IviLxiSync Reference
IIviLxiSyncTriggerSources.Add Method

This function creates a new trigger source.

[Visual Basic]
Public Sub Add(_
    ByVal SourceName As String _
)

[C#]
public void Add(
    String SourceName
);

[C++]
HRESULT Add(
    BSTR SourceName
);

Parameters

SourceName
    Specifies the name of the trigger source to create.

Remarks

The SourceName parameter is case-insensitive but case-preserving. This means that any casing of the SourceName parameter can be used to access the arm source within the repeated capability collection, but the original casing is used by the specific driver when identifying trigger source events on the LAN.

See Also

IIviLxiSyncTriggerSources Interface | IIviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace
IIviLxiSyncTrigger.Alarms Property

Pointer to the ILxiSyncTriggerAlarms interface.

[Visual Basic]
Public ReadOnly Property Alarms As IIviLxiSyncTriggerAlarms

[C#]
public IIviLxiSyncTriggerAlarms Alarms {get;}

[C++]
HRESULT get_Alarms(
    IIviLxiSyncTriggerAlarms** retval
);

See Also

IIviLxiSyncTrigger Interface | IIviLxiSyncTrigger Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSync.Arm Property

Pointer to the ILxiSyncArm interface.

[Visual Basic]
Public ReadOnly Property Arm As IIviLxiSyncArm

[C#]
public IIviLxiSyncArm Arm {get;}

[C++]
HRESULT get_Arm(
    IIviLxiSyncArm** retval
);

See Also

IIviLxiSync Interface | IIviLxiSync Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArm.ArmCount Property

Specifies the number of times the arm has to occur to complete the arm loop; that is, the number of arms that are accepted before the measurement must be initiated again.

[Visual Basic]
Public Property ArmCount As Int32

[C#]
public Int32 ArmCount {get; set;}

[C++]
HRESULT get_ArmCount(
    long* retval
);
HRESULT put_ArmCount(
    long val
);

See Also

IIviLxiSyncArm Interface | IIviLxiSyncArm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncEventLog.Clear Method

This function removes all existing entries from the event log.

[Visual Basic]
Public Sub Clear(  
)

[C#]
public void Clear(  
);

[C++]
HRESULT Clear(  
);

See Also

IIviLxiSyncEventLog Interface | IIviLxiSyncEventLog Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarm.Configure Method

This function configures the most commonly configured attributes of the arm alarm sub-system.

[Visual Basic]
Public Sub Configure(_
    ByVal Enabled As Boolean, _
    ByVal TimeSeconds As Double, _
    ByVal TimeFraction As Double, _
    ByVal Period As Double, _
    ByVal RepeatCount As Int32 _
)

[C#]
public void Configure(
    Boolean Enabled,
    Double TimeSeconds,
    Double TimeFraction,
    Double Period,
    Int32 RepeatCount
);

[C++]
HRESULT Configure(
    VARIANT_BOOL Enabled,
    double TimeSeconds,
    double TimeFraction,
    double Period,
    long RepeatCount
);

Parameters

Enabled
   Enables or disables the arm alarm.
TimeSeconds
   Specifies the seconds part of 1588 time.
**TimeFraction**
Specifies the fractional part of 1588 time.

**Period**
Specifies the period of the arm alarm.

**RepeatCount**
Specifies the number of times to repeat the trigger at the period specified by the Arm Alarm Repeat Period attribute.

**See Also**

[IIviLxiSyncArmAlarm Interface](#) | [IIviLxiSyncArmAlarm Members](#) | [Ivi.LxiSync.Interop Namespace](#)

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
Configuring the Driver

The [IVI Configuration Store](#) is part of the [IVI Shared Components](#) distributed by the IVI Foundation. The IVI Configuration Store serves as a central repository for driver registration information. Physically, the IVI Configuration Store is an XML file installed by the IVI Shared Component Installer in the following directory:

`<IviInstallDir>\Data\IviConfigurationStore.xml`

where, `<IviInstallDir>` is typically `C:\Program Files\IVI`.

The IviLxiSync installer populates the IVI Configuration Store (Config Store) with the required entries for using the driver. Specifically, the driver installer adds a [Software Module](#) entry to the Config Store to represent the driver DLL itself. No other entries in the Config Store are created by the driver. If you only wish to access the instrument specific functionality of the driver, then no other entries in the Config Store are required.

To use the driver in application programs that will be interchangeable, you must edit the Config Store and add a few specific entries. Instrument vendors and other third parties provide graphical tools for conveniently editing the Config Store. One such popular tool is NI-MAX from National Instruments.

**Note** Although the IVI Configuration Store is a human readable XML file, the IVI Foundation strongly discourages you from directly editing this file. The Config Store file is highly self-referential and easily corrupted if directly modified. Instead, use one of the available third-party editors (such as NI-MAX from National Instruments). Alternatively, the IVI Shared Components include a programmatic interface to the Config Store, known as the [IVI Configuration Server](#).
Editing the IVI Configuration Store

The driver installer creates only one entry in the Config Store -- the Software Module. This entry is removed by the driver uninstaller. *End users should never modify the Software Module entry.*

To author test programs that allow for interchangeability, you must create three additional entries in the Config Store:

- Hardware Asset
- Driver Session
- Logical Name

The following sections provide descriptions of each of the Config Store entries mentioned here. Consult the documentation for your Config Store editor for detailed instructions on modifying these items.

**Software Module**

The Software Module identifies the instrument driver DLL, as well as various pieces of important driver information, such as the *physical repeated capability names* used for the *repeated capabilities*, IVI compliance information, as well as a variety of other items.

**Hardware Asset**

The Hardware Asset identifies a specific piece of instrument hardware. The main purpose of the Hardware Asset is to house the I/O resource descriptor used to communicate with the device.

**Driver Session**

The Driver Session associates the Software Module with the Hardware Asset. The Driver Session also specifies such things as driver initialization values and mappings between *physical repeated capability names* and *virtual repeated capability names*. Virtual names are used by client programs to access repeated capabilities in an interchangeable fashion.
For a discussion of repeated capabilities, physical names, and virtual names, see the Repeated Capabilities topic in the IVI Backgrounder.

Users create a separate Driver Session for each unique combination of driver software, instrument hardware, and initialization settings. You may have multiple Driver Sessions for the same Software Module, if, for instance, the driver is being used to control several instances of the same instrument at different GPIB addresses.

**Logical Name**

The Logical Name allows users to associate an arbitrary name with a Driver Session. Since the Driver Session is tied to a particular driver and instrument, a level of indirection is needed to abstract these details in interchangeable client applications. The Logical Name serves as this extra level of abstraction -- a simple string identifier for a Driver Session.

The Logical Name is what client programs must use to build interchangeable systems. Applications create instances of drivers by using the COM Session Factory and specifying the Logical Name. Thus, no instrument-specific details exist in the client program. Swapping instruments in such a system involves merely associating the Logical Name in the Config Store with a different Driver Session (i.e. a different driver-instrument combination).

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSources.Count Property

The number of sources.

[Visual Basic]
Public ReadOnly Property Count As Int32

[C#]
public Int32 Count {get;}

[C++]
HRESULT get_Count(
    long* retval
);

See Also

IIviLxiSyncTriggerSources Interface | IIviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncTriggerSource.Delay Property**

Specifies the trigger source delay from when the trigger logic is satisfied until the device specific action (for instance a measurement) is triggered.

*[Visual Basic]*

```vbnet
Public Property Delay As Double
```

*[C#]*

```csharp
public Double Delay {get; set;}
```

*[C++]*

```cpp
HRESULT get_Delay(
    double* retval
);
HRESULT put_Delay(
    double val
);
```

**Remarks**

A negative value implies pre-trigger acquisition. The units are seconds.

**See Also**

[IIviLxiSyncTriggerSource Interface] | [IIviLxiSyncTriggerSource Members] | [Ivi.LxiSync.Interop Namespace]

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent.DestinationPath Property

Specifies a list of places to send the event.

[Visual Basic]
Public Property DestinationPath As String

[C#]
public String DestinationPath {get; set;}

[C++]
HRESULT get_DestinationPath(
    BSTR* retval
);
HRESULT put_DestinationPath(
    BSTR val
);

Remarks

The default value for this attribute is the repeated capability name.

See Also

IIviLxiSyncEvent Interface | IIviLxiSyncEvent Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmSource.Detection Property

Specifies the style of arm source detection.

[Visual Basic]
Public Property Detection As IviLxiSyncArmSourceDetectionEnum

[C#]
public IviLxiSyncArmSourceDetectionEnum Detection {get; set;}

[C++]
HRESULT get_Detection(

   IviLxiSyncArmSourceDetectionEnum* retval
);
HRESULT put_Detection(

   IviLxiSyncArmSourceDetectionEnum val
);

Remarks

If the source is a LAN event and the source detection is set to rise, this Arm repeated capability will be satisfied when the designated LAN packet arrives with a True indication. If the source detection is set to fall, this Arm repeated capability will be satisfied when a LAN packet arrives with a False indication. If the detection is set to high, the source will be satisfied when the designated LAN packet arrives with a True indication and remain satisfied until the designated LAN packet arrives with a False indication. If the detection is to low, the source will be satisfied when the designated LAN packet arrives with a False indication and remain satisfied until the designated LAN packet arrives with a True indication.

See Also

IIviLxiSyncArmSource Interface | IIviLxiSyncArmSource Members | Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
IIviLxiSyncArmSources.DisableAll Method

This function disables all arm sources.

[Visual Basic]
Public Sub DisableAll(_)

[C#]
public void DisableAll(_);

[C++]
HRESULT DisableAll(_);

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent.DriveMode Property

Specifies how this event is transmitted.

[Visual Basic]
Public Property DriveMode As IIviLxiSyncEventDriveModeEnum

[C#]
public IIviLxiSyncEventDriveModeEnum DriveMode {get; set;}

[C++]
HRESULT get_DriveMode(
    IIviLxiSyncEventDriveModeEnum* retval
);  
HRESULT put_DriveMode(
    IIviLxiSyncEventDriveModeEnum val
);

Remarks

It is an error to turn on the Wired OR Bias Mode for this device for a particular LXI trigger line and then set the Event Enabled attribute to On instead of Wired OR for an event whose destination path includes that LXI trigger line.

See Also

IIviLxiSyncEvent Interface | IIviLxiSyncEvent Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IviLxiSyncErrorCodesEnum
Enumeration

[Visual Basic]
Public Enum IviLxiSyncErrorCodesEnum

[C#]
public enum IviLxiSyncErrorCodesEnum

[C++]
enum IviLxiSyncErrorCodesEnum

Members

Member name

E_IVILXISYNC_ALARM_TIME_INVALID

E_IVILXISYNC_EVENT_SOURCE_EXISTS

E_IVILXISYNC_OUT_OF_EVENT_RESOURCES

E_IVILXISYNC_EVENT_SOURCE_DOES_NOT_EXIST

E_IVILXISYNC_EVENT_SOURCE_NOT_SET
E_IVILXISYNC_INVALID_EVENT_SOURCE

E_IVILXISYNC_ALARM_EXISTS

E_IVILXISYNC_ALARM_DOES_NOT_EXIST

E_IVILXISYNC_WIRED_OR_MODE_INVALID

E_IVILXISYNC_CANT_REMOVE_RESERVED_REPEATED_CAPABILITY

Requirements

Namespace: Ivi.LxiSync.Interop

See Also

Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncArmSource.Enabled Property**

Enables or disables the arm source.

[Visual Basic]
Public Property Enabled As Boolean

[C#]
public Boolean Enabled {get; set;}

[C++]
HRESULT get.Enabled(
    VARIANT_BOOL* retval
);
HRESULT put.Enabled(
    VARIANT_BOOL val
);

**Remarks**

If a source is disabled, it has no affect on the summary arm signal.

**See Also**

[IIviLxiSyncArmSource Interface] | [IIviLxiSyncArmSource Members] | [Ivi.LxiSync.Interop Namespace]

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncEventLog.EntryCount Property

Returns the number of event log entries available.

[Visual Basic]
Public ReadOnly Property EntryCount As Int32

[C#]
public Int32 EntryCount {get;}

[C++]
HRESULT get_EntryCount(
    long* retval
);

See Also

IIviLxiSyncEventLog Interface | IIviLxiSyncEventLog Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSource.EventId Property

Specifies the LAN event identifier that is associated with this trigger source.

[Visual Basic]
Public Property EventId As String

[C#]
pUBLIC String EventId {get; set;}

[C++]
HRESULT get_EventId(
    BSTR* retval
);
HRESULT put_EventId(
    BSTR val
);

Remarks

LAN Events with this identifier are accepted from the source described in the filter. The default value for the Event ID is the repeated capability specifier for this trigger source.

See Also

IIviLxiSyncTriggerSource Interface | IIviLxiSyncTriggerSource Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSync.EventLog Property**

Pointer to the ILxiSyncEventLog interface.

[Visual Basic]
Public ReadOnly Property EventLog As IIviLxiSyncEventLog

[C#]
public IIviLxiSyncEventLog EventLog {get;}

[C++]
HRESULT get_EventLog(IIviLxiSyncEventLog** retval);

See Also

IIviLxiSync Interface | IIviLxiSync Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSync.Events Property

Pointer to the ILxiSyncEvents interface.

[Visual Basic]
Public ReadOnly Property Events As IIviLxiSyncEvents

[C#]
public IIviLxiSyncEvents Events {get;}

[C++]
HRESULT get_Events(
    IIviLxiSyncEvents** retval
);

See Also

IIviLxiSync Interface | IIviLxiSync Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSource.Filter Property

Specifies a filter for restricting trigger sources.

