Interchangeable Virtual Instruments

IVI-3.18: IVI.NET Utility Classes and Interfaces Specification

October 16, 2013 Edition
Revision 1.2

Important Information

The IVI.NET Utility Classes and Interfaces specification is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at www.ivifoundation.org.

Table of Contents

Table of Contents

Revision History

This section is an overview of the revision history of the IVI.NET Utility Classes and Interfaces Specification. Specific individual additions/modifications to the document in draft revisions are denoted with diff-marks, "|", in the right hand column of each line of text to which the change/modification applies.

Table 1. IVI.NET Utility Classes and Interfaces Specification

Revision Number	Date of Revision	Revision Notes
Revision 1.0	June 9, 2010	First approved version
Revision 1.1	October 14, 2010	Changes to Waveform and Spectrum interface and class descriptions that arose during final implementation and unit testing.
Revision 1.1	April 15, 2011	Editorial Changes: Clarify that classes defined in this spec are not guaranteed to be thread safe. Clarify the description of the Data property in IMemoryWaveform/Spectrum.
Revision 1.1	June 30, 2011	Editorial Changes: Add the TriggerSources class.
Revision 1.2	March10, 2012	Editorial Changes: Add three trigger strings to section 14.
Revision 1.2	August 6, 2012	Editorial Changes: Change references to the WaveformDataArrayTooSmallException and SpectrumDataArrayTooSmallException to DataArrayTooSmallException
Revision 1.2	October 16, 2013	Editorial Changes: Change Waveform/Spectrum StartTime and EndTime from PrecisionDateTime to PrecisionTimeSpan, and document the semantics. Correct the description of scaled array data.

1. Summary of Contents

The IVI.NET Utility Classes and Interfaces provide classes, interfaces, and other IVI.NET API elements that can be used in IVI.NET instrument class APIs and IVI.NET drivers as needed.

The following table summarizes the classes and interfaces described in this document.

Table 1-1. IVI.NET Utility Classes and Interfaces and Intended Users

Class/Interface	Intended Use
PrecisionDateTime	A date/time class with resolution suitable for representing IEEE 1588 times
PrecisionTimeSpan	A time span class with resolution suitable for representing IEEE 1588 times.
IWaveform<T>	A representation of a waveform of time domain values.
IMemoryWaveform<T>	A representation of a basic waveform that allows for streaming time domain values.
ISpectrum<T>	A representation of a spectrum of frequency domain values.
IMemorySpectrum<T>	A representation of a basic spectrum that allows for streaming frequency domain values.
Waveform<T>	An implementation of a waveform of time domain values.
Spectrum<T>	An implementation of a spectrum of frequency domain values.
Repeated Capability Base Interfaces	Interfaces that are extended to create repeated capability collections and collection members.
Auto & Slope Enumerations	Commonly used enumerations with general use in IVI instrument classes and specific drivers.
TriggerSource	Standard trigger source strings

1.1 References

Several other documents and specifications are related to this specification. These other related documents are as follows:

- IVI 3.1—Driver Architecture Specification
- IVI 3.2—Inherent Capabilities Specification
- IVI 3.2—Installation Requirements Specification
- IVI 3.3—Standard Cross-Class Capabilities Specification

1.2 Implementation

The current installation package for the IVI Foundation IVI.NET Shared Components, including the IVI.NET Utility Classes and Interfaces, is available from the IVI Foundation website at http://www.ivifoundation.org.

All of the IVI defined API elements in this specficication are defined in the Ivi.Driver namespace.

The IVI defined classes in this specficication are not guaranteed to be thread-safe.

2. PrecisionDateTime Struct

2.1 Overview

Instruments sometimes require an absolute time which exceeds the resolution of the .NET Framework DateTime struct. To address these cases, IVI.NET provides the PrecisionDateTime class, which provides a level of resolution similar to that defined by the IVI LXI Sync standard.

PrecisionDateTime supports a range of dates from the beginning of the IEEE 1588 epoch 0 (that is, 1/1/1970) through December 31, 9999. Time is internally represented in seconds since January 1, 1970 (the IEEE 1588 epoch 0). PrecisionDateTime stores date and time with femtosecond (1.0e-15 second) resolution.

The PrecisionDateTime class is always based on the Gregorian calendar. Time may be UTC time or local time.

