAT_IP_TUNER_STATE_STREAMING,
```

```
  NOVASPREAD_SAT_IP_TUNER_STATE_ERROR

} NovaspreadTSatIpTunerState;
```

COMPONENTS

NOVASPREAD_SAT_IP_TUNER_STATE_DISCONNECTED
The SatIpTuner is not connected to a SAT>IP server.

NOVASPREAD_SAT_IP_TUNER_STATE_CONNECTED
The SatIpTuner is connected to a SAT>IP server, but does not yet receive transport stream data.

NOVASPREAD_SAT_IP_TUNER_STATE_STREAMING
The SatIpTuner receives transport stream data from the SAT>IP server.

NOVASPREAD_SAT_IP_TUNER_STATE_ERROR
An error occurred. Disconnect the tuner to leave the error state.

2.10.3. NovaspreadTSatIpTunerStateChangeListener

A function of this type can be set at a SatIpTuner. It is called every time the State of the SatIpTuner changes.

SYNTAX

```
typedef void
(* NovaspreadTSatIpTunerStateChangeListener ) (
  void *                    aContext,
  NovaspreadTSatIpTuner     aSatIpTuner,
  NovaspreadTSatIpTunerState aNewState );
```

PARAMETERS

aContext
This context is passed unchanged from NovaspreadSatIpTunerSetStateChangeListener().

aSatIpTuner
The state of this tuner has changed.

aNewState
The new state.

SEE ALSO

```
NovaspreadSatIpTunerSetStateChangeListener()
```

2.10.4. NovaspreadTSatIpTunerDataAvailableListener

A function of this type can be set at the SatIpTuner. When NovaspreadSatIpTunerReadData() returns 0, because no data is available, the registered DataAvailableListener will be called as soon as data is available again.

SYNTAX

```
typedef void
(* NovaspreadTSatIpTunerDataAvailableListener ) (
  void * aContext );
```

PARAMETERS

aContext
This context is passed unchanged from the NovaspreadSatIpTunerSetDataAvailableListener() function.

SEE ALSO

```
NovaspreadSatIpTunerSetDataAvailableListener()
NovaspreadSatIpTunerReadData()
```

2.10.5. NovaspreadSatIpTunerDestroy

Destroys the given SatIpTuner. The SatIpTuner may not be accessed after calling this function.

SYNTAX

```
PUBLIC void
NovaspreadSatIpTunerDestroy(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

2.10.6. NovaspreadSatIpTunerSetParameters

Sets the tuning parameters of this SatIpTuner. Tuning parameters define the transponder from where the transport stream is to be received.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerSetParameters(
  NovaspreadTSatIpTuner        This,
  NovaspreadTTunerParameters * aParameters );
```

PARAMETERS

This
The SatIpTuner.

aParameters
The TunerParametesr. See data type NovaspreadTTunerParameters for a description of all tuning parameters.

RETURN VALUE

NOVASPREAD_TRUE
if the parameters were set successfully.

NOVASPREAD_FALSE
if an error occurred.

SEE ALSO

NovaspreadTTunerParameters
NovaspreadSatIpTunerGetParameters()

2.10.7. NovaspreadSatIpTunerGetParameters

Gets the currently set TunerParameters.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerGetParameters(
  NovaspreadTSatIpTuner        This,
  NovaspreadTTunerParameters * aParameter );
```

PARAMETERS

This
The SatIpTuner.

aParameter
OUT: Pointer to variable of type NovaspreadTTunerParameters, where the function returns the currently set TunerParameters.

RETURN VALUE

NOVASPREAD_TRUE
if the TunerParameters are returned successfully.

NOVASPREAD_FALSE
if an error occurred. In this case the variable aParameters points to is not unchanged.

SEE ALSO

NovaspreadTTunerParameters
NovaspreadSatIpTunerSetParameters()

2.10.8. NovaspreadSatIpTunerConnect

This function is called to establish a connection of the SatIpTuner with a SAT>IP server in the network. During this call the SatIpTuner changes its ConnectionStatus to CONNECTING.

As soon as the connection is established successfully, the ConnectionStatus is changed to CONNECTED. From this point in time received transport stream packets can be retrieved with the function NovaspreadSatIpTunerReadData().

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerConnect(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

RETURN VALUE

NOVASPREAD_TRUE
if the tuner started connecting successfully.

NOVASPREAD_FALSE
otherwise.

SEE ALSO

NovaspreadTSatIpTunerState
NovaspreadSatIpTunerDisconnect()

2.10.9. NovaspreadSatIpTunerDisconnect

Disconnects a SatIpTuner from a SAT>IP server.

SYNTAX

```
PUBLIC void
NovaspreadSatIpTunerDisconnect(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

SEE ALSO

NovaspreadTSatIpTunerState
NovaspreadSatIpTunerConnect()

2.10.10. NovaspreadSatIpTunerGetState

Gets the ConnectionStatus of a SatIpTuner.

SYNTAX

```
PUBLIC NovaspreadTSatIpTunerState
NovaspreadSatIpTunerGetState(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

RETURN VALUE

The current ConnectionStatus of the SatIpTuner.

SEE ALSO

NovaspreadTSatIpTunerState
NovaspreadSatIpTunerConnect()

2.10.11. NovaspreadSatIpTunerSetStateChangeListener

This function sets a ConnectionStatusChangeListener at a SatIpTuner.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerSetStateChangeListener(
  NovaspreadTSatIpTuner                  This,
  NovaspreadTSatIpTunerStateChangeListener aListener,
  void *                                 aContext );
```

PARAMETERS

This
The SatIpTuner

aListener
The listener to be set. PASS NOVASPREAD_NULL to unset the listener.

aContext
This context is passed unchanged to the listener.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadTSatIpTunerStateChangeListener

2.10.12. NovaspreadSatIpTunerStart

When the SatIpTuner is connected, this function sends the PLAY command to the SAT>IP server. Now data can be retrieved by calling NovaspreadSatIpTunerReadData().

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerStart(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadSatIpTunerConnect()
NovaspreadSatIpTunerStop()

2.10.13. NovaspreadSatIpTunerStop

This function stops the SatIpTuner. No more data can be read.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerStop(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadSatIpTunerStart()

2.10.14. NovaspreadSatIpTunerSetPids

Sets the pids which shall be available in the stream received by this Tuner. This function overwrites the pids previously enabled for the SatIpTuner. To reset all pids, pass aPids=NOVASPREAD_NULL and aNoOfPids=0.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerSetPids(
  NovaspreadTSatIpTuner This,
  NovaspreadTUInt16 *   aPids,
  NovaspreadTUInt32     aNoOfPids );
```

PARAMETERS

This
The SatIpTuner.

aPids
The array of pids.

aNoOfPids
The number of pids in the array.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

`NovaspreadSatIpTunerGetPids()`

2.10.15. NovaspreadSatIpTunerSetAllPids

All pids of a transport stream shall be streamed. This function overwrites the pids previously enabled for the SatIpTuner. To reset all pids, pass aPids=NOVASPREAD_NULL and aNoOfPids=0 to NovaspreadSatIpTunerSetPids().