[Visual Basic]
Public Property Filter As String

[C#]
public String Filter {get; set;}

[C++]
HRESULT get_Filter(
    BSTR* retval
);
HRESULT put_Filter(
    BSTR val
);

Remarks

The filter specified by this attribute denotes the accepted sources

See Also

IIviLxiSyncTriggerSource Interface | IIviLxiSyncTriggerSource Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEventLog.GetNextEntry Method

This function retrieves and clears the oldest event log entry for the IVI session.

[Visual Basic]
Public Function GetNextEntry( ) As String

[C#]
public String GetNextEntry( );

[C++]
HRESULT GetNextEntry( 
    BSTR* retval
 );

Return Value

Remarks

If there are no entries in the event log, then function returns an empty string.

See Also

IIviLxiSyncEventLog Interface | IIviLxiSyncEventLog Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTime.GetSystemTime Method

This function retrieves the current 1588 time.

[Visual Basic]
Public Sub GetSystemTime( _
    ByRef TimeSeconds As Double, _
    ByRef TimeFractional As Double _
)

[C#]
public void GetSystemTime(
    ref Double TimeSeconds,
    ref Double TimeFractional
);

[C++]
HRESULT GetSystemTime(
    double* TimeSeconds,
    double* TimeFractional
);

Parameters

TimeSeconds
Indicates the seconds portion of the current 1588 time.

TimeFractional
Indicates the fractional portion of the current 1588 time.

See Also

IIviLxiSyncTime Interface | IIviLxiSyncTime Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSync Members

IIviLxiSync overview

Public Properties

<table>
<thead>
<tr>
<th>Arm</th>
<th>Pointer to the ILxiSyncArm interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventLog</td>
<td>Pointer to the ILxiSyncEventLog interface.</td>
</tr>
<tr>
<td>Events</td>
<td>Pointer to the ILxiSyncEvents interface.</td>
</tr>
<tr>
<td>Time</td>
<td>Pointer to the ILxiSyncTime interface.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Pointer to the ILxiSyncTrigger interface.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSync Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSync Interface

LxiSync root interface.

For a list of all members of this type, see IIviLxiSync Members.

[Visual Basic]
Public Interface IIviLxiSync

[C#]
public interface IIviLxiSync

[C++]
__interface IIviLxiSync

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSync Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSync Properties

The properties of the IIviLxiSync interface are listed here. For a complete list of IIviLxiSync interface members, see the IIviLxiSync Members topic.

Public Properties

- **Arm**: Pointer to the ILxiSyncArm interface.
- **EventLog**: Pointer to the ILxiSyncEventLog interface.
- **Events**: Pointer to the ILxiSyncEvents interface.
- **Time**: Pointer to the ILxiSyncTime interface.
- **Trigger**: Pointer to the ILxiSyncTrigger interface.

See Also

IIviLxiSync Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSync.Time Property

Pointer to the ILxiSyncTime interface.

[Visual Basic]
Public ReadOnly Property Time As IIviLxiSyncTime

[C#]
public IIviLxiSyncTime Time {get;}

[C++]
HRESULT get_Time(
    IIviLxiSyncTime** retval
);

See Also

IIviLxiSync Interface | IIviLxiSync Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSync.Trigger Property

Pointer to the ILxiSyncTrigger interface.

[Visual Basic]
Public ReadOnly Property Trigger As IIviLxiSyncTrigger

[C#]
public IIviLxiSyncTrigger Trigger {get;}

[C++]
HRESULT get_Trigger(
    IIviLxiSyncTrigger** retval
);

See Also

IIviLxiSync Interface | IIviLxiSync Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArm Members

IIviLxiSyncArm overview

Public Properties

| Alarms | Pointer to the ILxiSyncArmAlarms interface. Specifies the number of times the arm has to occur to complete the arm loop; that is, the number of arms that are accepted before the measurement must be initiated again. |
| ArmCount | Specifies the delay from when the arm logic satisfied until the waiting for trigger state is entered. The units are seconds. |
| Delay | Pointer to the ILxiSyncArmSources interface. |
| Sources |

See Also

IIviLxiSyncArm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncArm Interface**

LxiSync arm interface.

For a list of all members of this type, see [IIviLxiSyncArm Members](#).

```vbnet
Public Interface IIviLxiSyncArm
```

```csharp
public interface IIviLxiSyncArm
```

```cpp
__interface IIviLxiSyncArm
```

**Requirements**

**Namespace:** [Ivi.LxiSync.Interop](#)

**Assembly:** Ivi.LxiSync.Interop (in Ivi.LxiSync.Interop.dll)

**See Also**

[IIviLxiSyncArm Members](#) | [Ivi.LxiSync.Interop Namespace](#)

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArm Properties

The properties of the IIviLxiSyncArm interface are listed here. For a complete list of IIviLxiSyncArm interface members, see the IIviLxiSyncArm Members topic.

Public Properties

- **Alarms**: Pointer to the ILxiSyncArmAlarms interface. Specifies the number of times the arm has to occur to complete the arm loop; that is, the number of arms that are accepted before the measurement must be initiated again.

- **ArmCount**: Specifies the delay from when the arm logic satisfied until the waiting for trigger state is entered. The units are seconds.

- **Sources**: Pointer to the ILxiSyncArmSources interface.

See Also

IIviLxiSyncArm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArm.Alarms Property

Pointer to the ILxiSyncArmAlarms interface.

[Visual Basic]
Public ReadOnly Property Alarms As IIviLxiSyncArmAlarms

[C#]
public IIviLxiSyncArmAlarms Alarms {get;}

[C++]
HRESULT get_Alarms(    
IIviLxiSyncArmAlarms** retval
);

See Also

IIviLxiSyncArm Interface | IIviLxiSyncArm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArm.Delay Property

Specifies the delay from when the arm logic satisfied until the waiting for trigger state is entered. The units are seconds.

[Visual Basic]
Public Property Delay As Double

[C#]
public Double Delay {get; set;}

[C++]
HRESULT get_Delay(
    double* retval
);
HRESULT put_Delay(
    double val
);

See Also

IIviLxiSyncArm_Interface | IIviLxiSyncArm_Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArm.Sources Property

Pointer to the ILxiSyncArmSources interface.

[Visual Basic]
Public ReadOnly Property Sources As IIviLxiSyncArmSources

[C#]
public IIviLxiSyncArmSources Sources {get;}

[C++]
HRESULT get_Sources(
    IIviLxiSyncArmSources** retval
);

See Also

IIviLxiSyncArm Interface | IIviLxiSyncArm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarm Members

IIviLxiSyncArmAlarm overview

Public Properties

**Enabled**
Enables or disables the arm alarm.

**Period**
Specifies the period of the arm alarm in seconds; that is, the amount of time in seconds that transpire before the alarm repeats.

**RepeatCount**
Specifies the number of times to repeat the trigger at the period specified by the Arm Alarm Period attribute.

**TimeFraction**
Specifies the fractional portion of the time at which the alarm will go off.

**TimeSeconds**
Specifies the seconds portion of time at which the alarm will go off.

Public Methods

**Configure**
This function configures the most commonly configured attributes of the arm alarm sub-system.

See Also

IIviLxiSyncArmAlarm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarm Interface

LxiSync arm alarm interface.

For a list of all members of this type, see IIviLxiSyncArmAlarm Members.

[Visual Basic]
Public Interface IIviLxiSyncArmAlarm

[C#]
public interface IIviLxiSyncArmAlarm

[C++]
__interface IIviLxiSyncArmAlarm

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncArmAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarm Methods

The methods of the IIviLxiSyncArmAlarm interface are listed here. For a complete list of IIviLxiSyncArmAlarm interface members, see the IIviLxiSyncArmAlarm Members topic.

Public Methods

Configure

This function configures the most commonly configured attributes of the arm alarm sub-system.

See Also

IIviLxiSyncArmAlarm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarm Properties

The properties of the IIviLxiSyncArmAlarm interface are listed here. For a complete list of IIviLxiSyncArmAlarm interface members, see the IIviLxiSyncArmAlarm Members topic.

Public Properties

- **Enabled**
  Enables or disables the arm alarm.
  Specifies the period of the arm alarm in seconds; that is, the amount of time in seconds that transpire before the alarm repeats.

- **Period**
  Specifies the number of times to repeat the trigger at the period specified by the Arm Alarm Period attribute.

- **RepeatCount**
  Specifies the fractional portion of the time at which the alarm will go off.

- **TimeFraction**
  Specifies the seconds portion of time at which the alarm will go off.

See Also

IIviLxiSyncArmAlarm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarm.Enabled Property

Enables or disables the arm alarm.

[Visual Basic]
Public Property Enabled As Boolean

[C#]
public Boolean Enabled {get; set;}

[C++]
HRESULT get_Enabled(
   VARIANT_BOOL* retval
);
HRESULT put_Enabled(
   VARIANT_BOOL val
);

See Also

IIviLxiSyncArmAlarm Interface | IIviLxiSyncArmAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarm.Period Property

Specifies the period of the arm alarm in seconds; that is, the amount of time in seconds that transpire before the alarm repeats.

[Visual Basic]
Public Property Period As Double

[C#]
public Double Period {get; set;}

[C++]
HRESULT get_Period(
    double* retval
);
HRESULT put_Period(
    double val
);

Remarks

A period of zero means there is no repeat.

See Also

IIviLxiSyncArmAlarm Interface | IIviLxiSyncArmAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarm.RepeatCount Property

Specifies the number of times to repeat the trigger at the period specified by the Arm Alarm Period attribute.

[Visual Basic]
Public Property RepeatCount As Int32

[C#]
public Int32 RepeatCount {get; set;}

[C++]
HRESULT get_RepeatCount(
    long* retval
);
HRESULT put_RepeatCount(
    long val
);

Remarks

If Arm Alarm Repeat Count is zero, then the alarm shall repeat forever at the Arm Alarm Period.

See Also

IIviLxiSyncArmAlarm Interface | IIviLxiSyncArmAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarm.TimeFraction Property

Specifies the fractional portion of the time at which the alarm will go off.

[Visual Basic]
Public Property TimeFraction As Double

[C#]
public Double TimeFraction {get; set;}

[C++]
HRESULT get_TimeFraction(
    double* retval
);
HRESULT put_TimeFraction(
    double val
);

Remarks

Note that the actual time of the alarm is the sum of Arm Alarm Time Second and Arm Alarm Time Fraction. The time is specified as the sum of two values because a single double-precision floating-point value does not have sufficient range and resolution to specify the time. Once the alarm goes off, it will repeat at the period set by Arm Alarm Period the number of times set by Arm Alarm Count.

See Also

IIviLxiSyncArmAlarm Interface | IIviLxiSyncArmAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarm.TimeSeconds Property

Specifies the seconds portion of time at which the alarm will go off.

[Visual Basic]
Public Property TimeSeconds As Double

[C#]
public Double TimeSeconds {get; set;}

[C++]
HRESULT get_TimeSeconds(
    double* retval
);
HRESULT put_TimeSeconds(
    double val
);

Remarks

Note that the actual time of the alarm is the sum of Arm Alarm Time Seconds and Arm Alarm Time Fraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. Once the alarm goes off, it will repeat at the period set by Arm Alarm Period the number of times set by Arm Alarm Count.

See Also

IIviLxiSyncArmAlarm Interface | IIviLxiSyncArmAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarms Members

IIviLxiSyncArmAlarms overview

Public Properties

<table>
<thead>
<tr>
<th>Count</th>
<th>The number of available arm alarms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>An interface reference pointer to the ILxiSyncArmAlarm interface which is selected by the alarm name.</td>
</tr>
<tr>
<td>Name</td>
<td>The alarm name for a given index.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Add</th>
<th>This function creates a new arm alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisableAll</td>
<td>This function disables all arm alarms.</td>
</tr>
<tr>
<td>Remove</td>
<td>This function removes an arm alarm.</td>
</tr>
<tr>
<td>RemoveAllCustomArmAlarms</td>
<td>This function removes all of the custom arm alarms that were added using the Add Arm Alarm function.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncArmAlarms Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms Interface

LxiSync repeated capability interface containing methods and properties that apply to all arm alarms defined for the device. A particular arm alarm can be accessed using the Item property.

For a list of all members of this type, see IIviLxiSyncArmAlarms Members.

[Visual Basic]
Public Interface IIviLxiSyncArmAlarms

[C#]
public interface IIviLxiSyncArmAlarms

[C++]
__interface IIviLxiSyncArmAlarms

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarms Methods

The methods of the IIviLxiSyncArmAlarms interface are listed here. For a complete list of IIviLxiSyncArmAlarms interface members, see the IIviLxiSyncArmAlarms Members topic.

Public Methods

- **Add**
  This function creates a new arm alarm

- **DisableAll**
  This function disables all arm alarms.

- **Remove**
  This function removes an arm alarm.

- **RemoveAllCustomArmAlarms**
  This function removes all of the custom arm alarms that were added using the Add Arm Alarm function.

See Also

IIviLxiSyncArmAlarms Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms Properties

The properties of the IIviLxiSyncArmAlarms interface are listed here. For a complete list of IIviLxiSyncArmAlarms interface members, see the IIviLxiSyncArmAlarms Members topic.

Public Properties

- **Count**: The number of available arm alarms.
- **Item**: An interface reference pointer to the ILxiSyncArmAlarm interface which is selected by the alarm name.
- **Name**: The alarm name for a given index.