2.1.1 Relationship to .NET Framework DateTime Struct

The PrecisionDateTime class is modeled on the .NET Framework System.DateTime struct. The primary differences between the two are (1) DateTime only provides resolution to 100 nanoseconds, while PrecisionDateTime provides resolution to 1 femtosecond, and (2) DateTime can represent dates from 1/1/0001 through 12/31/9999, while PrecisionDateTime can only represent dates from 1/1/1970 through 12/31/9999.

PrecisionDateTime contains a method, ToDateTime(), that creates a corresponding DateTime object. While PrecisionDateTime duplicates many useful methods and properties of DateTime such as the Year, Month, and Day properties, some DateTime properties and methods are best accessed by using ToDateTime().

Since PrecisionDateTime is targeted at a test and measurement market, it does not try to duplicate all of the general purpose features of DateTime. For example, PrecisionDateTime does not support the full variety of DateTime format specifiers, globalization, or serialization, and does not support the Unspecified DateTimeKind.

2.1.2 Relationship to LXISync

IVI 3.15: IviLxiSync Specification includes techniques that allow instrument operation to be triggered at given times and for timestamps to be associated with measured data. *IVI 3.15: IviLxiSync Specification* also specifies a particular data format (a pair of double values) that is used to contain a high-resolution time stamp value. The first double is Time Seconds and the second is Time Fraction. The sum refers to the time since IEEE 1588 epoch 0. To allow IVI.NET drivers to interoperate with LXI sync times, the PrecisionDateTime class provides constructors and properties that represent time with two doubles - Seconds Since Epoch and Seconds Fractional, but note that the range of years available in PrecisionDateTime is more limited than in LXISync. This restriction is more theoretical than practical in a test and measurement context.

2.1.3 Inherited Interfaces

The PrecisionDateTime class derives from the following interfaces:

- IComparable
- IComparable<PrecisionDateTime>
- IConvertible interface
- IEquatable<PrecisionDateTime>
- IComparable defines “public int CompareTo(object obj)”. Refer to Section 2.4.14, CompareTo, for more details.

IComparable<PrecisionDateTime> defines "public int CompareTo(PrecisionDateTime other)". Refer to Section 2.4.14, If either pdt1 or pdt2 is NotATime, but not both, this method throws the Not A Time exception.

CompareTo, for more details.

IEquatable<PrecisionDateTime> defines "public bool Equals(PrecisionDateTime other)". Refer to Section □, If either this instance or pdt is NotATime, but not both, this property throws the Not A Time exception.

Equals, for more details.

2.2 PrecisionDateTime Constructors

Description

PrecisionDateTime has two types of constructors. PrecisionDateTime-based constructors include a DateTime parameter and, optionally, additional parameters to support additional resolution.

Seconds-based constructors include one or more parameters that represent the number of seconds since the beginning of the IEEE 1588 epoch (that is, the total number of seconds since 1/1/1970), to femtosecond resolution.

Description – DateTime-based Constructors

The basic PrecisionDateTime based constructors take only a .NET Framework DateTime parameter. Since the DateTime class only supports a resolution of 100 nanoseconds, there is an overload that takes a DateTime parameter and Double parameter of additional seconds. The double allows for femtosecond resolution. If the resulting date is greater than 12/31/9999, the constructor throws an exception.

The new PrecisionDateTime object has the same DateTimeKind as the DateTime parameter, with the exception that Unspecified time is coerced to Local time.

.NET Prototypes – DateTime-based Constructers

```
public PrecisionDateTime(DateTime dateTime)

public PrecisionDateTime(DateTime dateTime,
                         Double deltaSeconds)
```

Description – Seconds-based Constructors

Seconds-based constructors all take one or two parameters, where the parameter units are seconds since the beginning of the IEEE 1588 epoch (that is, the total number of seconds since 1/1/1970). If the total number of seconds results in a date beyond 12/31/9999, the constructor throws an exception.

If necessary, the result is rounded to the nearest Femtosecond. Results that are exactly exactly .5 femtoseconds from a valid whole femtosecond are rounded up.