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerSetAllPids(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

2.10.16. NovaspreadSatIpTunerGetPids

Gets the pids that are currently enabled for streaming.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerGetPids(
  NovaspreadTSatIpTuner This,
  NovaspreadTUInt16 *   aPids,
  NovaspreadTUInt32     aMaxNoOfPids,
  NovaspreadTUInt32 *   aNoOfPids );
```

PARAMETERS

This
The SatIpTuner.

aPids
OUT: Pointer to an array of UInt16 where the function stores the currently enabled pids

aMaxNoOfPids
The maximal number of pids that can be copied into the aPids array.

aNoOfPids
OUT: The number of pids that are copied into the aPids array.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadSatIpTunerSetPids()

2.10.17. NovaspreadSatIpTunerAddPids

Adds pids, which shall additionally be received by this Tuner.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerAddPids(
  NovaspreadTSatIpTuner This,
  NovaspreadTUInt16 *   aPids,
  NovaspreadTUInt32     aNoOfPids );
```

PARAMETERS

This
The SatIpTuner.

aPids
The array of pids that shall additionally be received.

aNoOfPids
The number of pids in the array.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.10.18. NovaspreadSatIpTunerRemovePids

Removes pids, which should no longer be received by this Tuner.

SYNTAX

```
PUBLIC void
NovaspreadSatIpTunerRemovePids(
  NovaspreadTSatIpTuner This,
  NovaspreadTUInt16 *   aPids,
  NovaspreadTUInt32     aNoOfPids );
```

PARAMETERS

This
The SatIpTuner.

aPids
The array of pids that should no longer be streamed.

aNoOfPids
The number of pids in the array.

2.10.19. NovaspreadSatIpTunerIsLocked

Returns the lock status of the tuner. A Tuner is locked if a signal is detected for the set TunerParameter and the demodulator is able to decode the signal. A Tuner receives data only if it is locked.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerIsLocked(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

RETURN VALUE

NOVASPREAD_TRUE
if the SatIpTuner is locked.

NOVASPREAD_FALSE
otherwise.

2.10.20. NovaspreadSatIpTunerGetSignalInfo

Gets the current SignalInfo of the SatIpTuner. See data type NovaspreadTTunerSignalInfo for a description of the returned data.

SYNTAX

```
PUBLIC NovaspreadTTunerSignalInfo
NovaspreadSatIpTunerGetSignalInfo(
  NovaspreadTSatIpTuner This );
```

PARAMETERS

This
The SatIpTuner.

RETURN VALUE

The current SignalInfo.

SEE ALSO

NovaspreadTTunerSignalInfo

2.10.21. NovaspreadSatIpTunerSetDataAvailableListener

Sets a DataAvailableListener. Only one DataAvailableListener can be set at a SatIpTuner.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadSatIpTunerSetDataAvailableListener(
  NovaspreadTSatIpTuner                    This,
  NovaspreadTSatIpTunerDataAvailableListener aListener,
  void *                                   aContext );
```

PARAMETERS

This
The SatIpTuner.

aListener
The DataAvailableListener. Pass NOVASPREAD_NULL to unset the listener.

aContext
This context is passed unchanged to the listener.

RETURN VALUE

NOVASPREAD_TRUE

if successful.

NOVASPREAD_FALSE
otherwise.

SEE ALSO

```
NovaspreadTSatIpTunerDataAvailableListener
NovaspreadSatIpTunerReadData()
```

2.10.22. NovaspreadSatIpTunerReadData

As soon as the SatIpTuner is in ConnectionStatus CONNECTED, this function will write 188 bytes long transport stream packets to the buffer. This function must be called periodically to avoid a SatIpTuner internal buffer overflow.

If this function is called when the SatIpTuner is in a different ConnectionStatus, it will not write data to the buffer and return 0.

If NovaspreadSatIpTunerReadData() is called in ConnectionStatus CONNECTED and no data is available, 0 will be returned. As soon as data is available again, a previously set DataAvailableListener will be called. Do not call NovaspreadSatIpTunerReadData() in the context of the DataAvailableListener.

SYNTAX

```
PUBLIC NovaspreadTUInt32
NovaspreadSatIpTunerReadData(
  NovaspreadTSatIpTuner This,
  NovaspreadTUInt8 *    aBuffer,
  NovaspreadTUInt32     aBufferSize );
```

PARAMETERS

This
The SatIpTuner.

aBuffer
Transport stream packets are written to this buffer.

aBufferSize
The size of the buffer. Any buffer size is allowed.

RETURN VALUE

The number of bytes written to the buffer. If there are no transport stream packets available 0 is returned.

2.11. NovaspreadCaInfo

A NovaspreadCaInfo represent all information that is returned via a "GET /rc/ca" request as defined in "FreeTV Remote Control Specification v1.0" and "FREETVA-RC Profile AS-30102 HD+ Platform v1.0".

2.11.1. NovaspreadTCaInfo

This type defines the **NovaspreadTCaInfo** handle.

SYNTAX

```
typedef struct NovaspreadTCaInfoStruct * NovaspreadTCaInfo;
```

2.11.2. NovaspreadTCaInfoSmartcardStatus

This type defines various smartcard status.

SYNTAX

```
typedef enum
{
  NOVASPREAD_CA_INFO_SMARTCARD_STATUS_ACTIVATING,
  NOVASPREAD_CA_INFO_SMARTCARD_STATUS_NOT_ACTIVATED,
  NOVASPREAD_CA_INFO_SMARTCARD_STATUS_ACTIVATED,
  NOVASPREAD_CA_INFO_SMARTCARD_STATUS_TUNE,
  NOVASPREAD_CA_INFO_SMARTCARD_STATUS_EXPIRED

} NovaspreadTCaInfoSmartcardStatus;
```

COMPONENTS

NOVASPREAD_CA_INFO_SMARTCARD_STATUS_ACTIVATING
The smartcard is currently activating.

NOVASPREAD_CA_INFO_SMARTCARD_STATUS_NOT_ACTIVATED
The smartcard is not yet activated.

NOVASPREAD_CA_INFO_SMARTCARD_STATUS_ACTIVATED
The smartcard is activated.

NOVASPREAD_CA_INFO_SMARTCARD_STATUS_TUNE
Tune to a specific channel.

NOVASPREAD_CA_INFO_SMARTCARD_STATUS_EXPIRED
The smartcard is expired.

2.11.3. NovaspreadCaInfoCreate

Creates a new CaInfo.

SYNTAX