See Also

IIviLxiSyncArmAlarms Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms.Add Method

This function creates a new arm alarm

[Visual Basic]
Public Sub Add(  
    ByVal AlarmName As String  
)

[C#]
public void Add(  
    String AlarmName  
);

[C++]
HRESULT Add(  
    BSTR AlarmName  
);

Parameters

AlarmName
    Specifies the name of the arm alarm to create.

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms.Count Property

The number of available arm alarms.

[Visual Basic]
Public ReadOnly Property Count As Int32

[C#]
public Int32 Count {get;}

[C++]
HRESULT get_Count(
    long* retval
);

Remarks

The count returned includes the reserved repeated capability name 'ALARM0'.

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmAlarms.DisableAll Method

This function disables all arm alarms.

[Visual Basic]
Public Sub DisableAll( _
)

[C#]
public void DisableAll(_
);

[C++]
HRESULT DisableAll(_
);

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms.Item
Property

An interface reference pointer to the ILxiSyncArmAlarm interface which is selected by the alarm name.

[Visual Basic]
Public ReadOnly Property Item( ByVal Name As String ) As IIviLxiSyncArmAlarm

[C#]
public IIviLxiSyncArmAlarm get_Item( String Name );

[C++]
HRESULT get_Item( BSTR Name, IIviLxiSyncArmAlarm** retval );

Parameters

Name
The name of a alarm. It may be either a name returned by the Name property or a name mapped to a particular repeated capability in the session in the configuration store.

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms.Name Property

The alarm name for a given index.

[Visual Basic]
Public ReadOnly Property Name( _
    ByVal Index As Int32 _
) As String

[C#]
public String get_Name(
    Int32 Index
);

[C++]
HRESULT get_Name(
    long Index,
    BSTR* retval
);

Parameters

Index
    One based index into the collection of alarms.

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmAlarms.Remove Method

This function removes an arm alarm.

[Visual Basic]
Public Sub Remove( _
    ByVal AlarmName As String _
)

[C#]
public void Remove(
    String AlarmName
);

[C++]
HRESULT Remove(
    BSTR AlarmName
);

Parameters

AlarmName
    Specifies the name of the arm alarm to remove.

Remarks

Any resources associated with this alarm should be freed. The AlarmName parameter is case-insensitive.

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members | Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
IIviLxiSyncArmAlarms.RemoveAllCustomArmAlarms Method

This function removes all of the custom arm alarms that were added using the Add Arm Alarm function.

[Visual Basic]
Public Sub RemoveAllCustomArmAlarms(
)

[C#]
public void RemoveAllCustomArmAlarms(
);

[C++]
HRESULT RemoveAllCustomArmAlarms(
);

See Also

IIviLxiSyncArmAlarms Interface | IIviLxiSyncArmAlarms Members |
Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmSource Members

**IIviLxiSyncArmSource overview**

**Public Properties**

- **Detection**
  Specifies the style of arm source detection.

- **Enabled**
  Enables or disables the arm source.

- **EventId**
  Specifies the LAN event identifier that is associated with this arm source. LAN Events with this identifier are accepted from the source described in the filter.

- **Filter**
  Specifies a filter for restricting arm sources.

**Public Methods**

- **Configure**
  This function configures the most commonly configured attributes of the arm source sub-system.

**See Also**

IIviLxiSyncArmSource Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSource Interface

LxiSync arm source interface.

For a list of all members of this type, see IIviLxiSyncArmSource Members.

[Visual Basic]
Public Interface IIviLxiSyncArmSource

[C#]
public interface IIviLxiSyncArmSource

[C++]
__interface IIviLxiSyncArmSource

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncArmSource Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmSource Methods

The methods of the IIviLxiSyncArmSource interface are listed here. For a complete list of IIviLxiSyncArmSource interface members, see the IIviLxiSyncArmSource Members topic.

Public Methods

| Configure | This function configures the most commonly configured attributes of the arm source sub-system. |

See Also

IIviLxiSyncArmSource Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
### IIviLxiSyncArmSource Properties

The properties of the **IIviLxiSyncArmSource** interface are listed here. For a complete list of **IIviLxiSyncArmSource** interface members, see the [IIviLxiSyncArmSource Members](#) topic.

#### Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detection</strong></td>
<td>Specifies the style of arm source detection.</td>
</tr>
<tr>
<td><strong>Enabled</strong></td>
<td>Enables or disables the arm source.</td>
</tr>
<tr>
<td><strong>EventId</strong></td>
<td>Specifies the LAN event identifier that is associated with this arm source. LAN Events with this identifier are accepted from the source described in the filter.</td>
</tr>
<tr>
<td><strong>Filter</strong></td>
<td>Specifies a filter for restricting arm sources.</td>
</tr>
</tbody>
</table>

#### See Also

[IIviLxiSyncArmSource Interface](#) | [Ivi.LxiSync.Interop Namespace](#)

---

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncArmSource.Configure Method

This function configures the most commonly configured attributes of the arm source sub-system.

[Visual Basic]
Public Sub Configure(  
    ByVal Enabled As Boolean,  
    ByVal Detection As IviLxiSyncArmSourceDetectionEnum  
)

[C#]
public void Configure(  
    Boolean Enabled,  
    IviLxiSyncArmSourceDetectionEnum Detection  
);

[C++]
HRESULT Configure(  
    VARIANT_BOOL Enabled,  
    IviLxiSyncArmSourceDetectionEnum Detection  
);

Parameters

Enabled
   Enables or disables the arm source.
Detection
   Specifies the style of the arm source.

See Also

IIviLxiSyncArmSource Interface | IIviLxiSyncArmSource Members | Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
IIviLxiSyncArmSource.EventId Property

Specifies the LAN event identifier that is associated with this arm source. LAN Events with this identifier are accepted from the source described in the filter.

[Visual Basic]
Public Property EventId As String

[C#]
public String EventId {get; set;}

[C++]
HRESULT get_EventId(
    BSTR* retval
);  
HRESULT put_EventId(
    BSTR val
);

Remarks

The default value for Event ID is the repeated capability specifier for this arm source.

See Also

IIviLxiSyncArmSource Interface | IIviLxiSyncArmSource Members | Ivi.LxiSync.Interop Namespace
IIviLxiSyncArmSource.Filter Property

Specifies a filter for restricting arm sources.

[Visual Basic]
Public Property Filter As String

[C#]
public String Filter {get; set;}

[C++]
HRESULT get_Filter(
    BSTR* retval
);
HRESULT put_Filter(
    BSTR val
);

Remarks

The filter specified by this attribute denotes the accepted sources.

See Also

IIviLxiSyncArmSource Interface | IIviLxiSyncArmSource Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources Members

IIviLxiSyncArmSources overview

Public Properties

<table>
<thead>
<tr>
<th></th>
<th>Count</th>
<th>The number of sources.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Item</td>
<td>An interface reference pointer to the ILxiSyncArmSource interface which is selected by the source name.</td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>The source name for a given index.</td>
</tr>
<tr>
<td></td>
<td>OrEnabled</td>
<td>Enables or disables the OR-summing of the arm sources.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th></th>
<th>Add</th>
<th>This function creates a new arm source.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DisableAll</td>
<td>This function disables all arm sources.</td>
</tr>
<tr>
<td></td>
<td>Remove</td>
<td>This function removes an arm source.</td>
</tr>
<tr>
<td></td>
<td>RemoveAllCustomArmSources</td>
<td>This function removes all of the custom arm sources that were added using the Add Arm Source function.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncArmSources Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources Interface

LxiSync repeated capability interface containing methods and properties that apply to all arm sources defined for the device. A particular arm source can be accessed using the Item property.

For a list of all members of this type, see IIviLxiSyncArmSources Members.

[Visual Basic]
Public Interface IIviLxiSyncArmSources

[C#]
public interface IIviLxiSyncArmSources

[C++]
__interface IIviLxiSyncArmSources

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources Methods

The methods of the IIviLxiSyncArmSources interface are listed here. For a complete list of IIviLxiSyncArmSources interface members, see the IIviLxiSyncArmSources Members topic.

Public Methods

Add
This function creates a new arm source.

DisableAll
This function disables all arm sources.

Remove
This function removes an arm source.

RemoveAllCustomArmSources
This function removes all of the custom arm sources that were added using the Add Arm Source function.

See Also

IIviLxiSyncArmSources Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources Properties

The properties of the **IIviLxiSyncArmSources** interface are listed here. For a complete list of **IIviLxiSyncArmSources** interface members, see the **IIviLxiSyncArmSources Members** topic.

## Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The number of sources.</td>
</tr>
<tr>
<td>Item</td>
<td>An interface reference pointer to the ILxiSyncArmSource interface which is selected by the source name.</td>
</tr>
<tr>
<td>Name</td>
<td>The source name for a given index.</td>
</tr>
<tr>
<td>OrEnabled</td>
<td>Enables or disables the OR-summing of the arm sources.</td>
</tr>
</tbody>
</table>

**See Also**

- IIviLxiSyncArmSources Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources.Add Method

This function creates a new arm source.

[Visual Basic]
Public Sub Add( _
    ByVal SourceName As String _
)

[C#]
public void Add(  
    String SourceName 
);

[C++]
HRESULT Add(  
    BSTR SourceName 
);

Parameters

*SourceName*

Specifies the name of the arm source to create.

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources.Count Property

The number of sources.

[Visual Basic]
Public ReadOnly Property Count As Int32

[C#]
public Int32 Count {get;}

[C++]
HRESULT get_Count(
    long* retval
);

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources.Item Property

An interface reference pointer to the ILxiSyncArmSource interface which is selected by the source name.

[Visual Basic]
Public ReadOnly Property Item( _
    ByVal Name As String _
) As IIviLxiSyncArmSource

[C#]
public IIviLxiSyncArmSource get_Item(
    String Name
);

[C++]
HRESULT get_Item(
    BSTR Name,
    IIviLxiSyncArmSource** retval
);

Parameters

Name
The name of a source. It may be either a name returned by the Name property or a name mapped to a particular repeated capability in the session in the configuration store.

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncArmSources.Name Property**

The source name for a given index.

**[Visual Basic]**

```vbnet
Public ReadOnly Property Name( ByVal Index As Int32 ) As String
```

**[C#]**

```csharp
public String get_Name( Int32 Index );
```

**[C++]**

```csharp
HRESULT get_Name( long Index, BSTR* retval );
```

**Parameters**

*Index*

One based index into the collection of sources.

**See Also**

[IIviLxiSyncArmSources Interface](#) | [IIviLxiSyncArmSources Members](#) | [Ivi.LxiSync.Interop Namespace](#)

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources.OrEnabled Property

Enables or disables the OR-summing of the arm sources.

[Visual Basic]
Public Property OrEnabled As Boolean

[C#]
public Boolean OrEnabled {get; set;}

[C++]
HRESULT get_OrEnabled(
    VARIANT_BOOL* retval
);
HRESULT put_OrEnabled(
    VARIANT_BOOL val
);

Remarks

When set to True, the arm sources use OR-summing. When set to False, the arm sources use AND-summing.

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncArmSources.Remove Method

This function removes an arm source.

[Visual Basic]
Public Sub Remove( _
    ByVal SourceName As String _
)

[C#]
public void Remove(  
    String SourceName        
);

[C++]
HRESULT Remove(  
    BSTR SourceName        
);

Parameters

SourceName
    Specifies the name of the arm source to remove.

Remarks

Any resources associated with this alarm should be freed. The SourceName parameter is case-insensitive.

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
IIviLxiSyncArmSources.RemoveAllCustomArmSources Method

This function removes all of the custom arm sources that were added using the Add Arm Source function.

[Visual Basic]
Public Sub RemoveAllCustomArmSources()

[C#]
public void RemoveAllCustomArmSources();

[C++]
HRESULT RemoveAllCustomArmSources();

See Also

IIviLxiSyncArmSources Interface | IIviLxiSyncArmSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent Members

IIviLxiSyncEvent overview

Public Properties

- **DestinationPath**
  Specifies a list of places to send the event.

- **DriveMode**
  Specifies how this event is transmitted.

- **Slope**
  Specifies the slope of the inbound event that will cause the generation of an outbound event.

- **Source**
  Specifies the signal which causes an event to be transmitted.

Public Methods

- **Configure**
  This function configures the most commonly configured attributes of the event sub-system.

See Also

IIviLxiSyncEvent Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent Interface

LxiSync event interface.

For a list of all members of this type, see IIviLxiSyncEvent Members.

[Visual Basic]
Public Interface IIviLxiSyncEvent

[C#]
public interface IIviLxiSyncEvent

[C++]
__interface IIviLxiSyncEvent

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncEvent Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncEvent Methods

The methods of the IIviLxiSyncEvent interface are listed here. For a complete list of IIviLxiSyncEvent interface members, see the IIviLxiSyncEvent Members topic.

Public Methods

Configure

This function configures the most commonly configured attributes of the event sub-system.

See Also

IIviLxiSyncEvent Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent Properties

The properties of the IIviLxiSyncEvent interface are listed here. For a complete list of IIviLxiSyncEvent interface members, see the IIviLxiSyncEvent Members topic.

Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestinationPath</td>
<td>Specifies a list of places to send the event.</td>
</tr>
<tr>
<td>DriveMode</td>
<td>Specifies how this event is transmitted.</td>
</tr>
<tr>
<td>Slope</td>
<td>Specifies the slope of the inbound event that will cause the generation of an outbound event.</td>
</tr>
<tr>
<td>Source</td>
<td>Specifies the signal which causes an event to be transmitted.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncEvent Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent.Configure Method

This function configures the most commonly configured attributes of the event sub-system.