The constructors that take doubles `secondsSinceEpoch` and `secondsFractional` as parameters accept both a `secondsSinceEpoch` parameter, which may optionally include a fractional part, and an `secondsFractional` parameter between 0.0 and 1.0, and adds the two together to get the correct date and time. While setting the `secondsSinceEpoch` parameter to a non-integer is not encouraged, this behavior avoids throwing an exception when the data can be interpreted in a meaningful way.The DateTimeKind may be explicitly defined as UTC time, local time, or unspecified using the overloads that include the `kind` parameter. Unspecified time is coerced to Local time.

.NET Prototypes – Seconds-based Constructers

```
public PrecisionDateTime(Decimal seconds)

public PrecisionDateTime(Decimal seconds,
                         DateTimeKind kind)

public PrecisionDateTime(Double secondsSinceEpoch,
                         Double fractionalSeconds)

public PrecisionDateTime(Double secondsSinceEpoch,
                         Double fractionalSeconds,
                         DateTimeKind kind)
```

Description – String-based Constructors

The string-based constructor takes one string parameters that indicates the time since 1/1/1970 in seconds, including the decimal point. If the total number of seconds results in a date beyond 12/31/9999, the constructor throws an error.

If necessary, the result is rounded to the nearest Femtosecond. Results that are exactly exactly .5 femtoseconds from a valid whole femtosecond are rounded up.

The DateTimeKind may be explicitly defined as UTC time, local time, or unspecified using the overloads that include the `kind` parameter. Unspecified time is coerced to Local time.

.NET Prototypes – String-based Constructers

```
public PrecisionDateTime(String timeSinceEpoch)

public PrecisionDateTime(String timeSinceEpoch,
                         DateTimeKind kind)
```

Parameters

Input	Description	Data Type
`dateTime`	A .NET Framework DateTime object that refers to a date after the beginning of the IEEE 1588 epoch (that is, after 1/1/1970). The number of ticks (100-nanosecond intervals) in the DateTime object is used to initialize the PrecisionDateTime object, adjusted for the difference in data range.	`DateTime`
`deltaSeconds`	The number of seconds to add to the `dateTime` object used to initialize the precision date time. May be negative or positive.	`Double`
`seconds`	The number of seconds (including fractional seconds) since the beginning of the IEEE 1588 epoch (that is, the total number of seconds since 1/1/1970). Must be positive.	`Decimal`
`secondsSinceEpoch`	The total number of seconds since the beginning of the IEEE 1588 epoch (that is, the total number of seconds since 1/1/1970), rounded to the nearest second.	`Double`
`fractionalSeconds`	A fractional number of seconds (greater than or equal to 0.0, and less than 1.0) added to the time represented by `lxiBaseSeconds`. This parameter provides for femtosecond resolution to the right of the decimal. Resolution finer than femtoseconds will be rounded. Must be positive.	`Double`
`kind`	One of the DateTimeKind values that indicates whether the date and time specify local time or Coordinated Universal Time (UTC). The default value is Local.	`DateTimeKind`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if a constructor attempts to construct a date before 1/1/1970.
- An exception will be thrown if a constructor attempts to construct a date beyond 12/31/9999.

- An exception will be thrown is the values for the seconds parameters are out of range.

2.3 PrecisionDateTime Properties

The PrecisionDateTime class defines the following properties:

- Day
- Day of Week
- Day of Year
- Femtosecond
- Hour
- Kind
- MaxValue
- Microsecond
- Millisecond
- Minute
- MinValue
- Month
- Nanosecond
- Now
- Picosecond
- Second
- Seconds Fractional
- Seconds Since Epoch
- Year

This section describes the behavior and requirements of each property.

2.3.1 Day

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Day;
```

Description

The day of the month represented by this instance, expressed as an integer value between 1 and 31.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.2 Day of Week

Data Type	Access
`DayOfWeek`	RO

.NET Prototype

```
public DayOfWeek DayOfWeek;
```

Description

A `DayOfWeek` enumerated constant that indicates the day of the week of this `PrecisionDateTime` value.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.3 Day of Year

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 DayOfYear;
```

Description

The day of the year represented by this instance, expressed as an integer value between 1 and 366.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.4 Femtosecond

Data Type	Access
`Int64`	RO

.NET Prototype

```
public Int64 Femtosecond;
```

Description

The femtoseconds component of the date and time represented by this instance, expressed as an integer value between 0 and 999,999,999,999,999.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.5 Hour

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Hour;
```

Description

The hour component of the date represented by this instance, expressed as an integer value between 0 and 23.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.6 IsNotATime

Data Type	Access
`Boolean`	RO

.NET Prototype