```
PUBLIC NovaspreadTCaInfo
NovaspreadCaInfoCreate(
  void );
```

RETURN VALUE

A new CaInfo if successful. NOVASPREAD_NULL otherwise.

SEE ALSO

NovaspreadCaInfoDestroy()

2.11.4. NovaspreadCaInfoDestroy

Destroys the given CaInfo.

SYNTAX

```
PUBLIC void
NovaspreadCaInfoDestroy(
  NovaspreadTCaInfo This );
```

PARAMETERS

This
This CaInfo.

SEE ALSO

NovaspreadCaInfoCreate()

2.11.5. NovaspreadCaInfoSetChipsetUid

Sets the chipset unique ID.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetChipsetUid(
  NovaspreadTCaInfo This,
  const char *      aChipsetUid );
```

PARAMETERS

This
This CaInfo.

aChipsetUid
The chipset unique ID.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.6. NovaspreadCaInfoSetChipsetType

Sets the chipset type.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetChipsetType(
  NovaspreadTCaInfo This,
  const char *      aChipsetType );
```

PARAMETERS

This
This CaInfo.

aChipsetType
The type of the chipset.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.7. NovaspreadCaInfoSetChipsetRevision

Sets the chipset revision.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetChipsetRevision(
  NovaspreadTCaInfo This,
  const char *      aChipsetRevision );
```

PARAMETERS

This
This CaInfo

aChipsetRevision
The chipset revision.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.8. NovaspreadCaInfoSetCaVendor

Sets the CAS vendor.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetCaVendor(
  NovaspreadTCaInfo This,
  const char *      aCaVendor );
```

PARAMETERS

This
This CaInfo.

aCaVendor
The CAS vendor.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.9. NovaspreadCaInfoSetCaVersion

Sets the CAS version.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetCaVersion(
  NovaspreadTCaInfo This,
  const char *      aCaVersion );
```

PARAMETERS

This
This CaInfo.

aCaVersion
The CAs version.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.10. NovaspreadCaInfoSetCaNumber

Sets the CAS serial number.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetCaNumber(
  NovaspreadTCaInfo This,
  const char *      aCaNumber );
```

PARAMETERS

This
This CaInfo.

aCaNumber
The CAS serial number.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.11. NovaspreadCaInfoSetSmartcardInserted

Sets whether a smartcard is inserted or not.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetSmartcardInserted(
  NovaspreadTCaInfo  This,
  NovaspreadTBoolean aInserted );
```

PARAMETERS

This
This CaInfo.

aInserted
NOVASPREAD_TRUE if a smartcard is inserted. NOVASPREAD_FALSE otherwise.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.12. NovaspreadCaInfoSetSmartcardSuitable

Sets if the smartcard is suitable for the Operator.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetSmartcardSuitable(
  NovaspreadTCaInfo  This,
  NovaspreadTBoolean aSuitable );
```

PARAMETERS

This
This CaInfo.

aSuitable
NOVASPREAD_TRUE if the inserted smartcard is suitable. NOVASPREAD_FALSE otherwise.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.13. NovaspreadCaInfoSetSmartcardType

Sets the type and/or version of the smartcard.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetSmartcardType(
  NovaspreadTCaInfo This,
  const char *      aType );
```

PARAMETERS

This
This CaInfo.

aType
The type and/or version.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.14. NovaspreadCaInfoSetSmartcardNumber

Sets the smartcard's serial number.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetSmartcardNumber(
  NovaspreadTCaInfo This,
  const char *      aNumber );
```

PARAMETERS

This
This CaInfo.

aNumber
The serial number.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

2.11.15. NovaspreadCaInfoSetSmartcardStatus

Sets the smartcard status information as defined by the operator.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetSmartcardStatus(
  NovaspreadTCaInfo                This,
  NovaspreadTCaInfoSmartcardStatus aStatus );
```

PARAMETERS

This
This CaInfo.

aStatus
The status.

RETURN VALUE

NOVASPREAD_TRUE
if successful.

NOVASPREAD_FALSE
otherwise.

SEE ALSO

NovaspreadTCaInfoSmartcardStatus

2.11.16. NovaspreadCaInfoSetExpirationDate

Sets the expiration date. The expiration date shall be set if the status of the smartcard is ACTIVATED.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaInfoSetExpirationDate(
  NovaspreadTCaInfo This,
  NovaspreadTUInt32 aDate );
```

PARAMETERS

This
The CaInfo

aDate
The expiration date in UTC (seconds since 1 Jan 1970).

RETURN VALUE

NOVASPREAD_TRUE
if successful.

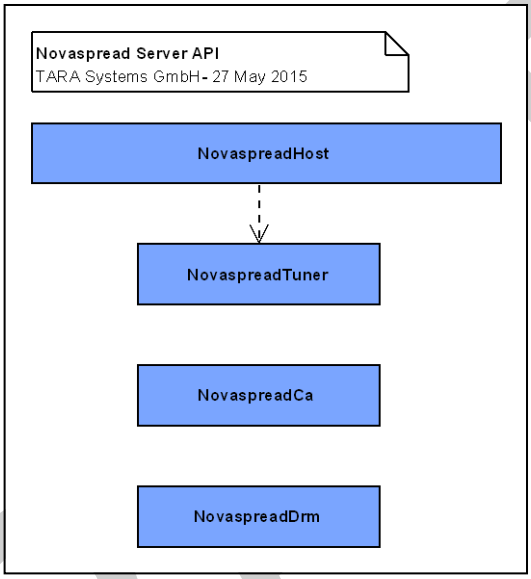
NOVASPREAD_FALSE
otherwise.

3. REQUIRED API

The following section describes the Application Programming Interface (API) which is required by Novaspread-S. All functions described in this section must be implemented for the target platform to which Novaspread-S is ported.

To use the interface include the file NovaspreadHost.h.

The following diagram gives an overview of the classes required by Novaspread-S.



3.1. NovaspreadHost

The Host is a required interface used by the NovaspreadServer. All methods must be implemented on target platforms to which NovaspreadServer is ported.

3.1.1. NovaspreadHostSetTunerReleaseRequestListener

Sets a TunerReleaseRequestListener at the NovaspreadHost. This function is called by NovaspreadServer once after initialization to set the listener. During shutdown, this function is called to unset the listener.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadHostSetTunerReleaseRequestListener(
  NovaspreadTTunerReleaseRequestListener aReleaseRequestListener );
```

PARAMETERS

aReleaseRequestListener
The ReleaseRequestListener to be set. NOVASPREAD_NULL to unset the listener.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadTTunerReleaseRequestListener

3.1.2. NovaspreadHostAllocateTuner

When a SAT>IP client connects to NovaspreadServer, NovaspreadServer calls this function to allocate a tuner for the given TunerAllocationParameters.

When the Host is able to fulfill the allocation request, the Host shall call the given AllocationFinishedListener and pass the Tuner to NovaspreadServer via this listener. The returned Tuner shall be tuned to the given tuning parameters.