[Visual Basic]
Public Sub Configure( _
    ByVal DriveMode As IviLxiSyncEventDriveModeEnum, _
    ByVal Source As String, _
    ByVal DestinationPath As String, _
    ByVal Slope As IviLxiSyncSourceSlopeEnum _
)

[C#]
public void Configure(  
    IviLxiSyncEventDriveModeEnum DriveMode,  
    String Source,  
    String DestinationPath,  
    IviLxiSyncSourceSlopeEnum Slope  
);

[C++]
HRESULT Configure(  
    IviLxiSyncEventDriveModeEnum DriveMode,  
    BSTR Source,  
    BSTR DestinationPath,  
    IviLxiSyncSourceSlopeEnum Slope  
);

Parameters

DriveMode
    Specifies the mode of the event.
Source
    Specifies the signal which causes an event to be transmitted.
DestinationPath
    Specifies a list of places to send the event.
Slope
    Specifies the slope of the event signal.
IviLxiSync Reference
IIviLxiSyncEvent.Slope Property

Specifies the slope of the inbound event that will cause the generation of an outbound event.

[Visual Basic]
Public Property Slope As IviLxiSyncSourceSlopeEnum

[C#]
public IviLxiSyncSourceSlopeEnum Slope {get; set;}

[C++]
HRESULT get_Slope(
    IviLxiSyncSourceSlopeEnum* retval
);
HRESULT put_Slope(
    IviLxiSyncSourceSlopeEnum val
);

Remarks

The outbound event shall be transmitted with the same slope as the inbound event.

See Also

IIviLxiSyncEvent Interface | IIviLxiSyncEvent Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvent.Source Property

Specifies the signal which causes an event to be transmitted.

[Visual Basic]
Public Property Source As String

[C#]
public String Source {get; set;}

[C++]
HRESULT get_Source(
    BSTR* retval
);
HRESULT put_Source(
    BSTR val
);

Remarks

This attribute is case-insensitive but case-preserving.

See Also

IIviLxiSyncEvent Interface | IIviLxiSyncEvent Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEventLog Members

IIviLxiSyncEventLog overview

Public Properties

<table>
<thead>
<tr>
<th>Enabled</th>
<th>Enables or disables the event logging feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryCount</td>
<td>Returns the number of event log entries available.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Clear</th>
<th>This function removes all existing entries from the event log.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetNextEntry</td>
<td>This function retrieves and clears the oldest event log entry for the IVI session.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncEventLog Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEventLog Interface

LxiSync event log interface.

For a list of all members of this type, see IIviLxiSyncEventLog Members.

[Visual Basic]
Public Interface IIviLxiSyncEventLog

[C#]
public interface IIviLxiSyncEventLog

[C++]
__interface IIviLxiSyncEventLog

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncEventLog Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLXiSync Reference
IIviLxiSyncEventLog Methods

The methods of the IIviLxiSyncEventLog interface are listed here. For a complete list of IIviLxiSyncEventLog interface members, see the IIviLxiSyncEventLog Members topic.

Public Methods

Clear

This function removes all existing entries from the event log.

GetNextEntry

This function retrieves and clears the oldest event log entry for the IVI session.

See Also

IIviLxiSyncEventLog Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEventLog Properties

The properties of the IIviLxiSyncEventLog interface are listed here. For a complete list of IIviLxiSyncEventLog interface members, see the IIviLxiSyncEventLog Members topic.

Public Properties

<table>
<thead>
<tr>
<th>Enabled</th>
<th>Enables or disables the event logging feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryCount</td>
<td>Returns the number of event log entries available.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncEventLog Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEventLog.Enabled Property

Enables or disables the event logging feature.

[Visual Basic]
Public Property Enabled As Boolean

[C#]
public Boolean Enabled {get; set;}

[C++]
HRESULT get_Enabled(
    VARIANT_BOOL* retval
);
HRESULT put_Enabled(
    VARIANT_BOOL val
);

See Also

IIviLxiSyncEventLog Interface | IIviLxiSyncEventLog Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncEvents Members

IIviLxiSyncEvents overview

Public Properties

| Count          | The number of available events. |
| Item           | An interface reference pointer to the ILxiSyncEvent interface which is selected by the event name. |
| Name           | The event name for a given index. |
| WiredOrBiasMode | Specifies whether the LXI device will serve as the wired-OR bias for specific LXI trigger bus lines. |

Public Methods

| Add            | This function creates a new event. |
| DisableAll     | This function disables all events. |
| Remove         | This function removes an event. |
| RemoveAllCustomEvents | This function removes all of the custom events that were added using the Add Event function. |

See Also

IIviLxiSyncEvents Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents Interface

LxiSync repeated capability interface containing methods and properties that apply to all events defined for the device. A particular event can be accessed using the Item property.

For a list of all members of this type, see IIviLxiSyncEvents Members.

[Visual Basic]
Public Interface IIviLxiSyncEvents

[C#]
public interface IIviLxiSyncEvents

[C++]
__interface IIviLxiSyncEvents

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents Methods

The methods of the IIviLxiSyncEvents interface are listed here. For a complete list of IIviLxiSyncEvents interface members, see the IIviLxiSyncEvents Members topic.

Public Methods

- Add
  This function creates a new event.
- DisableAll
  This function disables all events.
- Remove
  This function removes an event.
- RemoveAllCustomEvents
  This function removes all of the custom events that were added using the Add Event function.

See Also

IIviLxiSyncEvents Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents Properties

The properties of the **IIviLxiSyncEvents** interface are listed here. For a complete list of **IIviLxiSyncEvents** interface members, see the **IIviLxiSyncEvents Members** topic.

**Public Properties**

- **Count**
  The number of available events.

- **Item**
  An interface reference pointer to the **ILxiSyncEvent** interface which is selected by the event name.

- **Name**
  The event name for a given index.

- **WiredOrBiasMode**
  Specifies whether the LXI device will serve as the wired-OR bias for specific LXI trigger bus lines.

**See Also**

IIviLxiSyncEvents Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents.Add Method

This function creates a new event.

[Visual Basic]
Public Sub Add( _
    ByVal EventName As String _
)

[C#]
public void Add(
    String eventName
);

[C++]
HRESULT Add(
    BSTR eventName
);

Parameters

EventName
    Specifies the name of the event to create.

Remarks

The EventName parameter is case-insensitive but case-preserving. This means that any casing of the EventName parameter can be used to access the event within the repeated capability collection, but the original casing is used by the specific driver when transmitting events on the LAN.

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
IIviLxiSyncEvents.Count Property

The number of available events.

[Visual Basic]
Public ReadOnly Property Count As Int32

[C#]
public Int32 Count {get;}

[C++]
HRESULT get_Count(
    long* retval
);

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncEvents.DisableAll Method

This function disables all events.

[Visual Basic]
Public Sub DisableAll(_
)

[C#]
public void DisableAll(_
);

[C++]
HRESULT DisableAll(_
);

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents.Item Property

An interface reference pointer to the ILxiSyncEvent interface which is selected by the event name.

[Visual Basic]
Public ReadOnly Property Item( _
    ByVal Name As String _
) As IIviLxiSyncEvent

[C#]
public IIviLxiSyncEvent get_Item(
    String Name
);

[C++]
HRESULT get_Item(
    BSTR Name,
    IIviLxiSyncEvent** retval
);

Parameters

Name
The name of an event. It may be either a name returned by the Name property or a name mapped to a particular repeated capability in the session in the configuration store.

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents.Name Property

The event name for a given index.

[Visual Basic]
Public Readonly Property Name(_
    ByVal Index As Int32 _
) As String

[C#]
public String get_Name(
    Int32 Index
);

[C++]
HRESULT get_Name(
    long Index,
    BSTR* retval
);

Parameters

Index
    One based index into the collection of events.

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncEvents.Remove Method

This function removes an event.

[Visual Basic]
Public Sub Remove( _
    ByVal EventName As String _
)

[C#]
public void Remove(
    String EventName
);

[C++]
HRESULT Remove(
    BSTR EventName
);

Parameters

EventName
    Specifies the name of the event to remove.

Remarks

The EventName parameter is case-insensitive.

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents.RemoveAllCustomEvents Method

This function removes all of the custom events that were added using the Add Event function.

[Visual Basic]
Public Sub RemoveAllCustomEvents( )

[C#]
public void RemoveAllCustomEvents();

[C++]
HRESULT RemoveAllCustomEvents();

Remarks

The events associated with the reserved repeated capability identifiers are not affected by this function.

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncEvents.WiredOrBiasMode Property

Specifies whether the LXI device will serve as the wired-OR bias for specific LXI trigger bus lines.

[Visual Basic]
Public Property WiredOrBiasMode As Int32

[C#]
public Int32 WiredOrBiasMode {get; set;}

[C++]
HRESULT get_WiredOrBiasMode(
    long* retval
);
HRESULT put_WiredOrBiasMode(
    long val
);

Remarks

The allowed values for this attribute are 0 to 255. This attribute is a bit field, where bit 0 represents LXI0, bit 1 represents LXI1, and so on. A value of one in a particular bit indicates that the LXI device shall serve as the bias for the corresponding trigger bus line. A value of zero in a particular bit disables the bias for the corresponding trigger bus line. To use a trigger bus line in driven mode, the bias must be disabled. Enabling wired-OR bias has no impact on the device’s ability to either respond to signals on trigger bus lines or to send events on trigger bus lines. One and only one LXI device can serve as the wired-OR bias for a particular trigger bus line, although different devices can serve as the wired-OR bias for different trigger bus lines.

See Also

IIviLxiSyncEvents Interface | IIviLxiSyncEvents Members | Ivi.LxiSync.Interop
Namespace
IviLxiSync Reference
**IIviLxiSyncTime Members**

**IIviLxiSyncTime overview**

**Public Properties**

- **IsMaster**
  Indicates if this device is the 1588 master.

- **IsSynchronized**
  Indicates if the device is synchronized.

**Public Methods**

- **GetSystemTime**
  This function retrieves the current 1588 time.

**See Also**

- IIviLxiSyncTime Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTime Interface

LxiSync time interface.

For a list of all members of this type, see IIviLxiSyncTime Members.

[Visual Basic]
Public Interface IIviLxiSyncTime

[C#]
public interface IIviLxiSyncTime

[C++]
__interface IIviLxiSyncTime

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncTime Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTime Methods

The methods of the IIviLxiSyncTime interface are listed here. For a complete list of IIviLxiSyncTime interface members, see the IIviLxiSyncTime Members topic.

**Public Methods**

GetSystemTime

This function retrieves the current 1588 time.

**See Also**

IIviLxiSyncTime Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTime Properties

The properties of the IIviLxiSyncTime interface are listed here. For a complete list of IIviLxiSyncTime interface members, see the IIviLxiSyncTime Members topic.

Public Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsMaster</td>
<td>Indicates if this device is the 1588 master.</td>
</tr>
<tr>
<td>IsSynchronized</td>
<td>Indicates if the device is synchronized.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncTime Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTime.IsMaster Property

Indicates if this device is the 1588 master.

[Visual Basic]
Public ReadOnly Property IsMaster As Boolean

[C#]
public Boolean IsMaster {get;}

[C++]
HRESULT get_IsMaster(
    VARIANT_BOOL* retval
);

See Also

IIviLxiSyncTime_Interface | IIviLxiSyncTime_Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTime.IsSynchronized Property

Indicates if the device is synchronized.

[Visual Basic]
Public Readonly Property IsSynchronized As Boolean

[C#]
public Boolean IsSynchronized {get;}

[C++]
HRESULT get_IsSynchronized(
    VARIANT_BOOL* retval
);

See Also

IIviLxiSyncTime Interface | IIviLxiSyncTime Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTrigger Members

IIviLxiSyncTrigger overview

Public Properties

- **Alarms**: Pointer to the ILxiSyncTriggerAlarms interface.
- **Sources**: Pointer to the ILxiSyncTriggerSources interface.
- **TriggerCount**: Specifies the number of times a trigger has to occur to complete a measurement; that is, the number of triggers that are accepted before the measurement must be armed again.
- **TriggerSource**: Specifies which of the available trigger sources to use as the signal for triggering the device-specific operation (for example, a measurement).

See Also

IIviLxiSyncTrigger Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTrigger Interface

LxiSync trigger interface.

For a list of all members of this type, see IIviLxiSyncTrigger Members.

[Visual Basic]
Public Interface IIviLxiSyncTrigger

[C#]
public interface IIviLxiSyncTrigger

[C++]
__interface IIviLxiSyncTrigger

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncTrigger Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTrigger Properties

The properties of the IIviLxiSyncTrigger interface are listed here. For a complete list of IIviLxiSyncTrigger interface members, see the IIviLxiSyncTrigger Members topic.

Public Properties

<table>
<thead>
<tr>
<th>Alarms</th>
<th>Pointer to the ILxiSyncTriggerAlarms interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sources</td>
<td>Pointer to the ILxiSyncTriggerSources interface.</td>
</tr>
<tr>
<td>TriggerCount</td>
<td>Specifies the number of times a trigger has to occur to complete a measurement; that is, the number of triggers that are accepted before the measurement must be armed again.</td>
</tr>
<tr>
<td>TriggerSource</td>
<td>Specifies which of the available trigger sources to use as the signal for triggering the device-specific operation (for example, a measurement).</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncTrigger Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTrigger.Sources Property

Pointer to the ILxiSyncTriggerSources interface.

[Visual Basic]
Public ReadOnly Property Sources As IIviLxiSyncTriggerSources

[C#]
public IIviLxiSyncTriggerSources Sources {get;}

[C++]
HRESULT get_Sources(  
    IIviLxiSyncTriggerSources** retval
);

See Also

IIviLxiSyncTrigger Interface | IIviLxiSyncTrigger Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTrigger.TriggerCount Property

Specifies the number of times a trigger has to occur to complete a measurement; that is, the number of triggers that are accepted before the measurement must be armed again.