```
public Boolean IsNotATime;
```

Description

A value that indicates whether the time represented by this instance is an actual time, or NaT (Not a Time).

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

2.3.7 Kind

Data Type	Access
`DateTimeKind`	RO

.NET Prototype

```
public DateTimeKind Kind;
```

Description

A value that indicates whether the time represented by this instance is based on local time, Coordinated Universal Time (UTC), or neither.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.8 Max Value

Data Type	Access
`PrecisionDateTime`	RO, static

.NET Prototype

```
public static readonly PrecisionDateTime MaxValue;
```

Description

The largest possible value of PrecisionDateTime. This property is read-only. The value of this constant is equivalent to 23:59:59.999999999999999, December 31, 9999, exactly one femtosecond before 00:00:00, January 1, 10000.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

2.3.9 Microsecond

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Microsecond;
```

Description

The microsecond component of the date and time represented by this instance, expressed as an integer value between 0 and 999,999, rounded.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.10 Millisecond

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Millisecond;
```

Description

The milliseconds component of the date and time represented by this instance, expressed as an integer value between 0 and 999.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.11 Minute

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Minute;
```

Description

The minute component of the date represented by this instance, expressed as an integer value between 0 and 59.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.12 Min Value

Data Type	Access
`PrecisionDateTime`	RO, static

.NET Prototype

```
public static readonly PrecisionDateTime MinValue;
```

Description

The smallest possible value of PrecisionDateTime. This property is read-only. The value of this constant is the beginning of the IEEE 1588 epoch (that is, 00:00:00.000000000000000, January 1, 1970.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

2.3.13 Month

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Month;
```

Description

The month component of the date represented by this instance, expressed as an integer value between 1 and 12.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.14 Nanosecond

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Nanosecond;
```

Description

The nanosecond component of the date and time represented by this instance, expressed as an integer value between 0 and 999,999,999, rounded.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.15 NotATime

Data Type	Access
`PrecisionDateTime`	RO, static

.NET Prototype

```
public static PrecisionDateTime Now;
```

Description

A PrecisionDateTime instance that represents NaT (Not a Time).

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

2.3.16 Now

Data Type	Access
`PrecisionDateTime`	RO, static

.NET Prototype

```
public static PrecisionDateTime Now;
```

Description

The `PrecisionDateTime` object that is set to the current date and time on this computer, expressed as the local time, to `DateTime` resolution.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

2.3.17 Picosecond

Data Type	Access
`Int64`	RO

.NET Prototype

```
public int Picosecond;
```

Description

The picosecond component of the date and time represented by this instance, expressed as an integer value between 0 and 999,999,999,999, rounded.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.18 Second

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Second;
```

Description

The seconds component of the date represented by this instance, expressed as an integer value between 0 and 59.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.19 Seconds Fractional

Data Type	Access
`Double`	RO

.NET Prototype

```
public double SecondsFractional;
```

Description

The fractional portion (remainder) since the end of the last whole second. The value will always be greater that or equal to 0 and less than 1.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.20 Seconds Since Epoch

Data Type	Access
`Double`	RO

.NET Prototype

```
public double SecondsSinceEpoch;
```

Description

The total number of seconds since the beginning of the IEEE 1588 epoch (that is, the total number of seconds since 1/1/1970), rounded. The value does not have a fractional part. For the fractional part of the total number of seconds since the beginning of the IEEE 1588 epoch, see the Seconds Fractional property.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.3.21 Year

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Year;
```

Description

The year component of the date represented by this instance, expressed as an integer value between 1970 and 9999, inclusive.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If this instance of PrecisionDateTime is NotATime, this property throws the Not A Time exception.

2.4 PrecisionDateTime Methods

The PrecisionDateTime class defines the following methods:

- Add
- Add Days
- Add Femtoseconds
- Add Hours
- Add Microseconds
- Add Milliseconds
- Add Milliseconds
- Add Minutes
- Add Months
- Add Nanoseconds
- Add Picoseconds
- Add Seconds
- Add Years
- Compare
- CompareTo
- Subtract
- ToDateTime
- ToDecimal
- ToLocalTime
- ToString
- ToUniversalTime

The PrecisionDateTime class implements the following methods from the inherited IConvertible interface:

- IConvertible.GetTypeCode
- IConvertible.ToDateTime
- IConvertible.ToString

The PrecisionDateTime class overrides the following methods:

- Object.Equals
- Object.GetHashCode
- Object.ToString

This section describes the behavior and requirements of each of the above methods.