When the Host denies the allocation request, e.g. because all Tuners are in use by higher priority usages, the Host shall call the AllocationFinishedListener and pass NOVASPREAD_NULL as Tuner via this listener and pass a corresponding error code (see enumeration type NovaspreadTTunerAllocationError for a list of available error codes).

If no local tuner is available, the Host can create a NovaspreadSatIpTuner by calling NovaspreadServerCreateSatIpTuner() and try to connect it to a SAT>IP LNB by calling NovaspreadSatIpTunerSetParameter() and NovaspreadSatIpTunerConnect(). If the connection is established, this tuner can also be provided via the AllocationFinishedListener.

If no tuner is available, the Host can call the TunerReleaseRequestListener, to release a tuner to fulfill the new request. For details on releasing tuners refer to NovaspreadTTunerReleaseRequestListener.

NovaspreadHostAllocateTuner() shall not block and return immediately.

The AllocationFinishedListener resp. the ReleaseRequestListener shall not be called from this function. NovaspreadHostAllocateTuner() returns a TunerRequestId, which must be available in NovaspreadServer first, before any of the listeners is called.

SYNTAX

```
PUBLIC NovaspreadTTunerRequestId
NovaspreadHostAllocateTuner(
  const NovaspreadTTunerAllocationParameters * aTunerAllocationParameters,
  NovaspreadTTunerAllocationFinishedListener   aAllocationFinishedListener );
```

PARAMETERS

aTunerAllocationParameters
A Tuner is requested, which can fulfill these parameters.

aAllocationFinishedListener
The AllocationFinishedListener which shall be called when the tuner allocation finished. It shall not be called from this function.

RETURN VALUE

A unique allocation RequestId. If 0 is returned, NovaspreadServer assumes that a critical error occurred and neither the AllocationFinishedListener nor the ReleaseRequestListener will be called.

SEE ALSO

NovaspreadTTunerAllocationParameters
NovaspreadTTunerRequestId
NovaspreadTTunerAllocationFinishedListener
NovaspreadTTunerReleaseRequestListener
NovaspreadHostCancelAllocation()

3.1.3. NovaspreadHostCancelAllocation

This function is called by NovaspreadServer if NovaspreadHostAllocateTuner() was called to allocate a Tuner, the request has not yet been fulfilled and NovaspreadServer does no longer need the Tuner.

This could happen e.g. if a tuner allocation is started, but the SAT>IP client closes the connection before the tuner allocation was finished by calling the AllocationFinishedListener.

When NovaspreadServer calls this function, the NovaspreadHost shall not call the AllocationFinishedListener for the given RequestId.

SYNTAX

```
PUBLIC void
NovaspreadHostCancelAllocation(
  NovaspreadTTunerRequestId aRequestId );
```

PARAMETERS

aRequestId
A RequestId which was returned by NovaspreadHostAllocateTuner().

3.2. NovaspreadTuner

The NovaspreadTuner is an interface required by the NovaspreadServer and must be implemented on target platforms to which NovaspreadServer is ported.

The NovaspreadServer allocates a Tuner by calling the NovaspreadHostAllocateTuner() function of the NovaspreadHost. A Tuner is always allocated for a particular transponder. So NovaspreadServer cannot change the tuner parameters (i.e. the transponder) of an already allocated Tuner. Instead NovaspreadServer will release the Tuner and call NovaspreadHostAllocateTuner() for the new tuning parameters.

A Tuner is a combination of a local tuner, transcoder,and transcryptor.

So when a Tuner is allocated, a complete transport stream processing pipeline must be available for this Tuner. See NovaspreadHostAllocateTuner() for details. NovaspreadTunerReadData() can be called to receive transport stream packets from the tuner. Only transport stream packets for pids are received which were set before by calling NovaspreadTunerSetPids(). NovaspreadServer will send these transport stream packets via RTP/UDP to the SAT>IP client.

When a transcoder was requested during NovaspreadHostAllocateTuner(), it shall be possible to change the transcoding parameters of the Tuner at run-time, to allow streaming of a different audio stream. The transcryption parameter can also be changed at run-time.

All functions of the NovaspreadTuner class will be called by NovaspreadServer in the context of the thread which calls NovaspreadServerProcess(). Except of the following functions, which may be called from different threads:

- NovaspreadTunerIsLocked()
- NovaspreadTunerGetSignalInfo()
- NovaspreadTunerReadData()

3.2.1. NovaspreadTTuner

This type defines the **NovaspreadTTuner** handle.

SYNTAX

```
typedef void * NovaspreadTTuner;
```

3.2.2. NovaspreadTTunerRequestId

Every time NovaspreadHostAllocateTuner() is called, a new RequestId is returned. This RequestId is used by NovaspreadServer to identify a tuner allocation request, e.g. when the TunerAllocationFinishedListener is called. It is used by NovaspreadHost to identify a tuner allocation request when NovaspreadHostCancelAllocation() is called.

0 is not a valid RequestId.

SYNTAX

```
typedef NovaspreadTUInt32 NovaspreadTTunerRequestId;
```

3.2.3. NovaspreadTTunerError

This type defines various error codes. As long as no error occurred, NovaspreadTunerGetError() shall return NOVASPREAD_TUNER_ERROR_NONE.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_ERROR_NONE

} NovaspreadTTunerError;
```

COMPONENTS

NOVASPREAD_TUNER_ERROR_NONE
No error occurred.

3.2.4. NovaspreadTTunerState

This type defines various states of a tuner.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_STATE_STOPPED,
  NOVASPREAD_TUNER_STATE_STREAMING,
  NOVASPREAD_TUNER_STATE_ERROR

} NovaspreadTTunerState;
```

COMPONENTS

NOVASPREAD_TUNER_STATE_STOPPED
The tuner is stopped. No data can be read via the tuner's ReadData() function.

NOVASPREAD_TUNER_STATE_STREAMING
The tuner was started successfully. Data can be read via the tuner's ReadData() function.

NOVASPREAD_TUNER_STATE_ERROR
An error occurred. When this state is reached, an error code shall be returned when NovaspreadTunerGetError() is called. To leave this state, NovaspreadTunerStop() must be called.

SEE ALSO

NovaspreadTTunerStateChangeListener

3.2.5. NovaspreadTTunerStateChangeListener

A listener of this type can be set at a Tuner. It is called every time the tuner's state changes.

SYNTAX

```
typedef void
(* NovaspreadTTunerStateChangeListener ) (
  void *                 aContext,
  NovaspreadTTunerState aNewState );
```

PARAMETERS

aContext

This context is passed unchanged from the NovaspreadTunerSetStateChangeListener() function.

aNewState
The new state of the tuner.

SEE ALSO

NovaspreadTTunerState

3.2.6. NovaspreadTTunerDataAvailableListener

A function of this type can be set at the Tuner. When NovaspreadTunerReadData() returns 0, because no data is available, the registered DataAvailableListener will be called as soon as data is available.