[Visual Basic]
Public Property TriggerCount As Int32

[C#]
public Int32 TriggerCount {get; set;}

[C++]
HRESULT get_TriggerCount(
    long* retval
);
HRESULT put_TriggerCount(
    long val
);

See Also

IIviLxiSyncTrigger Interface | IIviLxiSyncTrigger Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTrigger.TriggerSource Property

Specifies which of the available trigger sources to use as the signal for triggering the device-specific operation (for example, a measurement).

[Visual Basic]
Public Property TriggerSource As String

[C#]
public String TriggerSource {get; set;}

[C++]
HRESULT get_TriggerSource(
    BSTR* retval
);    
HRESULT put_TriggerSource(
    BSTR val
);

Remarks

The value specified for this attribute may be one of the names in the LxiTriggerSource repeated capability collection as returned from either the Trigger Source Name attribute (for IVI-COM) or the GetTriggerSourceName function (for IVI-C). The name specified here may also correspond to a non-LXI trigger event. For instance, the caller can use this attribute to program the trigger source to external or immediate triggering, by specifying values such as EXT or INT. Such trigger source names are device-dependent. If the device trigger source has been programmed to a non-LXI event using an attribute or function other than the Trigger Source attribute, then this attribute shall return that value when read. For instance, if the specific driver implements an IVI instrument class and the class-compliant API is used to set the trigger source to external, then this property shall return a string that reflects the value set through the class-compliant API.
See Also

IIviLxiSyncTrigger Interface | IIviLxiSyncTrigger Members | IIvi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTriggerAlarm Members

IIviLxiSyncTriggerAlarm overview

Public Properties

- **Enabled**: Enables or disables the trigger alarm.
- **Period**: Specifies the period of the trigger alarm in seconds; that is, the amount of time in seconds that transpire before the alarm repeats.
- **RepeatCount**: Specifies the number of times to repeat the trigger at the period specified by the Trigger Alarm Period attribute.
- **TimeFraction**: Specifies the fractional seconds portion of the time at which the alarm will go off.
- **TimeSeconds**: Specifies the seconds portion of the time at which the alarm will go off.

Public Methods

- **Configure**: This function configures the most commonly configured attributes of the trigger alarm sub-system.

See Also

IIviLxiSyncTriggerAlarm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm Interface

LxiSync trigger alarm interface.

For a list of all members of this type, see IIviLxiSyncTriggerAlarm Members.

[Visual Basic]
Public Interface IIviLxiSyncTriggerAlarm

[C#]
public interface IIviLxiSyncTriggerAlarm

[C++]
__interface IIviLxiSyncTriggerAlarm

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm Methods

The methods of the IIviLxiSyncTriggerAlarm interface are listed here. For a complete list of IIviLxiSyncTriggerAlarm interface members, see the IIviLxiSyncTriggerAlarm Members topic.

Public Methods

Configure

This function configures the most commonly configured attributes of the trigger alarm sub-system.

See Also

IIviLxiSyncTriggerAlarm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm Properties

The properties of the IIviLxiSyncTriggerAlarm interface are listed here. For a complete list of IIviLxiSyncTriggerAlarm interface members, see the IIviLxiSyncTriggerAlarm Members topic.

Public Properties

- **Enabled**
  Enables or disables the trigger alarm.

- **Period**
  Specifies the period of the trigger alarm in seconds; that is, the amount of time in seconds that transpire before the alarm repeats.

- **RepeatCount**
  Specifies the number of times to repeat the trigger at the period specified by the Trigger Alarm Period attribute.

- **TimeFraction**
  Specifies the fractional seconds portion of the time at which the alarm will go off.

- **TimeSeconds**
  Specifies the seconds portion of the time at which the alarm will go off.

See Also

IIviLxiSyncTriggerAlarm Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm.Configure Method

This function configures the most commonly configured attributes of the trigger alarm sub-system.

[Visual Basic]
Public Sub Configure( _
    ByVal TimeSeconds As Double, _
    ByVal TimeFraction As Double, _
    ByVal Period As Double, _
    ByVal RepeatCount As Int32 _
)

[C#]
public void Configure(  
    Double TimeSeconds,  
    Double TimeFraction,  
    Double Period,  
    Int32 RepeatCount  
);

[C++]
HRESULT Configure(  
    double TimeSeconds,  
    double TimeFraction,  
    double Period,  
    long RepeatCount  
);

Parameters

TimeSeconds
    Specifies the seconds part of 1588 time.
TimeFraction
    Specifies the fractional part of 1588 time.
Period
    Specifies the period of the trigger alarm.
RepeatCount
Specifies the number of times to repeat the trigger at the period specified by the Trigger Alarm Repeat Period attribute.

See Also

IIviLxiSyncTriggerAlarm Interface | IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm.Enabled Property

Enables or disables the trigger alarm.

[Visual Basic]
Public Property Enabled As Boolean

[C#]
public Boolean Enabled {get; set;}

[C++]
HRESULT get_Enabled(
    VARIANT_BOOL* retval
);
HRESULT put_Enabled(
    VARIANT_BOOL val
);

See Also

IIviLxiSyncTriggerAlarm Interface | IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm.Period Property

Specifies the period of the trigger alarm in seconds; that is, the amount of time in seconds that transpire before the alarm repeats.

[Visual Basic]
Public Property Period As Double

[C#]
public Double Period {get; set;}

[C++]
HRESULT get_Period(
    double* retval
);    
HRESULT put_Period(
    double val
);

Remarks

A period of zero means there is no repeat.

See Also

IIviLxiSyncTriggerAlarm Interface | IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm.RepeatCount Property

Specifies the number of times to repeat the trigger at the period specified by the Trigger Alarm Period attribute.

[Visual Basic]
Public Property RepeatCount As Int32

[C#]
public Int32 RepeatCount {get; set;}

[C++]
HRESULT get_RepeatCount(
    long* retval
);
HRESULT put_RepeatCount(
    long val
);

Remarks

If Trigger Alarm Repeat Period is non-zero and Trigger Alarm Repeat Count is zero, then the alarm shall repeat forever at the Trigger Alarm Period.

See Also

IIviLxiSyncTriggerAlarm Interface | IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm.TimeFraction Property

Specifies the fractional seconds portion of the time at which the alarm will go off.

[Visual Basic]
Public Property TimeFraction As Double

[C#]
public Double TimeFraction {get; set;}

[C++]
HRESULT get_TimeFraction(
    double* retval
);
HRESULT put_TimeFraction(
    double val
);

Remarks

Note that the actual time of the alarm is the sum of Trigger Alarm Time Seconds and Trigger Alarm Time Fraction. The time is specified as the sum of two values because a single double-precision floating-point value does not have sufficient range and resolution to specify the time. Once the alarm goes off, it will repeat at the period set by Trigger Alarm Period the number of times set by Trigger Alarm Count.

See Also

IIviLxiSyncTriggerAlarm Interface | IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarm·TimeSeconds
Property

Specifies the seconds portion of the time at which the alarm will go off.

[Visual Basic]
Public Property TimeSeconds As Double

[C#]
public Double TimeSeconds {get; set;}

[C++]
HRESULT get_TimeSeconds(
    double* retval
);
HRESULT put_TimeSeconds(
    double val
);

Remarks

Note that the actual time of the alarm is the sum of Trigger Alarm Time Seconds and Trigger Alarm Time Fraction. The time is specified as the sum of two values because a single double-precision floating-point does not have sufficient range and resolution to specify the time. Once the alarm goes off, it will repeat at the period set by Trigger Alarm Period the number of times set by Trigger Alarm Count.

See Also

IIviLxiSyncTriggerAlarm Interface | IIviLxiSyncTriggerAlarm Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms Members

IIviLxiSyncTriggerAlarms overview

Public Properties

- **Count**
  Returns the number of currently available trigger alarms.

- **Item**
  An interface reference pointer to the ILxiSyncTriggerAlarm interface which is selected by the alarm name.

- **Name**
  Returns the physical repeated capability identifier defined by the specific driver for the trigger alarm that corresponds to the one-based index that the user specifies.

Public Methods

- **Add**
  This function creates a new trigger alarm.

- **DisableAll**
  This function disables all trigger alarms.

- **Remove**
  This function removes a trigger alarm.

- **RemoveAllTriggerAlarms**
  This function removes all of the trigger alarms that were added using the Add Trigger Alarm function.

See Also

IIviLxiSyncTriggerAlarms Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTriggerAlarms Interface

LxiSync repeated capability interface containing methods and properties that apply to all trigger alarms defined for the device. A particular trigger alarm can be accessed using the Item property.

For a list of all members of this type, see IIviLxiSyncTriggerAlarms Members.

[Visual Basic]
Public Interface IIviLxiSyncTriggerAlarms

[C#]
public interface IIviLxiSyncTriggerAlarms

[C++]
__interface IIviLxiSyncTriggerAlarms

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncTriggerAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms Methods

The methods of the **IIviLxiSyncTriggerAlarms** interface are listed here. For a complete list of **IIviLxiSyncTriggerAlarms** interface members, see the **IIviLxiSyncTriggerAlarms Members** topic.

**Public Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>This function creates a new trigger alarm</td>
</tr>
<tr>
<td>DisableAll</td>
<td>This function disables all trigger alarms.</td>
</tr>
<tr>
<td>Remove</td>
<td>This function removes a trigger alarm.</td>
</tr>
<tr>
<td>RemoveAllTriggerAlarms</td>
<td>This function removes all of the trigger alarms that were added using the Add Trigger Alarm function.</td>
</tr>
</tbody>
</table>

**See Also**

[IIviLxiSyncTriggerAlarms Interface] | [Ivi.LxiSync.Interop Namespace]

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms Properties

The properties of the IIviLxiSyncTriggerAlarms interface are listed here. For a complete list of IIviLxiSyncTriggerAlarms interface members, see the IIviLxiSyncTriggerAlarms Members topic.

Public Properties

<table>
<thead>
<tr>
<th>Count</th>
<th>Returns the number of currently available trigger alarms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>An interface reference pointer to the ILxiSyncTriggerAlarm interface which is selected by the alarm name.</td>
</tr>
<tr>
<td>Name</td>
<td>Returns the physical repeated capability identifier defined by the specific driver for the trigger alarm that corresponds to the one-based index that the user specifies</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncTriggerAlarms Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncTriggerAlarms.Add Method**

This function creates a new trigger alarm

[Visual Basic]
```vbnet
Public Sub Add( _
    ByVal AlarmName As String _
)
```

[C#]
```csharp
public void Add(
    String AlarmName
);
```

[C++]
```cpp
HRESULT Add(
    BSTR AlarmName
);
```

**Parameters**

*AlarmName*
Specifies the name of the trigger alarm to create.

**See Also**

IIviLxiSyncTriggerAlarms Interface | IIviLxiSyncTriggerAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms.Count Property

Returns the number of currently available trigger alarms.

[Visual Basic]
Public ReadOnly Property Count As Int32

[C#]
public Int32 Count {get;}

[C++]
HRESULT get_Count(
    long* retval
);

Remarks

The count returned includes the reserved repeated capability named 'ALARM0'.

See Also

IIviLxiSyncTriggerAlarms Interface  |  IIviLxiSyncTriggerAlarms Members  |  Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms.DisableAll Method

This function disables all trigger alarms.

[Visual Basic]
Public Sub DisableAll( _
)

[C#]
public void DisableAll(
);

[C++]
HRESULT DisableAll(  
);

See Also

IIviLxiSyncTriggerAlarms Interface | IIviLxiSyncTriggerAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms.Item Property

An interface reference pointer to the ILxiSyncTriggerAlarm interface which is selected by the alarm name.

[Visual Basic]
Public ReadOnly Property Item(  
   ByVal Name As String  
) As IIviLxiSyncTriggerAlarm

[C#]
public IIviLxiSyncTriggerAlarm get_Item(  
   String Name  
);

[C++]
HRESULT get_Item(  
   BSTR Name,  
   IIviLxiSyncTriggerAlarm** retval  
);

Parameters

Name
The name of a alarm. It may be either a name returned by the Name property or a name mapped to a particular repeated capability in the session in the configuration store.

See Also

IIviLxiSyncTriggerAlarms Interface | IIviLxiSyncTriggerAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncTriggerAlarms.Name Property**

Returns the physical repeated capability identifier defined by the specific driver for the trigger alarm that corresponds to the one-based index that the user specifies.

[Visual Basic]
Public ReadOnly Property Name( _
    ByVal Index As Int32 _
) As String

[C#]
public String get_Name(
    Int32 Index
);

[C++]
HRESULT get_Name(
    long Index, 
    BSTR* retval 
);

**Parameters**

*Index*

One based index into the collection of alarms.

**Remarks**

For custom arm sources added with the Add Trigger Source function, this function returns the arm source name in the original casing used when Add Trigger Source was called.

**See Also**
IviLxiSync Reference
IIviLxiSyncTriggerAlarms.Remove Method

This function removes a trigger alarm.

[Visual Basic]
Public Sub Remove(_
    ByVal AlarmName As String _
)

[C#]
public void Remove(_
    String AlarmName
);

[C++]
HRESULT Remove(_
    BSTR AlarmName
);

Parameters

AlarmName
    Specifies the name of the trigger alarm to remove.

Remarks

The AlarmName parameter is case-insensitive.