The PrecisionDateTime class does not implement other methods and properties from the inherited IConvertible interface because they do not return meaningful results. They return System.InvalidCastException, and are not otherwise documented in this specification.

2.4.1 Add

Description

Adds the value of the specified PrecisionTimeSpan or TimeSpan to the value of this instance.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototypes

```
public PrecisionDateTime Add(PrecisionTimeSpan pts)

public PrecisionDateTime Add(TimeSpan ts)
```

Parameters

Input	Description	Data Type
`pts`	A PrecisionTimeSpan that contains the interval to add.	`PrecisionTimeSpan`
`ts`	A TimeSpan that contains the interval to add.	`TimeSpan`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by pts or ts, respectively. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.2 AddDays

Description

Adds the specified number of days to the value of this instance.

The days parameter is the number of 24-hour periods to add. The fractional part of days is the fractional part of a day. For example, 4.5 is equivalent to 4 days, 12 hours, 0 minutes, 0 seconds, 0 milliseconds, and 0 ticks.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddDays(Double days)
```

Parameters

Input	Description	Data Type
`days`	A number of whole and fractional days. The parameter can be negative or positive.	`Double`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `days`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.3 AddFemtoseconds

Description

Adds the specified number of femtoseconds to the value of this instance.

A femtosecond is 1.0e-15 second.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddFemtoseconds(Int64 femtoseconds)
```

Parameters

Input	Description	Data Type
`femtoseconds`	A number of whole femtoseconds. The parameter can be negative or positive.	`Int64`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `femtoseconds`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.4 AddHours

Description

Adds the specified number of hours to the value of this instance.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddHours(Double hours)
```

Parameters

Input	Description	Data Type
`hours`	A number of whole and fractional hours. The parameter can be negative or positive.	`Double`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `hours`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.5 AddMicroseconds

Description

Adds the specified number of microseconds to the value of this instance.

A microsecond is 1.0e-6 second.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddMicroseconds(Double microseconds)
```

Parameters

Input	Description	Data Type
`microseconds`	A number of whole and fractional microseconds. The parameter can be negative or positive.	`Double`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `microseconds`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.6 AddMilliseconds

Description

Adds the specified number of milliseconds to the value of this instance.

A millisecond is 1.0e-3 second.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddMilliseconds(Double milliseconds)
```

Parameters

Input	Description	Data Type
`milliseconds`	A number of whole milliseconds. The parameter can be negative or positive.	`Double`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by milliseconds. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.7 AddMinutes

Description

Adds the specified number of minutes to the value of this instance.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddMinutes(Double minutes)
```

Parameters

Input	Description	Data Type
`minutes`	A number of whole and fractional minutes. The parameter can be negative or positive.	`Double`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `minutes`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.8 AddMonths

Description

This method calculates the resulting month and year, taking into account leap years and the number of days in a month, then adjusts the day part of the resulting PrecisionDateTime object. If the resulting day is not a valid day in the resulting month, the last valid day of the resulting month is used. For example, March 31st + 1 month = April 30th. The time-of-day part of the resulting PrecisionDateTime object remains the same as this instance.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddMonths(Int32 months)
```

Parameters

Input	Description	Data Type
`months`	A number of whole and fractional months. The parameter can be negative or positive.	`Int32`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `months`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if an operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.9 AddNanoseconds

Description

Adds the specified number of nanoseconds to the value of this instance.

A nanosecond is 1.0e-9 second.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddNanoseconds(Int64 nanoseconds)
```

Parameters

Input	Description	Data Type
`nanoseconds`	A number of whole nanoseconds. The parameter must be positive.	`Int64`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by nanoseconds. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if the resulting date would be beyond 12/31/9999.

2.4.10 AddPicoseconds

Description

Adds the specified number of picoseconds to the value of this instance.