SYNTAX

```
typedef void
(* NovaspreadTTunerDataAvailableListener ) (
  void * aContext );
```

PARAMETERS

aContext
This context is passed unchanged from the NovaspreadTunerSetDataAvailableListener() function.

SEE ALSO

NovaspreadTunerSetDataAvailableListener()
NovaspreadTunerReadData()

3.2.7. NovaspreadTTunerAllocationMode

This type defines tuner allocation modes. An AllocationMode is passed to NovaspreadHostAllocateTuner() within the NovaspreadTTunerAllocationParameters.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_ALLOCATION_MODE_BEG,
  NOVASPREAD_TUNER_ALLOCATION_MODE_FORCE

} NovaspreadTTunerAllocationMode;
```

COMPONENTS

NOVASPREAD_TUNER_ALLOCATION_MODE_BEG
NovaspreadHost shall not release any Tuners used by NovaspreadServer.

If it is possible to fulfill the request without releasing Tuners used by NovaspreadServer, the Tuner shall be allocated.

If it is not possible to fulfill the request without releasing Tuners used by NovaspreadServer, NovaspreadHost shall only check, if the request could be fulfilled successfully, if Tuners would be released. If this is the case, the AllocationFinishedListener shall be called with the error code NOVASPREAD_TUNER_ALLOCATION_ERROR_FORCE_POSSIBLE. If it would not be possible to fulfill the request, even if Tuners would be released, a regular error code (like NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TUNER_AVAILABLE) shall be passed to the AllocationFinishedListener.

NOVASPREAD_TUNER_ALLOCATION_MODE_FORCE
NovaspreadHost shall release Tuners as described for NovaspreadTTunerReleaseRequestListener, to fulfill the allocation request.

SEE ALSO

NovaspreadTTunerAllocationParameters
NovaspreadTTunerReleaseRequestListener

3.2.8. NovaspreadTTunerAllocationParameters

A Tuner is allocated for particular tuner parameters. Additionally it is defined whether a transcoder and a transcryptor are required.

When the allocation request is fulfilled, the Tuner must be tuned to the given tuner parameters.

SYNTAX

```
typedef struct
{
  NovaspreadTTunerParameters     TunerParameters;
  NovaspreadTBoolean             AllocateTranscoder;
  NovaspreadTBoolean             AllocateTranscryptor;
  NovaspreadTDvbId               DvbId;
  NovaspreadTTunerAllocationMode AllocationMode;
  NovaspreadTUInt16              Priority;

} NovaspreadTTunerAllocationParameters;
```

COMPONENTS

TunerParameters
A Tuner is requested, which can fulfill this TunerParameter.

AllocateTranscoder
NOVASPREAD_TRUE if a transcoder shall be allocated. NOVASPREAD_FALSE if no transcoder is required.

AllocateTranscryptor
NOVASPREAD_TRUE if a transcryptor shall be allocated. NOVASPREAD_FALSE if no transcryptor is required.

DvbId
If AllocateTranscoder or AllocateTranscryptor is set to NOVASPREAD_TRUE, this DvbId identifies the service whose audio stream and video stream shall be transcoded or transcrypted. If neither a Transcoder nor a Transcryptor are allocated, all members of this DvbId shall be set to 0.

AllocationMode
The AllocationMode. See NovaspreadTTunerAllocationMode for details.

Priority
The Priority of this Request. The allocated Tuner shall get this priority. See NovaspreadTunerSetPriority() for details.

SEE ALSO

NovaspreadTTunerParameters
NovaspreadTTunerAllocationMode
NovaspreadTunerSetPriority()

3.2.9. NovaspreadTTunerAllocationError

This type defines errors, that may occur during tuner allocation.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_ALLOCATION_ERROR_NONE,
  NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TUNER_AVAILABLE,
  NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TRANSCODER_AVAILABLE,
  NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TRANSCRYPTOR_AVAILABLE,
  NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_LNB_AVAILABLE,
  NOVASPREAD_TUNER_ALLOCATION_ERROR_FORCE_POSSIBLE

} NovaspreadTTunerAllocationError;
```

COMPONENTS

NOVASPREAD_TUNER_ALLOCATION_ERROR_NONE

No error occurred. A tuner is available for Novaspread.

NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TUNER_AVAILABLE
No tuner is available to fulfill the allocation request.

NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TRANSCODER_AVAILABLE
No transcoder is available to fulfill the allocation request.

NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_TRANSCRYPTOR_AVAILABLE
No transcryptor is available to fulfill the allocation request.

NOVASPREAD_TUNER_ALLOCATION_ERROR_NO_LNB_AVAILABLE
For the allocation an unused LNB would be needed. All LNBs are already in used for either another frequency band or polarization.

NOVASPREAD_TUNER_ALLOCATION_ERROR_FORCE_POSSIBLE
It was not possible to fulfill the request in AllocationMode BEG. But an allocation with the same TunerAllocationParameters will be possible in AllocationMode FORCE.

SEE ALSO

NovaspreadTTunerAllocationFinishedListener
NovaspreadTTunerAllocationMode

3.2.10. NovaspreadTTunerAllocationFinishedListener

An AllocationFinishedListener is passed to NovaspreadHostAllocateTuner(). The listener shall be called when a Tuner is available, resp. when the allocation request is denied. This callback shall be called only once per allocation request.

This listener shall be called from the same thread which calls NovaspreadServerProcess().

SYNTAX

```
typedef void
(* NovaspreadTTunerAllocationFinishedListener ) (
  NovaspreadTTunerRequestId       aRequestId,
  NovaspreadTTuner                aTuner,
  NovaspreadTTunerAllocationError aAllocationError );
```

PARAMETERS

aRequestId
The RequestId, which was returned during NovaspreadHostAllocateTuner().

aTuner
A Tuner, if the allocation request could be fulfilled successfully. NOVASPREAD_NULL if the request was denied.

aAllocationError
If no tuner could be allocated, aAllocationError defines the reason.

SEE ALSO

NovaspreadTTunerRequestId
NovaspreadTTunerAllocationError
NovaspreadHostAllocateTuner()

3.2.11. NovaspreadTTunerReleaseReason

If a tuner release is requested, NovaspreadServer is informed about the reason.

SYNTAX

```
typedef enum
{
  NOVASPREAD_TUNER_RELEASE_REASON_PLAYER,
  NOVASPREAD_TUNER_RELEASE_REASON_RECORDING,
  NOVASPREAD_TUNER_RELEASE_REASON_MULTISCREEN,
```

```
  NOVASPREAD_TUNER_RELEASE_REASON_SHUT_DOWN,
  NOVASPREAD_TUNER_RELEASE_REASON_OTHER

} NovaspreadTTunerReleaseReason;
```

COMPONENTS

NOVASPREAD_TUNER_RELEASE_REASON_PLAYER
The host main player needs the tuner, which is currently used by NovaspreadServer.