See Also

IIviLxiSyncTriggerAlarms Interface | IIviLxiSyncTriggerAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerAlarms.RemoveAllTriggerAlarms Method

This function removes all of the trigger alarms that were added using the Add Trigger Alarm function.

[Visual Basic]
Public Sub RemoveAllTriggerAlarms( _
)

[C#]
public void RemoveAllTriggerAlarms( 
);  

[C++]
HRESULT RemoveAllTriggerAlarms( 
);

See Also

IIviLxiSyncTriggerAlarms Interface | IIviLxiSyncTriggerAlarms Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSource Members

IIviLxiSyncTriggerSource overview

Public Properties

- **Delay**
  Specifies the trigger source delay from when the trigger logic is satisfied until the device specific action (for instance a measurement) is triggered.

- **Detection**
  Specifies the slope of the trigger source.

- **EventId**
  Specifies the LAN event identifier that is associated with this trigger source.

- **Filter**
  Specifies a filter for restricting trigger sources.

Public Methods

- **Configure**
  This function configures the most commonly configured attributes of the trigger source sub-system.

See Also

IIviLxiSyncTriggerSource Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSource Interface

LxiSync trigger source interface.

For a list of all members of this type, see IIviLxiSyncTriggerSource Members.

[Visual Basic]
Public Interface IIviLxiSyncTriggerSource

[C#]
public interface IIviLxiSyncTriggerSource

[C++]
__interface IIviLxiSyncTriggerSource

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncTriggerSource Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTriggerSource Methods

The methods of the **IIviLxiSyncTriggerSource** interface are listed here. For a complete list of **IIviLxiSyncTriggerSource** interface members, see the **IIviLxiSyncTriggerSource Members** topic.

**Public Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure</strong></td>
<td>This function configures the most commonly configured attributes of the trigger source sub-system.</td>
</tr>
</tbody>
</table>

**See Also**

[IIviLxiSyncTriggerSource Interface] | [Ivi.LxiSync.Interop Namespace]

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSource Properties

The properties of the IIviLxiSyncTriggerSource interface are listed here. For a complete list of IIviLxiSyncTriggerSource interface members, see the IIviLxiSyncTriggerSource Members topic.

Public Properties

- **Delay**
  Specifies the trigger source delay from when the trigger logic is satisfied until the device specific action (for instance a measurement) is triggered.

- **Detection**
  Specifies the slope of the trigger source.

- **EventId**
  Specifies the LAN event identifier that is associated with this trigger source.

- **Filter**
  Specifies a filter for restricting trigger sources.

See Also

IIviLxiSyncTriggerSource Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSource.Configure Method

This function configures the most commonly configured attributes of the trigger source sub-system.

[Visual Basic]
Public Sub Configure(ByVal Delay As Double, ByVal Detection As IviLxiSyncTriggerSourceDetectionEnum)

[C#]
public void Configure(Double Delay, IviLxiSyncTriggerSourceDetectionEnum Detection);

[C++]
HRESULT Configure(double Delay, IviLxiSyncTriggerSourceDetectionEnum Detection);

Parameters

Delay
   Specifies the trigger source delay. The units are seconds. A negative value implies pre-trigger acquisition.

Detection
   Specifies the slope of the trigger source.

See Also

IIviLxiSyncTriggerSource Interface | IIviLxiSyncTriggerSource Members | Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
IIviLxiSyncTriggerSource.Detection Property

Specifies the slope of the trigger source.

[Visual Basic]
Public Property Detection As IviLxiSyncTriggerSourceDetectionEnum

[C#]
public IviLxiSyncTriggerSourceDetectionEnum Detection {get; set;}

[C++]
HRESULT get_Detection(IviLxiSyncTriggerSourceDetectionEnum* retval);
HRESULT put_Detection(IviLxiSyncTriggerSourceDetectionEnum val);

Remarks

If the source is a LAN event and the source slope is set to positive, this Trigger repeated capability will be satisfied when the designated LAN packet arrives with a true indication. If the source slope is set to negative, this Trigger repeated capability will be satisfied when a LAN packet arrives with a false indication.

See Also

IIviLxiSyncTriggerSource_Interface | IIviLxiSyncTriggerSource_Members | Ivi.LxiSync.Interop_Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSources Members

IIviLxiSyncTriggerSources overview

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The number of sources.</td>
</tr>
<tr>
<td>Item</td>
<td>An interface reference pointer to the ILxiSyncTriggerSource interface which is selected by the source name.</td>
</tr>
<tr>
<td>Name</td>
<td>The source name for a given index.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>This function creates a new trigger source.</td>
</tr>
<tr>
<td>Remove</td>
<td>This function removes a trigger source.</td>
</tr>
<tr>
<td>RemoveAllCustomTriggerSources</td>
<td>This function removes all of the custom trigger sources that were added using the Add Trigger Source function.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncTriggerSources Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSources Interface

LxiSync repeated capability interface containing methods and properties that apply to all trigger sources defined for the device. A particular trigger source can be accessed using the Item property.

For a list of all members of this type, see IIviLxiSyncTriggerSources Members.

[Visual Basic]
Public Interface IIviLxiSyncTriggerSources

[C#]
public interface IIviLxiSyncTriggerSources

[C++]
__interface IIviLxiSyncTriggerSources

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

IIviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSources Methods

The methods of the IIviLxiSyncTriggerSources interface are listed here. For a complete list of IIviLxiSyncTriggerSources interface members, see the IIviLxiSyncTriggerSources Members topic.

Public Methods

- Add
  This function creates a new trigger source.
- Remove
  This function removes a trigger source.
- RemoveAllCustomTriggerSources
  This function removes all of the custom trigger sources that were added using the Add Trigger Source function.

See Also

IIviLxiSyncTriggerSources Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSources Properties

The properties of the IIviLxiSyncTriggerSources interface are listed here. For a complete list of IIviLxiSyncTriggerSources interface members, see the IIviLxiSyncTriggerSources Members topic.

Public Properties

<table>
<thead>
<tr>
<th>Count</th>
<th>The number of sources.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>An interface reference pointer to the ILxiSyncTriggerSource interface which is selected by the source name.</td>
</tr>
<tr>
<td>Name</td>
<td>The source name for a given index.</td>
</tr>
</tbody>
</table>

See Also

IIviLxiSyncTriggerSources Interface | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IlviLxiSyncTriggerSources.Item Property

An interface reference pointer to the ILxiSyncTriggerSource interface which is selected by the source name.

[Visual Basic]
Public ReadOnly Property Item( _
    ByVal Name As String _
) As IIviLxiSyncTriggerSource

[C#]
public IIviLxiSyncTriggerSource get_Item(
    String Name
);

[C++]
HRESULT get_Item(
    BSTR Name,
    IIviLxiSyncTriggerSource** retval
);

Parameters

Name
The name of a source. It may be either a name returned by the Name property or a name mapped to a particular repeated capability in the session in the configuration store.

See Also

IlviLxiSyncTriggerSources Interface | IlviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
**IIviLxiSyncTriggerSources.Name Property**

The source name for a given index.

[Visual Basic]
```
Public ReadOnly Property Name( ByVal Index As Int32 ) As String
```

[C#]
```
public String getName( Int32 Index );
```

[C++]
```
HRESULT getName( long Index, BSTR* retval );
```

**Parameters**

*Index*

One based index into the collection of sources.

**See Also**

IIviLxiSyncTriggerSources Interface | IIviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IIviLxiSyncTriggerSources.Remove Method

This function removes a trigger source.

[Visual Basic]
Public Sub Remove( _
    ByVal SourceName As String _
)

[C#]
public void Remove(  
    String SourceName
);

[C++]
HRESULT Remove(  
    BSTR SourceName
);

Parameters

SourceName
    Specifies the name of the trigger source to remove.

Remarks

The SourceName parameter is case-insensitive.

See Also

IIviLxiSyncTriggerSources Interface | IIviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IIviLxiSyncTriggerSources.RemoveAllCustomTriggerSources Method

This function removes all of the custom trigger sources that were added using the Add Trigger Source function.

[Visual Basic]
Public Sub RemoveAllCustomTriggerSources( _
)

[C#]
public void RemoveAllCustomTriggerSources(
);

[C++]
HRESULT RemoveAllCustomTriggerSources(
);

See Also

IIviLxiSyncTriggerSources Interface | IIviLxiSyncTriggerSources Members | Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
Installation
Location of Files

The IviLxiSync installer deploys and registers a variety of files on the target computer. The notative <IviInstallDir> is used to denote the directory in which the IVI Shared Components have been installed. By default, this is C:\Program Files\IVI. Substitute your install directory if the IVI Shared Components were installed in an alternate location.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;IviInstallDir&gt;\Bin*.dll</td>
<td>Driver DLL.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Bin\Primary Interop Assemblies*.dll</td>
<td>Interop assembly for using the driver in .NET environments.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Bin\Primary Interop Assemblies*.xml</td>
<td>XML documentation file for providing IntelliSense support in .NET environments.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Include*.h</td>
<td>Driver header file.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Lib\msc*.lib</td>
<td>Microsoft-format import library.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Drivers\IviLxiSync*.fp</td>
<td>Function panel file.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Drivers\IviLxiSync*.sub</td>
<td>Attribute information file.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Drivers\IviLxiSync\IviLxiSync.chm</td>
<td>HTML 1.0 format help file.</td>
</tr>
<tr>
<td>&lt;IviInstallDir&gt;\Drivers\IviLxiSync\IviLxiSync.HxS</td>
<td>HTML 2.0 format help file.</td>
</tr>
</tbody>
</table>

Installation on a System with the .NET Framework

In addition to deploying the files noted above, the IviLxiSync installer also detects if the .NET Framework is installed and, if so, registers the driver's Primary Interop Assembly (PIA) in the .NET Global Assembly Cache (GAC).

**Note** If the .NET Framework is installed on the target computer after the
IviLxiSync installer runs, then the PIA must be registered manually. To do this, simply navigate to the PIA (<IviInstallDir>Bin\Primary Interop Assemblies\dll) and execute the following command: "gacutil -i .dll". You can also simply drag-n-drop the PIA from Windows explorer into the GAC folder (C:\Windows\Assembly).
IVI Shared Components

Before any IVI driver may be installed on a system, the IVI Shared Components must be properly installed. For an introduction to the IVI Shared Components, see the topic IVI Shared Components.

For additional information on the IVI Shared Components, visit the IVI Foundation web site.

Checking the Version of IVI Shared Components

To determine which version of the IVI Shared Components are installed on your system

1. Navigate to <IviInstallDir>\Bin. Typically this is C:\Program Files\IVI\Bin.
2. Locate the file IviSharedComponentVersion.dll.
3. Right-click on that file and choose Properties.
4. In the Properties Window, click the Version tab.
5. The File version field indicates which version of the IVI Shared Components are installed.
Uninstallation

The only proper way to remove the driver from the system is to use the uninstallation program. Simply removing the installed files will not properly uninstall the driver.

To uninstall the driver

1. From the Windows Start menu, choose All Programs and select Control Panel.
2. Double-click Add or Remove Programs.
3. Select the driver and click Remove.

Note In order to prevent loss of important user configuration information between uninstall/reinstall operations, the uninstallation program does not remove any of the following items from the IVI Configuration Store:

- Logical Name
- Driver Session
- Hardware Asset

The only IVI Configuration Store item that is deleted during an uninstall is the Software Module.
For a discussion of IVI Configuration Store entries, such as the Software Module, see Configuring the Driver.

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
Instrument Command Reference

This is the reference for all the instrument commands issued by the driver.

Instrument Commands

<table>
<thead>
<tr>
<th>Instrument Command</th>
<th>Reference</th>
</tr>
</thead>
</table>

© 2006 IVI Foundation. All rights reserved.
The IVI Foundation is an open consortium founded in 1998 to promote standards for programming test instruments. Composed primarily of instrument manufacturers, end-users, software vendors, and system integrators, the Foundation strives to create specifications that govern the development of instrument drivers.

IVI-compliant drivers offer a number of benefits, such as a consistent programming interface, better performance, and, most notably, interchangeability. Interchangeability refers to the ability of an end user to replace one IVI driver in an application program with another IVI driver from a different vendor, without modifying or even recompiling the application program. To achieve interchangeability, IVI driver developers must carefully follow all of the requirements laid out in the IVI Foundation specifications.

The following sections present an overview of some of the most important IVI topics for driver developers. This document does not attempt to reproduce all of the details contained in the IVI specifications. For complete information on IVI and for the most up-to-date versions of all IVI specifications and components, visit the IVI Foundation web site.

In This Section

**IVI Specifications**
- Instrument Class Specifications
- Architecture Specifications

**IVI Shared Components**
- IVI Shared Component Installer
- IVI Configuration Store
- IVI Configuration Server
- COM Session Factory
- IVI-COM Type Libraries
- IVI Event Server

**Types Of IVI Drivers**
**Inherent Capabilities**
Class-Compliant Interfaces
Base Capabilities Group
Extension Groups
Instrument-Specific Interfaces
Repeated Capabilities
  • Accessing Repeated Capabilities
  • Physical Repeated Capability Names Checking
  • Virtual Repeated Capability Names
Standard IVI Driver Features
  • State Caching
  • Range Checking
  • Coercion and Coercion Recording
  • Simulation
IVI Specifications

The primary work product of the IVI Foundation is the IVI specifications. These documents are publicly available on the [IVI Foundation web site](http://ivifoundation.org). There are two sets of IVI specifications -- instrument class specifications and architecture specifications.

**Instrument Class Specifications**

The IVI Foundation categorizes test instrumentation according to *instrument class*. Each IVI instrument class specification describes the standard functionality required of IVI drivers that want to be compliant with that IVI instrument class. Currently, the IVI Foundation has defined and approved specifications for eight instrument classes.