A nanosecond is 1.0e-12 second.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddPicoseconds(Int64 picoseconds)
```

Parameters

Input	Description	Data Type
`picoseconds`	A number of whole picoseconds. The parameter must be positive.	`Int64`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by picoseconds. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if the resulting date would be beyond 12/31/9999.

2.4.11 AddSeconds

Description

Adds the specified number of seconds to the value of this instance. If necessary, the result is rounded to the nearest Femtosecond. Results that are exactly exactly .5 femtoseconds from a valid whole femtosecond are rounded up.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

If the seconds parameter is a Decimal or a Double, the fractional part is the fractional part of a second, and any resolution beyond femtosecond resolution is rounded. For example,

4.53945761103247 is equivalent to 4 seconds and 539457611032470 femtoseconds.

0.00000061103247123 is equivalent to 0 seconds and 611032471 femtoseconds.

0.00000061103247199 is equivalent to 0 seconds and 611032472 femtoseconds.

.NET Prototype

```
public PrecisionDateTime AddSeconds(Double seconds)

public PrecisionDateTime AddSeconds(Int64 seconds)
```

Parameters

Input	Description	Data Type
`seconds`	A number of seconds. The parameter must be positive.	`Double`
`seconds`	A number of whole seconds. The parameter must be positive.	`Int64`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by seconds. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if the resulting date would be beyond 12/31/9999.

2.4.12 AddYears

Description

Adds the specified number of years to the value of this instance.

This method calculates the resulting year taking into account leap years. The month and time-of-day part of the resulting PrecisionDateTime object remains the same as this instance.

If this instance is set to Not a Time, the method returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime AddYears(Int32 years)
```

Parameters

Input	Description	Data Type
`years`	A number of whole and fractional months. The parameter can be negative or positive.	`Int32`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by this instance and the time interval represented by `years`. If this instance is set to Not a Time, the method returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if a operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.4.13 Compare

Description

Compares two instances of PrecisionDateTime and returns an indication of their relative values.

PrecisionDateTime objects, are compared using their UTC time equivalents.

.NET Prototype

```
public static int Compare(PrecisionDateTime pdt1,
                          PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	The first PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	The second PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	A signed number indicating the relative values of pdt1 and pdt2. If the return value is less than zero, then pdt1 falls before pdt2. If the return value is equal to zero, then pdt1 and pdt2 are the same date and time. If the return value is greater than zero, then pdt1 falls after pdt2. If both pdt1 and pdt2 are NaT (Not a Time), the return value is zero.	`Int32`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If either pdt1 or pdt2 is NotATime, but not both, this method throws the Not A Time exception.

2.4.14 CompareTo

Description

Compares this instance to a specified PrecisionDateTime object and returns an indication of their relative values.

PrecisionDateTime objects are compared using their UTC time equivalents.

Any instance of PrecisionDateTime, regardless of its value, is considered greater than a null reference.

.NET Prototype

```
public int CompareTo(PrecisionDateTime pdt)

public int CompareTo(object obj)
```

Parameters

Input	Description	Data Type
`other`	A PrecisionDateTime object to compare.	`PrecisionDateTime`
`obj`	A boxed PrecisionDateTime object to compare, or a null reference.	`Object`

Output	Description	Data Type
`return value`	A signed number indicating the relative values of this instance and pdt. If the return value is less than zero, then this instance falls before pdt. If the return value is equal to zero, then this instance and pdt are the same date and time. If the return value is greater than zero, then this instance falls after pdt, or pdt is a null reference. If both this instance and pdt are NaT (Not a Time), the return value is zero.	`Int32`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If either this instance or pdt is NotATime, but not both, this property throws the Not A Time exception.

2.4.15 Equals

Description

This method returns true if this instance is the same instance as pdt.

.NET Prototype

```
public override bool Equals(PrecisionDateTime pdt)
```

Parameters

Output	Description	Data Type
`pdt`	A PrecisionDateTime.	

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.4.16 Subtract

Description

Subtracts the value of the specified PrecisionTimeSpan or TimeSpan from the value of this instance.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionDateTime Subtract(PrecisionTimeSpan pts)

public PrecisionDateTime Subtract(TimeSpan ts)
```

Parameters

Input	Description	Data Type
`pts`	A PrecisionTimeSpan that contains the interval to subtract.	`PrecisionTimeSpan`
`ts`	A TimeSpan that contains the interval to subtract.	`TimeSpan`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the date and time represented by this instance less the time interval represented by pts or ts, respectively.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if a operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.
- If this instance of PrecisionDateTime is NotATime, this method throws the Not A Time exception.