NOVASPREAD_TUNER_RELEASE_REASON_RECORDING
A PVR recording is started. NovaspreadServer's tuner is needed for this recording.

NOVASPREAD_TUNER_RELEASE_REASON_MULTISCREEN
A tuner shall be released, to start another Multiscreen streaming.

NOVASPREAD_TUNER_RELEASE_REASON_SHUT_DOWN
The Multiscreen server is shut down.

NOVASPREAD_TUNER_RELEASE_REASON_OTHER
The tuner shall be released due to another reason (e.g. PiP)

SEE ALSO

NovaspreadTTunerReleaseRequestListener

3.2.12. NovaspreadTTunerReleaseRequestListener

A ReleaseRequestListener is passed to NovaspreadHostSetTunerReleaseRequestListener(). The listener can be called by the Host, if no tuner is available to fulfill a tuner allocation request, e.g. for starting a player, a recording or a Novaspread streaming. NovaspreadHost passes the Tuner which shall be released by NovaspreadServer. NovaspreadServer calls NovaspreadTunerRelease() for this Tuner when NovaspreadServerProcess() is called next.

If one or multiple tuners must be released, to fulfill an allocation request of NovaspreadServer, only Tuners may be released, which have an lower priority than the priority given in the allocation request. This means:

```
if ( Tuner.Priority < Request.Priority )
  Tuner release is allowed
```

NovaspreadHost must assure, that only Tuners are released, if it is really necessary.

```
Example: Current tuner allocation:
    /-- Tuner1, priority 2500
RF1
    \-- Tuner2, priority 2497

    /-- Tuner3, priority 2499
RF2
    \-- Tuner4, priority 2498
```

To get a free RF interface, NovaspreadHost shall not release tuners simply in sequence according to their priorities. Because in this case Tuner2, Tuner4, Tuner3 would be released in this order. However, it is not necessary to release Tuner2, to fulfill the request.

NovaspreadHost shall built-up groups of Tuners with the following properties:

1) If the Tuners of one group are released the allocation request can be fulfilled.

2) The groups shall be minimal, i.e. all sub-groups shall no longer fulfill the tuner allocation request.

For each Tuner Group a priority shall be calculated by the NovaspreadHost based on the Tuner priorities. The Group priority shall be the maximum priority of the Tuners contained in the Group.

e.g. (see example above)

```
Group1: contains Tuner1, Tuner2. Gets GroupPriority 2500.
Group2: contains Tuner3, Tuner4. Gets GroupPriority 2498.
```

The tuners in the group with the lowest priority shall be released. In the example above the group with the lowest priority would be Tuner3 and Tuner4. So the TunerReleaseRequestListener must be called once for Tuner3 and once for Tuner4.

This listener shall be called from the same thread which calls NovaspreadServerProcess().

SYNTAX

```
typedef void
(* NovaspreadTTunerReleaseRequestListener ) (
  NovaspreadTTunerReleaseReason aReleaseReason,
  NovaspreadTTuner              aTuner );
```

PARAMETERS

aReleaseReason
Defines why tuners shall be released. If NovaspreadHostAllocateTuner() was called to request a tuner, the ReleaseReason shall be MULTISCREEN.

aTuner
The Tuner which shall be released by NovaspreadServer.

SEE ALSO

```
NovaspreadHostSetTunerReleaseRequestListener()
NovaspreadTTunerReleaseReason
NovaspreadTunerSetPriority()
```

3.2.13. NovaspreadTunerRelease

This function is called by NovaspreadServer if a tuner is no longer used by NovaspreadServer, e.g. if a SAT>IP client closed the connection. The Host can use the tuner for another task.

SYNTAX

```
PUBLIC void
NovaspreadTunerRelease(
  NovaspreadTTuner This );
```

PARAMETERS

This
This Tuner is no longer accessed by NovaspreadServer.

3.2.14. NovaspreadTunerSetPriority

Sets the priority of this Tuner. The priority of a Tuner is used to decide which Tuner to release in the case of a tuner resource conflict. For details see description of NovaspreadTTunerReleaseRequestListener.

Lower values represent lower priorities, higher values represent higher priorities. This means it is possible to use the standard integer comparison operators to compare priorities.

Currently only values between 1000 and 3999 are used by Novaspread. The priority range between 2000 and 2999 is reserved for priorities of Novaspread Clients. However, Novaspread is not restricted to this range when tuners are allocated. Novaspread can use the full range of defined priorities from 1000 to 3999.

For example Novaspread can use priority 3600 in a tuner allocation request, in which case a tuner used for HDMI shall be released if no other tuner is available.

0000-0999
reserved for future use

1000-1999
low priority host features (PIP, Download, etc.)

2000-2999
NovaspreadClient features

3000-3999
high priority host features (HDMI=3500)

4000-65535
reserved for future use

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerSetPriority(
  NovaspreadTTuner  This,
  NovaspreadTUInt16 aPriority );
```

PARAMETERS

This
The Tuner

aPriority
The priority to be set.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadTTunerAllocationParameters
NovaspreadTTunerReleaseRequestListener

3.2.15. NovaspreadTunerGetTransportSessionId

Gets the 32-bit TransportSessionId of the Tuner. The returned ID uniquely identifies the stream received by the tuner. This ID is used to indicate the stream to be decrypted with NovaspreadCa functions and for control DRM specific re-encryption with NovaspreadDrm.

SYNTAX

```
PUBLIC NovaspreadTUInt32
NovaspreadTunerGetTransportSessionId(
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

The 32-bit TransportSessionId.

SEE ALSO

NovaspreadTCaServiceUsageRulesReceivedListener
NovaspreadCaSetServiceUsageRulesReceivedListener()

3.2.16. NovaspreadTunerSetTranscoding

Sets the Transcoding of the Tuner. This function will be called by NovaspreadServer only, if the Tuner was requested for transcoding during NovaspreadHostAllocateTuner().

The transcoding can be changed even if the Tuner is already started. In this case the tuner must check, which part of the transcoding parameters has changed. E.g. if the transcoding parameters for video did not change and only the parameters for audio changed, the video stream shall not be stopped. This is necessary to allow clients to request a different audio stream, e.g. in a different language, without interrupting the video during this change.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerSetTranscoding(
  NovaspreadTTuner          This,
  NovaspreadTTranscoding *  aTranscoding );
```

PARAMETERS

This
The Tuner.

aTranscoding
The Transcoding to be set.

RETURN VALUE

NOVASPREAD_TRUE
if the Transcoding was set successfully.

NOVASPREAD_FALSE
otherwise.

SEE ALSO

NovaspreadTTranscoding

3.2.17. NovaspreadTunerSetStateChangeListener

Sets a StateChangeListener at this tuner.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerSetStateChangeListener(
  NovaspreadTTuner                    This,
  NovaspreadTTunerStateChangeListener aListener,
  void *                              aContext );
```

PARAMETERS

This
The tuner.

aListener
The listener. Pass NOVASPREAD_NULL to unset the listener.

aContext
This context shall be passed unchanged to the listener.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadTTunerStateChangeListener
NovaspreadTTunerState

3.2.18. NovaspreadTunerGetError

If the Tuner changed to state ERROR, this function shall return an error code.