- **IVI-4.1**: IviScope Class Specification
- **IVI-4.2**: IviDmm Class Specification
- **IVI-4.3**: IviFgen Class Specification
- **IVI-4.4**: IviDCPwr Class Specification
- **IVI-4.6**: IviSwtch Class Specification
- **IVI-4.7**: IviPwrMeter Class Specification
- **IVI-4.8**: IviSpecAn Class Specification
- **IVI-4.10**: IviRFSigGen Class Specification

Each class specification prescribes methods, properties, errors, and other behaviors that IVI drivers must implement if they can claim to be *class compliant*. IVI drivers can support more than one instrument class, or they may not support an instrument class at all. That is to say, a driver may correctly claim IVI compliance without being class-compliant. All IVI drivers, at a minimum, must support a base level of functionality and must follow the IVI architecture. These fundamental rules for building IVI drivers are established by another series of IVI documents known as the [IVI architecture specifications](http://ivifoundation.org).

**Architecture Specifications**

In order to be considered an IVI-compliant driver, irrespective of instrument class, a driver must be implemented according to the rules and guidelines
expressed in the IVI architecture specifications. The following documents comprise the IVI architecture specifications.

- **IVI-3.1: Driver Architecture Specification**
- **IVI-3.2: Inherent Capabilities Specification**
- **IVI-3.3: Standard Cross-Class Capabilities Specification**
- **IVI-3.4: API Style Guide**
- **IVI-3.5: Configuration Server Specification**
- **IVI-3.6: COM Session Factory Specification**
- **IVI-3.7: IVI Event Server Specification**
- **IVI-3.9: C Shared Components Specification**
- **IVI-3.10: Measurement and Stimulus Subsystem Specification**
- **IVI-3.12: C Shared Components Specification**
- **IVI-3.12: Floating Point Services Specification**
- **IVI-5.0: Glossary**

Collectively, the architecture specifications describe such things as common features and capabilities of IVI drivers, required APIs drivers must provide, interface design guidelines, and configuration and deployment requirements.
IVI Shared Components

In order to facilitate IVI driver development and to ensure a consistent experience for end users building IVI-based test systems, the IVI Foundation develops and maintains a series of freely available software components. These components are collectively referred to as the IVI Shared Components, and they must be installed on any computer that will use IVI drivers. In fact, IVI driver installation programs are required to check for the presence of the IVI Shared Components before installing.

The following sections provide a brief overview of some of the most important IVI Shared Components.

**IVI Shared Component Installer**

The IVI Foundation provides an installer for the IVI Shared Components. Moreover, the Foundation's Shared Component installer is the only IVI-compliant way to install the IVI Shared Components. Third parties are not allowed to develop custom installation programs for the IVI Shared Components.

The IVI Shared Component installer creates a directory structure to house the IVI Shared Components as well as IVI drivers themselves. The root of this directory structure is referred to as the IVI install directory (<IviInstallDir>) and is typically <ProgramFilesDir>\IVI.

The IVI Shared Component installer is available for download from the IVI Foundation web site.

**IVI Configuration Store**

When the IVI Shared Components are installed, a special XML file named IviConfigurationStore.xml is placed in the <IviInstallDir>\Data directory. This file is known as the IVI Configuration Store and serves as a central repository for IVI driver configuration information.

IVI driver installers are required to populate the Configuration Store with
information such as the instrument classes supported, the instrument models for which the driver is designed, and the physical channel names understood by the driver. End users add information to the Configuration Store to describe how they intend to use IVI drivers in their system. End users can specify logical names for IVI drivers and virtual names for channels.

For more information on the structure of the IVI Configuration Store, see IVI-3.5: Configuration Server Specification available on the IVI Foundation web site.

**IVI Configuration Server**

The IVI Foundation discourages users and driver developers from directly manipulating the Configuration Store XML file (IviConfigurationStore.xml). The Configuration Store is highly self-referential and easily corrupted if modified directly. Thus, the Foundation provides a COM component known as the IVI Configuration Server for programmatically reading and writing information to the Configuration Store.

The IVI Configuration Server is installed automatically by the IVI Shared Component installer. For more information on how to use the IVI Configuration Server, see IVI-3.5: Configuration Server Specification available on the IVI Foundation web site.

**COM Session Factory**

IVI-COM drivers, like all COM components, must be created using a specific, hard-coded reference to the main COM class. In Visual Basic, this reference takes the form of a class data type, while in Visual C++, a CLSID must be specified. Irrespective of the IDE in use, this identifier must somehow be provided to instantiate the driver and begin calling methods and properties. Yet, having a direct reference to such an instrument-specific item in an application program renders that application non-interchangeble. Specifically, if the user wishes to use a different instrument (and, hence, a different driver), then this CLSID (or class name in Visual Basic) must be manually changed and the program re-complied and re-linked.

The COM Session Factory (or, simply, the session factory) is a very simple COM component supplied with the IVI Shared Components. In spite of its simplicity,
the session factory is a key to writing interchangeable applications that use IVI-COM drivers.

Rather than directly creating an IVI-COM driver using a CLSID (or the equivalent), the application program creates an instance of the session factory and then asks it to instantiate a specific IVI-COM driver. For this to work, the user must create a Logical Name in the IVI Configuration Store and associate that Logical Name with the specific IVI-COM driver they wish to instantiate. The COM Session Factory simply looks up the Logical Name in the IVI Configuration Store, locates the associated IVI-COM driver and instantiates it accordingly.

Note that, as the name implies, the COM Session Factory is only relevant for IVI-COM drivers. IVI-C drivers use a completely different mechanism (namely, IVI Class Drivers), to facilitate interchangeability.

The COM Session Factory is installed automatically by the IVI Shared Component installer. For more information on how to use the IVI Session Factory, see IVI-3.6: COM Session Factory Specification available on the IVI Foundation web site.

**IVI-COM Type Libraries**

Part of the IVI Shared Components include a series of COM type libraries that contain the interface definitions for the IVI inherent capabilities as well as for all IVI instrument classes. These type libraries are installed in <IviInstallDir>\Bin and are generally named <IviClass>TypeLib.dll. For instance, the type library containing the interface definitions for the IviDmm instrument class is IviDmmTypeLib.dll.

Client applications must directly reference the IVI-COM type libraries from their projects in order to call methods and properties in IVI-defined interfaces. All client applications must include a reference to IviDriverTypeLib.dll, since this type library houses the interface definitions for the IVI inherent capabilities. Application programs must also reference the type library containing the interface definitions for any class-compliant interfaces they wish to support.

**IVI Event Server**
The IVI Foundation supplies a software component known as the IVI Event Server for communicating between IVI drivers (and other components) in a test system. Using the Event Server, IVI drivers may send status events, warning events, debug events, and other types of events, to other IVI drivers in the same process, in a different process, or on a different machine on the network.

The IVI Event Server is installed automatically by the IVI Shared Component installer. For more information on how to use the IVI Event Server, see IVI-3.7: Event Server Specification available on the IVI Foundation web site.
Types of IVI Drivers

The IVI Foundation defines two fundamental types of IVI drivers -- IVI-COM drivers and IVI-C drivers. IVI-COM drivers are bonafide COM components that use standard COM interfaces to expose both IVI-defined and instrument-specific functionality. IVI-C drivers are simple DLLs that export C-based entry points.

IVI-COM drivers reference the standard IVI-COM type libraries to implement IVI-defined functionality while IVI-C drivers use standard include (.h) files to define function signatures and attribute values. While IVI-COM drivers offer interchangeability directly, through the use of class-compliant type libraries, IVI-C drivers require a special class driver to achieve interchangeability. This class driver serves as a level of indirection between the client program and the specific IVI driver in question. Application programs that want to allow for interchangeability of IVI-C drivers must only reference the class driver. Application programs that want to allow for interchangeability of IVI-COM drivers must only reference the IVI-COM type libraries. While the IVI-COM type libraries are freely available from the IVI Foundation, class drivers are only available as products from third parties.
**Inherent Capabilities**

All IVI drivers must support a fundamental set of methods and properties. These methods and properties are known as *inherent capabilities*, and address such common functionality as driver initialization and shutdown, driver identification, and error reporting. The details of the IVI inherent capabilities are laid out in *IVI-3.2: Inherent Capabilities Specification*.

IVI-COM drivers provide support for the inherent capabilities by referencing the standard IVI-COM type library IviDriverTypeLib.dll, located in `<IviInstallDir>\Bin`. 
Class-Compliant Interfaces

The set of IVI-COM interfaces the IVI Foundation defines for a particular instrument class, such as the IviDmm instrument class, are known as *class-compliant interfaces*. Class-compliant interfaces are the fundamental mechanism by which IVI drivers achieve interchangeability. Since the definition of class-compliant interfaces is fixed by the IVI Foundation, end users programming against these interfaces can substitute one class-compliant IVI driver for another, without modifying or recompiling their application program.

There is one IVI specification for each class-compliant interface. For example, *IVI-4.2: IviDmm Class Specification* describes the syntax and semantics for the IviDmm instrument class.

The class-compliant interface definitions are packaged into IVI-COM type libraries and installed as part of the IVI Shared Components. For instance, the class-compliant interfaces for the IviDmm instrument class are located in the file `<IviInstallDir>\Bin\IviDmmTypeLib.dll`. IVI-COM drivers that wish to provide support for a particular instrument class must reference the appropriate IVI-COM type library.

Each IVI class specification partitions IVI-defined functionality into two groups -- *base capabilities* and *extension groups*.

**Base Capabilities Group**

The Base Capabilities group contains functionality that *all* IVI drivers must implement if they wish to claim conformance with the instrument class in question. Typically, the Base Capabilities group represents the "least common denominator" of functionality for an instrument class, so most commercial instruments can be expected to support this level of capability.

As an example, the IviDmm Base Capabilities group contains functions for range, resolution, and triggering. Most commercial DMMs possess at least these capabilities, so any IVI driver that wishes to claim IviDmm-compliance must support this group of functionality.
Extension Groups

IVI uses Extension Groups to define functionality that is found in many instruments of a particular class but may not be as common as the functionality expressed in the Base Capabilities. One may think of Extension Groups as being standard definitions for "optional" functionality in an IVI driver.

Extension Groups provide a means for more advanced instruments to expose functionality in an interchangeable fashion, while still allowing simpler instruments to be class-compliant. For instance, not all DMMs have the ability to perform temperature measurements. Thus, the IviDmm class specification uses the IviDmmTemperature Extension Group to define temperature measurement capabilities.

Simply put, an Extension Group is making the statement to the driver developer, "You do not have to provide these capabilities in your IVI driver, but if you do, you must do it according to the IVI definition."
Instrument-Specific Interfaces

Modern instruments typically possess much more functionality than is represented in the IVI instrument class specifications. End users must have a way to access the full capability of the instrument they have purchased. Thus, the driver must expose interfaces that are specific to that instrument alone. These are known as *instrument-specific* interfaces.

Instrument-specific interfaces are obviously not interchangeable. End users programming against a driver's instrument-specific capabilities typically do so to access functionality that is not available in the class-compliant interfaces. Nevertheless, for complex instruments such as spectrum analyzers, there will often be far more functionality in the IVI driver's instrument-specific hierarchy than in the class-compliant hierarchy.

Although, by definition, IVI does not dictate the syntax and semantics of instrument-specific interfaces, there are still guidelines that must be followed in the design of the interfaces if the driver is to be IVI compliant. Most of these rules are laid out in *IVI-3.4: API Style Guide*. 
Repeated Capabilities

Instruments often contain multiple instances of the same kind of functionality or capability. For instance, modern oscilloscopes may have multiple channels, each with independent settings for such things as vertical scale. IVI refers to functionality that is duplicated in an instrument as a repeated capability.

For a complete explanation of repeated capabilities in IVI instrument drivers, see the IVI specification IVI-3.1: Driver Architecture Specification.

Accessing Repeated Capabilities from Client Code

The repeated capability structure for IVI-COM drivers is designed to facilitate easy access from client programming languages. The following C++ code snippet shows how a client would access a repeated capability instance called "REF" in the Channels collection of an IviScope driver.

```cpp
IIviScopePtr spScope(CLSID_AcmeE4321);
// ... Initialize called
IIviScopeChannelPtr spChan = spScope->Channels->Item["REF"]; spChan->Enabled = true;
```

Instances of repeated capabilities are identified by name. The Item method uses this name to locate and return a pointer to the desired instance.

There are two types of repeated capability names -- physical names and virtual names.

Physical Repeated Capability Names

Physical repeated capability names are those that the driver writer specifies during driver development. These are the names that the driver natively understands.

Virtual Repeated Capability Names

Virtual names allow end users to specify their own names for repeated capability
instances and map those names to the physical names that the driver natively understands. The user establishes the virtual-to-physical name mappings in the IVI Configuration Store. At runtime, the user can supply a virtual channel name to the Item method and the driver will use the mappings in the IVI Configuration Store to translate the virtual name into a physical name that the driver understands.

The purpose of virtual names is to facilitate driver interchangeability. If a user writes application code using virtual names, then no modifications to the application code are necessary when an existing driver is replaced with one that natively understands different physical names. The user need only modify the appropriate settings in the IVI Configuration Store to map their virtual names to the new driver's physical names.
Standard IVI Driver Features

Many of the customer benefits of IVI drivers lie in the standard features that IVI drivers support. The following section discusses some of the most important features for users of IVI drivers to understand.

State Caching

As a means to offer better performance in customer test applications, IVI drivers often support a feature known as state caching. The principle of state caching is quite simple, although in practice it can be extraordinarily difficult to implement properly.