2.4.17 ToDateTime

Description

A new DateTime is returned whose value is is the value of this instance of PrecisionDateTime, rounded to the nearest 100 nanoseconds.[1] The DateTimeKind for the new DateTime is the same as for this instance of PrecisionDateTime

.NET Prototype

```
public DateTime ToDateTime()
```

Parameters

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the date and time represented by this instance with femtoseconds rounded to the nearest 100 nanoseconds.	`DateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If this instance of PrecisionDateTime is NotATime, this method throws the Not A Time exception.

[1] Rounding down insures that a valid DateTime object can (in theory, at least) be constructed from this instance of PrecisionDateTime.

2.4.18 ToDecimal

Description

A new Decimal is returned whose value is is the value of this instance of PrecisionDateTime in seconds, with resolution to the nearest femtosecond.

.NET Prototype

```
public Decimal ToDecimal()
```

Parameters

Output	Description	Data Type
`return value`	A decimal number that represents the number of seconds since the IEEE 1588 began (Jan. 1, 1970).	`DateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If this instance of PrecisionDateTime is NotATime, this method throws the Not A Time exception.

2.4.19 ToLocalTime

Description

A new PrecisionDateTime is returned whose value is is the value of this instance of PrecisionDateTime converted to local time (if needed). If DateTimeKind is unspecified in this instance, it is treated as local time.

.NET Prototype

```
public PrecisionDateTime ToLocalTime()
```

Parameters

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the date and time represented by this instance, converted to local time (if needed).	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If this instance of PrecisionDateTime is NotATime, this method throws the Not A Time exception.

2.4.20 ToString

Description

Converts the value of the current PrecisionDateTime object to its equivalent string representation.

In most cases, this string will be equivalent to `this.ToDateTime().ToString()`, with the addition of fractional seconds to femtosecond resolution whenever long times are used.

The format parameter is a PrecisionDateTime format string, which may be a single standard PrecisionDateTime format specifier as defined in the first table below, or a format string composed of custom PrecisionDateTime format specifier as defined in the second table below.

The following subset of standard DateTime format specifiers is allowed for PrecisionDateTime:

"d"	Short date.
"t"	Short time.
"T"	Long time.
"g"	General date / short time.
"G"	General / long time (default).
"s"	Sortable.
"u"	Universal sortable (invariant, valid only for UTC times).

Any string that is not in the list above is interpreted as a custom PrecisionDateTime format string.

A custom PrecisionDateTime format string consists of one or more custom PrecisionDateTime format specifiers, and that format string defines the text representation of a PrecisionDateTime object that is produced by a formatting operation.

The following subset of custom DateTime format specifiers is allowed for PrecisionDateTime:

Years	"yyyy"	Represents the year as a four digit number.
Months	"M"	Represents the month as a number from 1 through 12. A single-digit month is formatted without a leading zero.
	"MM"	Represents the month as a number from 01 through 12. A single-digit month is formatted with a leading zero.
Days	"d"	Represents the day of the month as a number from 1 through 31. A single-digit day is formatted without a leading zero.
	"dd"	Represents the day of the month as a number from 01 through 31. A single-digit day is formatted with a leading zero.
Hours	"h"	Represents the hour as a number from 1 through 12, that is, the hour as represented by a 12-hour clock. A single-digit hour is formatted without a leading zero.
	"hh"	Represents the hour as a number from 01 through 12, that is, the hour as represented by a 12-hour clock. A single-digit hour is formatted with a leading zero.
	"H"	Represents the hour as a number from 0 through 23, that is, the hour as represented by a 24-hour clock that counts the hours since