SYNTAX

```
PUBLIC NovaspreadTTunerError
NovaspreadTunerGetError(
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

The error code.

SEE ALSO

NovaspreadTTunerError

3.2.19. NovaspreadTunerSetPids

Sets the pids of the transport stream packets that shall be available in the Tuner's received stream. All previously set pids are replaced by this list. To reset all pids, pass NOVASPREAD_NULL for aPids and set aNoOfPids to 0. This function can be called when the tuner is stopped as well as when the tuner is started.

To avoid video flickering on client side, the Tuner shall not close all filters for previously set pids and open new filters for the pids in this list. Instead the Tuner must check, which filters must remain open. These shall not be closed.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerSetPids(
  NovaspreadTTuner    This,
  NovaspreadTUInt16 * aPids,
  NovaspreadTUInt32   aNoOfPids );
```

PARAMETERS

This
The Tuner.

aPids
The array of pids.

aNoOfPids
The number of pids in the array.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

3.2.20. NovaspreadTunerSetAllPids

The complete transport stream shall be available in the Tuner's received stream. All previously set pids are replaced by this list.

To reset all pids, call NovaspreadTunerSetPids() and pass NOVASPREAD_NULL for aPids and set aNoOfPids to 0. This function can be called when the tuner is stopped as well as when the tuner is started.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerSetAllPids(
```

```
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

3.2.21. NovaspreadTunerStart

This function starts the Tuner. When TunerStart() was called, NovaspreadTunerReadData() can be called to receive transport stream data.

Transcryption can be changed only when the tuner is stopped. Pids can be set when the tuner is stopped as well as when the tuner is started.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerStart(
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadTunerStop()

3.2.22. NovaspreadTunerStop

This function stops the data reception of this Tuner.

When this function returns, a registered DataAvailableListener shall no longer be called.

SYNTAX

```
PUBLIC void
NovaspreadTunerStop(
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

SEE ALSO

NovaspreadTunerStart()

3.2.23. NovaspreadTunerIsLocked

Returns whether the Tuner is locked or not. A Tuner is locked if a signal is detected for the set TunerParameter and the demodulator is able to decode the signal. This means a Tuner receives data only if it is locked. A valid lock status is returned if the tuner is started as well as when the tuner is stopped.

This function is called by NovaspreadServer in intervals of about 150 milliseconds.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerIsLocked(
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

NOVASPREAD_TRUE
if the Tuner is locked.

NOVASPREAD_FALSE
otherwise

3.2.24. NovaspreadTunerGetSignalInfo

Gets the SignalInfo of this Tuner. See the data type NovaspreadTTunerSignalInfo for a full description of the SignalInfo.

This function is called by NovaspreadServer in intervals of about 150 milliseconds.

SYNTAX

```
PUBLIC NovaspreadTTunerSignalInfo
NovaspreadTunerGetSignalInfo(
  NovaspreadTTuner This );
```

PARAMETERS

This
The Tuner.

RETURN VALUE

The SignalInfo

SEE ALSO

NovaspreadTTunerSignalInfo

3.2.25. NovaspreadTunerSetDataAvailableListener

Sets a DataAvailableListener. Only one DataAvailableListener can be set at a Tuner.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadTunerSetDataAvailableListener(
  NovaspreadTTuner                       This,
  NovaspreadTTunerDataAvailableListener aListener,
  void *                                 aContext );
```

PARAMETERS

This

The Tuner.

aListener
The DataAvailableListener. NOVASPREAD_NULL is passed to unset the listener.

aContext
This context shall be passed unchanged to the listener.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

```
NovaspreadTTunerDataAvailableListener
NovaspreadTunerReadData()
```

3.2.26. NovaspreadTunerReadData

As soon as the Tuner is started, this function will write 188 bytes long transport stream packets to the buffer. This function must be called periodically to avoid a Tuner internal buffer overflow.

It is not required to write only complete transport stream packets to the buffer. If e.g. a buffer size of 200 bytes is given, and 200 bytes are available, they shall be written to the buffer.

If this function is called when the Tuner is stopped, it will not write data to the buffer and return 0.

If the tuner is started, NovaspreadTunerReadData() will return 0, if no data is available. As soon as data is available again, a previously set DataAvailableListener shall be called.

NovaspreadTunerReadData() will not be called in the context of the DataAvailableListener.

SYNTAX

```
PUBLIC NovaspreadTUInt32
NovaspreadTunerReadData(
  NovaspreadTTuner   This,
  NovaspreadTUInt8 * aBuffer,
  NovaspreadTUInt32  aBufferSize );
```

PARAMETERS

This
The Tuner.

aBuffer
Transport stream packets are written to this buffer.

aBufferSize
The size of the buffer.

RETURN VALUE

The number of bytes written to the buffer. If there are no transport stream packets available 0 is returned.

3.3. NovaspreadCa

The NovaspreadCa interface contains functions Novaspread requires from the CA system. The main purpose of this interface is to retrieve the UsageRules on platform and service level from the CA system for a specific stream.

3.3.1. NovaspreadTCaPlatformUsageRulesReceivedListener

This listener must be called, when platform dependent UsageRules have been received. The structure of the passed UsageRules depends on the used CA system.

SYNTAX

```
typedef void
(* NovaspreadTCaPlatformUsageRulesReceivedListener ) (
  void *              aContext,
  NovaspreadTUInt8 *  aPlatformUsageRules,
  NovaspreadTUInt32   aLength );
```

PARAMETERS

aContext
The context which was given to NovaspreadCaSetPlatformUsageRulesReceivedListener() shall be passed unchanged to this listener.

aPlatformUsageRules
The UsageRules on platform level.

aLength
The length of the UsageRules buffer.

SEE ALSO

NovaspreadCaSetPlatformUsageRulesReceivedListener()

3.3.2. NovaspreadTCaServiceUsageRulesReceivedListener

A listener of this type can be registered at NovaspreadCa. It is to be called whenever new UsageRules for the particular service are received.

If UsageRules are only received if they are updated, this listener must be called at least once when it is registered with NovaspreadCa.

SYNTAX

```
typedef void
(* NovaspreadTCaServiceUsageRulesReceivedListener ) (
  void *              aContext,
  NovaspreadTUInt32   aTransportSessionId,
  NovaspreadTUInt8 *  aServiceUsageRules,
  NovaspreadTUInt32   aLength );
```

PARAMETERS

aContext
This context is passed unchanged from the NovaspreadCaSetUsageRulesReceivedListener() function.

aTransportSessionId
The UsageRules of this TransportSession have been updated.

aServiceUsageRules
The UsageRules on service level.

aLength
The length of the aServiceUsageRules.