When a user enables state caching for an IVI driver, the driver attempts to determine if new settings passed into the driver are different from the actual instrument state. In this way, I/O traffic between the driver and the instrument is reduced, as fewer I/O calls are performed.

In the simplest example of when state caching improves performance, consider what happens when a user passes identical values several times in a row for the frequency property on a function generator driver. Without state caching, each call into the driver would result in an I/O function call to the instrument, whereas a driver performing state caching would detect that subsequent calls to the frequency property were redundant and should not result in any communication with the instrument. Correspondingly, the driver can "remember" the last known value for instrument properties so that queries can be performed by retrieving data from computer memory, rather than by performing instrument I/O.

As with caching schemes in other types of software systems, IVI driver state caching is fraught with potential pitfalls. The chief challenge in properly implementing state caching is maintaining cache coherency. Complex instruments often have complex couplings between instrument settings, making it difficult for the driver to track whether or not settings stored in the driver cache are valid. Consequently, many instrument vendors choose not to implement state caching in the driver.

For more information on state caching, see IVI-3.1: Driver Architecture Specification.
Range Checking

Range checking in IVI drivers simply refers to the ability for the driver to test whether or not input values passed to the driver are within an acceptable range. End users typically use range checking during debugging and, after validation is complete, disable it to maximize performance.

For more information on range checking, see IVI-3.1: Driver Architecture Specification.

Coercion and Coercion Recording

Instruments often accept only a discrete set of values for certain properties. For example, a DMM might only accept values such as 10 V, 100 V, 300 V for the voltage range setting. These discrete value sets will necessarily be different for different vendor's instruments. An Agilent DMM may accept 10 V, 100 V, 300 V while a Fluke DMM accepts 20 V, 200 V, and 400 V. As a result, IVI class specifications are forced to define such real-valued properties as accepting a continuous range of values, even though the underlying instrument only accepts a discrete set of values.

Following along with the DMM voltage range example, if a user sends a value of 150 V for the voltage range, the instrument will internally coerce the value to one of the accepted values, such as 300 V. To maximize interchangeability, the IVI class specifications provide guidance on how IVI drivers should coerce user requested values for specific properties.

To aid end users in building interchangeable applications, IVI drivers may implement a feature called coercion recording. If a user enables coercion recording, the IVI driver retains a log of each user value that it coerces. Standard functions in the IVI Inherent Capabilities provide the user with access to the driver's coercion log.

For more information on coercion, see IVI-3.1: Driver Architecture Specification.

Simulation

IVI drivers are often constructed for complex instruments that are difficult to
procure or that are in high demand at the end user facility. Consequently, it may be the case that end users need to be able to test their application programs without being physically connected to an instrument. IVI driver simulation is a feature that allows end users to use IVI drivers without a physical connection to the instrument.

When an end user enables simulation in an IVI driver, the driver performs no I/O communication whatsoever. IVI drivers are free in their choice of mechanisms for returning meaningful output values to the end user, but the intent is that the user program should be able to proceed normally when calling functions on the driver. This allows the end user to exercise their test application before the actual hardware is available.

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IVI Compliance Information

This page provides IVI-required compliance information for the IviLxiSync driver.
**Compliance Category**

IVI drivers come in a variety of types and configurations. This section provides IVI-required compliance information on the various categories of IVI compliance for the IviLxiSync driver.

<table>
<thead>
<tr>
<th>Compliance Category</th>
<th>Compliance Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category Name</td>
<td></td>
</tr>
<tr>
<td>Class Specification</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
</tbody>
</table>

**Class Capability Groups**

In addition to the IVI Inherent Capabilities required of all IVI instrument drivers, the IviLxiSync driver supports the instrument class.

The following table lists each of the capability groups for the instrument class and indicates whether the group is supported by the IviLxiSync driver.

<table>
<thead>
<tr>
<th>Capability Name</th>
<th>Supported</th>
<th>Description</th>
</tr>
</thead>
</table>


Optional Features

This section provides IVI-required information regarding optional IVI driver features supported by the IviLxiSync driver.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interchangeability checking</td>
<td>No</td>
</tr>
<tr>
<td>State caching</td>
<td>No</td>
</tr>
<tr>
<td>Coercion recording</td>
<td>No</td>
</tr>
</tbody>
</table>
Driver Identification

This section provides IVI-required identity information for the IviLxiSync driver.

<table>
<thead>
<tr>
<th>Identification Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Revision</td>
<td>1.0</td>
</tr>
<tr>
<td>Driver Vendor</td>
<td>IVI Foundation</td>
</tr>
<tr>
<td>Driver Description</td>
<td></td>
</tr>
<tr>
<td>Prefix/Component Identifier</td>
<td>IviLxiSync</td>
</tr>
</tbody>
</table>
IviLxiSync Reference
Ivi.LxiSync.Interop Namespace

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IviLxiSync</td>
<td></td>
</tr>
</tbody>
</table>

Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IviLxiSync</td>
<td>LxiSync root interface.</td>
</tr>
<tr>
<td>IviLxiSyncArm</td>
<td>LxiSync arm interface.</td>
</tr>
<tr>
<td>IviLxiSyncArmAlarm</td>
<td>LxiSync arm alarm interface.</td>
</tr>
<tr>
<td>IviLxiSyncArmAlarms</td>
<td>LxiSync repeated capability interface containing methods and properties that apply to all arm alarms defined for the device. A particular arm alarm can be accessed using the Item property.</td>
</tr>
<tr>
<td>IviLxiSyncArmSource</td>
<td>LxiSync arm source interface.</td>
</tr>
<tr>
<td>IviLxiSyncArmSources</td>
<td>LxiSync repeated capability interface containing methods and properties that apply to all arm sources defined for the device. A particular arm source can be accessed using the Item property.</td>
</tr>
<tr>
<td>IviLxiSyncEvent</td>
<td>LxiSync event interface.</td>
</tr>
<tr>
<td>IviLxiSyncEventLog</td>
<td>LxiSync event log interface.</td>
</tr>
<tr>
<td>IviLxiSyncEvents</td>
<td>LxiSync repeated capability interface containing methods and properties that apply to all events defined for the device. A particular event can be accessed using the Item property.</td>
</tr>
<tr>
<td>IviLxiSyncTime</td>
<td>LxiSync time interface.</td>
</tr>
<tr>
<td>IviLxiSyncTrigger</td>
<td>LxiSync trigger interface.</td>
</tr>
</tbody>
</table>
**IlviLxiSyncTriggerAlarm**

LxiSync trigger alarm interface.

**IlviLxiSyncTriggerAlarms**

LxiSync repeated capability interface containing methods and properties that apply to all trigger alarms defined for the device. A particular trigger alarm can be accessed using the Item property.

**IlviLxiSyncTriggerSource**

LxiSync trigger source interface.

**IlviLxiSyncTriggerSources**

LxiSync repeated capability interface containing methods and properties that apply to all trigger sources defined for the device. A particular trigger source can be accessed using the Item property.

### Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IviLxiSyncArmSourceDetectionEnum</td>
<td>Defined values for the style of arm source detection.</td>
</tr>
<tr>
<td>IviLxiSyncEventDriveModeEnum</td>
<td>Event driver mode.</td>
</tr>
<tr>
<td>IviLxiSyncSourceSlopeEnum</td>
<td>Defined values for the slope of the event that is inbound to the event subsystem that will cause the generation of an outbound event.</td>
</tr>
<tr>
<td>IviLxiSyncTriggerSourceDetectionEnum</td>
<td>Defined values for the Trigger Source attribute.</td>
</tr>
<tr>
<td>IviLxiSyncErrorCodesEnum</td>
<td></td>
</tr>
</tbody>
</table>

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IviLxiSyncArmSourceDetectionEnum

 Enumeration

Defined values for the style of arm source detection.

[Visual Basic]
Public Enum IviLxiSyncArmSourceDetectionEnum

[C#]
public enum IviLxiSyncArmSourceDetectionEnum

[C++]
enum IviLxiSyncArmSourceDetectionEnum

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IviLxiSyncArmSourceDetectionRise</td>
<td>Configures the LXI device to arm on the rising edge of the arm source.</td>
</tr>
<tr>
<td>IviLxiSyncArmSourceDetectionFall</td>
<td>Configures the LXI device to arm on the falling edge of the arm source.</td>
</tr>
<tr>
<td>IviLxiSyncArmSourceDetectionHigh</td>
<td>Configures the LXI device to arm while the arm source is high, that is, while it remains true.</td>
</tr>
<tr>
<td>IviLxiSyncArmSourceDetectionLow</td>
<td>Configures the LXI device to arm while the arm source is low, that is, while it remains false.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Ivi.LxiSync.Interop

See Also

Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IviLxiSyncEventDriveModeEnum Enumeration

Event driver mode.

[Visual Basic]
Public Enum IviLxiSyncEventDriveModeEnum

[C#]
public enum IviLxiSyncEventDriveModeEnum

[C++]
enum IviLxiSyncEventDriveModeEnum

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IviLxiSyncEventDriveModeDriven</td>
<td>Enables the event in driven mode.</td>
</tr>
<tr>
<td>IviLxiSyncEventDriveModeOff</td>
<td>Disables the event.</td>
</tr>
<tr>
<td>IviLxiSyncEventDriveModeWiredOr</td>
<td>Enables the event in wired-OR mode.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
IviLxiSync Reference
IviLxiSyncSourceSlopeEnum Enumeration

Defined values for the slope of the event that is inbound to the event subsystem that will cause the generation of an outbound event.

[Visual Basic]
Public Enum IviLxiSyncSourceSlopeEnum

[C#]
public enum IviLxiSyncSourceSlopeEnum

[C++]
enum IviLxiSyncSourceSlopeEnum

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IviLxiSyncSourceSlopeRise</td>
<td>The event will be transmitted with a rising edge.</td>
</tr>
<tr>
<td>IviLxiSyncSourceSlopeFall</td>
<td>The event will be transmitted with a falling edge.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

Ivi.LxiSync.Interop Namespace
IviLxiSync Reference
**IviLxiSyncTriggerSourceDetectionEnum**

Defined values for the Trigger Source attribute.

[Visual Basic]
Public Enum IviLxiSyncTriggerSourceDetectionEnum

[C#]
public enum IviLxiSyncTriggerSourceDetectionEnum

[C++]
enum IviLxiSyncTriggerSourceDetectionEnum

**Members**

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IviLxiSyncTriggerSourceDetectionRise</strong></td>
<td>Configures the LXI device to trigger on the rising edge of the trigger source.</td>
</tr>
<tr>
<td><strong>IviLxiSyncTriggerSourceDetectionFall</strong></td>
<td>Configures the LXI device to trigger on the falling edge of the trigger source.</td>
</tr>
</tbody>
</table>

**Requirements**

**Namespace:** [Ivi.LxiSync.Interop](#)

**Assembly:** Ivi.LxiSync.Interop (in Ivi.LxiSync.Interop.dll)

**See Also**

[Ivi.LxiSync.Interop Namespace](#)
IviLxiSync Reference
IviLxiSync Class


[Visual Basic]
Public Class IviLxiSync

[C#]
public class IviLxiSync

[C++]
class IviLxiSync

Requirements

Namespace: Ivi.LxiSync.Interop


See Also

Ivi.LxiSync.Interop Namespace

© 2006 IVI Foundation. All rights reserved.
LxiSync Reference
Welcome to the LxiSync Interface Reference

This section contains the reference material for the LxiSync APIs. These APIs are implemented by the driver for all Class A/B LXI devices. They provide control of the event capabilities of LXI including configuring the trigger and its sources and configuring how events are transmitted to other devices from within the instrument. This includes three distinct trigger mechanisms:

- **Wired Trigger** – LXI devices have a wired trigger bus. This is a physical connection that is routed between each module. The LXI Wired Trigger bus provides eight parallel trigger lines. Each of the eight trigger lines can be used in either a driven mode or in a wired-OR mode. In driven mode a single device drives the trigger line either high or low, and any device on the bus can receive that signal. In wired-OR mode, the signal on the trigger line represents the logical OR of each of the devices that is currently enabled to drive the line.
- **LAN Triggers** – LXI defines two protocols for issuing triggers over the LAN. One is based on UDP, and is transmitted to multiple receivers (multicast). The other is based on TCP and is transmitted to a single receiver.
- **Alarms** – Accurate alarm triggers are enabled by the IEEE 1588 synchronized real-time clock in Class A and B LXI modules. The LxiSync API provides a way to set a specific time at which an event (for instance, a trigger) will occur. Since each device in the system has synchronized clocks, this mechanism can be used to take simultaneous, or other carefully synchronized measurements in a system.

The LxiSync Interface provides ways to control each of these clock sources and how they are combined through both Arm and Trigger interfaces to ultimately enable an action to take place in the instrument.

The Event subsystem controls when, and how, various events that are relevant to the particular instrument are transmitted to other instruments. For instance, a LAN trigger may be sent when a signal generator has completed a sweep.

Finally, the LxiSync API provides the necessary functions for interaction with
the IEEE 1588 clock in an instrument.

The LxiSync API is not an instrument class. Typically, instruments that logically belong to a class that is defined by IVI will implement the class API and the LxiSync API. The LxiSync API does not include the APIs for accessing the conventional trigger mechanisms since these are already in the existing classes. The key link between the two APIs is the trigger source, which is where the final source of the trigger is specified. Class compliant applications will use the Trigger Source property to set the trigger to an LXI based source, and use the trigger source property (if it exists) in the class API to access other sources.

2005 LXI Consortium. All rights reserved.