3.3.3. NovaspreadCaGetInfo

This function returns information about the CA system. The returned CaInfo will be destroyed by NovaspreadServer.

SYNTAX

```
PUBLIC NovaspreadTCaInfo
```

```
NovaspreadCaGetInfo(
  void );
```

RETURN VALUE

A new CaInfo if successful. NOVASPREAD_NULL otherwise.

SEE ALSO

NovaspreadTCaInfo

EXAMPLE

```
// An implementation of this function shall proceed as follows:

PUBLIC NovaspreadTCaInfo
NovaspreadCaGetInfo ( void )
{
  NovaspradTCaInfo  caInfo;
  const char *      caVendor = "Nagra";

  caInfo = NovaspreadCaInfoCreate();
  if ( ! caInfo )
    return NOVASPREAD_NULL;

  // For Nagra, set the NUId by calling
  // NovaspreadCaInfoSetChipsetUid().

  NovaspreadCaInfoSetCaVendor( caInfo, caVendor );

  // Call NovaspreadCaInfoSet..() functions here to set
  // information about the CA system and the smartcard.

  return caInfo;
}
```

3.3.4. NovaspreadCaSetPlatformUsageRulesReceivedListener

This functions sets a listener, which shall be called when platform specific usage rules are received.

SYNTAX

```
PUBLIC void
NovaspreadCaSetPlatformUsageRulesReceivedListener(
  NovaspreadTCaPlatformUsageRulesReceivedListener * aListener,
  void *                                            aContext );
```

PARAMETERS

aListener
The PlatformUsageRulesReceivedListener to be set. NOVASPREAD_NULL is passed to unset the listener.

aContext
The context which shall be passed unchanged to the listener.

SEE ALSO

```
NovaspreadTunerGetTransportSessionId()
NovaspreadTCaPlatformUsageRulesReceivedListener
```

3.3.5. NovaspreadCaSetServiceUsageRulesReceivedListener

Sets a ServiceUsageRulesReceivedListener. The TransportSessionId identifies the stream received by a Tuner for which the UsageRules should be acquired. The TransportSessionId can be retrieved with the function NovaspreadTunerGetTransportSessionId().

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadCaSetServiceUsageRulesReceivedListener(
  NovaspreadTUInt32                              aTransportSessionId,
  NovaspreadTCaServiceUsageRulesReceivedListener aListener,
  void *                                         aContext );
```

PARAMETERS

aTransportSessionId
For this TransportSession the listener is set.

aListener
The ServiceUsageRulesReceivedListener to be set. NOVASPREAD_NULL is passed to unset the listener.

aContext
This context shall be passed unchanged to the listener.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise

SEE ALSO

NovaspreadTunerGetTransportSessionId()
NovaspreadTCaServiceUsageRulesReceivedListener

3.4. NovaspreadDrm

The NovaspreadDrm interface contains functions to control the DRM system. Via NovaspreadDrmSetParameters() parameters can be set, which shall be used when the re-encryption is started. A LicenseChangeListener can be set at NovaspreadDrm. The listener shall be called whenever the license changed.

3.4.1. NovaspreadTDrmLicense

A license returned by the DRM system.

SYNTAX

```
typedef struct
{
  NovaspreadTUInt8 * License;
  NovaspreadTUInt32  LicenseLength;

} NovaspreadTDrmLicense;
```

COMPONENTS

License
The License as byte array.

LicenseLength
The length of the License.

3.4.2. NovaspreadTDrmLicenseParameters

These parameters are passed to the NovaspreadDrmSetParameters() function.

If an OldLicense is passed, this license shall be re-used. If this is not possible or if no OldLicense is used, the other components shall be used to create a new license.

SYNTAX

```
typedef struct
{
  NovaspreadTDrmLicense OldLicense;
  NovaspreadTUInt32     CollectionId;
  NovaspreadTUInt32     Duration;
  NovaspreadTUInt8 *    UsageRules;
  NovaspreadTUInt32     UsageRulesLength;

} NovaspreadTDrmLicenseParameters;
```

COMPONENTS

OldLicense
A license previously returned via a LicenseChangeListener. If no OldLicense is available, the content of this OldLicense is 0.

CollectionId
Defines the CollectionId which shall be passed to the underlying DRM system.

Duration
Defines how long a new license shall be valid. In seconds.

UsageRules
These UsageRules shall be set for encryption.

UsageRulesLength
This UsageRulesLength shall be set for encryption

3.4.3. NovaspreadTDrmLicenseChangeListener

A function of this type can be set at NovaspreadDrm. It shall be called every time the license changed.

SYNTAX

```
typedef void
(* NovaspreadTDrmLicenseChangeListener ) (
  void *                aContext,
  NovaspreadTDrmLicense * aLicense );
```

PARAMETERS

aContext
This context is passed unchanged from the NovaspreadDrmSetLicenseChangeListener() function.

aLicense
The new license. When this callback returned, NovaspreadServer does no longer access the memory of this license. So it can be released.

3.4.4. NovaspreadDrmSetParameters

This function sets parameters which shall be used for the re-encryption of the transport stream.

The TransportSessionId, which is passed to this function, can be got by a call to NovaspreadTunerGetTransportSessionId().

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadDrmSetParameters(
```

```
  NovaspreadTUInt32                 aTransportSessionId,
  NovaspreadTDrmLicenseParameters * aLicenseParameters );
```

PARAMETERS

aTransportSessionId
For this TransportSession the re-encryption is started.

aLicenseParameters
The parameters which shall be used for re-encryption.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise.

SEE ALSO

NovaspreadTunerGetTransportSessionId()
NovaspreadTDrmLicenseParameters
NovaspreadTDrmLicense

3.4.5. NovaspreadDrmSetLicenseChangeListener

This function sets a LicenseChangeListener at NovaspreadDrm.

SYNTAX

```
PUBLIC NovaspreadTBoolean
NovaspreadDrmSetLicenseChangeListener(
  NovaspreadTUInt32                   aTransportSessionId,
  NovaspreadTDrmLicenseChangeListener aListener,
  void *                              aContext );
```

PARAMETERS

aTransportSessionId
For this TransportSession the listener is set.

aListener
The listener to be set. Pass NOVASPREAD_NULL to unset the listener.

aContext
This context shall be passed unchanged to the listener.

RETURN VALUE

NOVASPREAD_TRUE
if successful

NOVASPREAD_FALSE
otherwise.

Published by:

SES Platform Services GmbH

Beta Straße 1-10

85774 Unterföhring

Germany

For more information about SES, visit

www.ses-ps.com or email **info@ses-ps.com**