		midnight. A single-digit hour is formatted without a leading zero.
	"HH"	Represents the hour as a number from 00 through 23, that is, the hour as represented by a 24-hour clock that counts the hours since midnight. A single-digit hour is formatted with a leading zero.
Minutes	"m"	Represents the minutes as a number from 0 through 59. A single-digit minute is formatted without a leading zero.
	"mm"	Represents the minutes as a number from 00 through 59. A single-digit minute is formatted with a leading zero.
Seconds	"s"	Represents the seconds as a number from 0 through 59. A single-digit second is formatted without a leading zero
	"ss"	Represents the seconds as a number from 00 through 59. A single-digit second is formatted with a leading zero.
	"f " "ff" "f…f"	N 'f' characters, where N is from 1 to 15, represent the N most significant digits of the seconds fraction. Note that this is an extension of the DateTime format specifier, where N cannot be greater than 7.
Special	":"	The time separator defined in the current System.Globalization.DateTimeFormatInfo.TimeSeparator property that is used to differentiate hours, minutes, and seconds.
	"/"	The date separator defined in the current System.Globalization.DateTimeFormatInfo.DateSeparator property that is used to differentiate years, months, and days.
	"'"	Quoted string (apostrophe). Displays the literal value of any string between two apostrophe (') characters.
	"%c"	Represents the result associated with a custom format specifier "*c*", when the custom DateTime format string consists solely of that custom format specifier. For example, to use the "d" custom format specifier by itself, specify "%d".
	"tt"	Represents the AM or PM indicator.
	"Z"	If this instance is UTC time, this displays a "Z". If this instance is local time, this displays a "Z" followed by the UTC offset in "-hh:mm" format.

For any other character, the literal value of the character is copied to the result string, and does not affect formatting.

If this instance is NaT (Not a Time), the method returns "NaT", regardless of the format string.

.NET Prototype

```
public string ToString(string format)
```

Parameters

Input	Description	Data Type
`format`	A PrecisionDateTime format string.	`IFormatProvider`

Output	Description	Data Type
`return value`	A string representation of the value of the current PrecisionDateTime object.	`string`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.4.21 ToUniversalTime

Description

A new PrecisionDateTime is returned whose value is is the value of this instance of PrecisionDateTime converted to universal time (if needed). If DateTimeKind is unspecified in this instance, it is treated as local time.

If this instance of PrecisionDateTime is NotATime, this method returns Not A Time.

.NET Prototype

```
public PrecisionDateTime ToUniversalTime()
```

Parameters

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the date and time represented by this instance, converted to universal time (if needed). If this instance of PrecisionDateTime is NotATime, this method returns Not A Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.4.22 IConvertible.ToDateTime

Description

A new DateTime is returned whose value is is the value of this instance of PrecisionDateTime. The DateTimeKind for the new DateTime is the same as for this instance of PrecisionDateTime.

If necessary, the result is rounded to the nearest Femtosecond. Results that are exactly exactly .5 femtoseconds from a valid whole femtosecond are rounded up.

.NET Prototype

```
DateTime IConvertible.ToDateTime(IFormatProvider provider)
```

Parameters

Input	Description	Data Type
`provider`	A format provider.	`IFormatProvider`

Output	Description	Data Type
`return value`	A DateTime whose value is the date and time represented by this instance, to 100 nanosecond resolution.	`DateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If this instance of PrecisionDateTime is NotATime, this method throws the Not A Time exception.

2.4.23 IConvertible.ToDecimal

Description

A new Decimal is returned whose value is is the value of this instance of PrecisionDateTime in seconds, with resolution to the nearest femtosecond.

.NET Prototype

```
Decimal IConvertible.ToDecimal(IFormatProvider provider)
```

Parameters

Input	Description	Data Type
`provider`	A format provider.	`IFormatProvider`

Output	Description	Data Type
`return value`	A decimal number that represents the number of seconds since the IEEE 1588 began (Jan. 1, 1970).	`Decimal`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If this instance of PrecisionDateTime is NotATime, this method throws the Not A Time exception.

2.4.24 IConvertible.ToString

Description

Refer to section 2.4.27, *Object.ToString* for details on the implementation of this method. The string returned is formatted using the “G” format specifier.

.NET Prototype

```
string IConvertible.ToString(IFormatProvider provider)
```

Parameters

Input	Description	Data Type
`provider`	A format provider.	`IFormatProvider`

Output	Description	Data Type
`return value`	A string representation of the value of the current PrecisionDateTime object. The string is formatted usings the “G” format specifier for precision date/time strings.	`string`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.4.25 Object.Equals

Description

This method returns true if this instance is the same instance as obj.

.NET Prototype

```
public override bool Equals(object obj)
```

Parameters

Output	Description	Data Type
`obj`	Any .NET object.	`object`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.4.26 Object.GetHashCode

Description

Returns a hash code for the object.

.NET Prototype

```
public override int GetHashCode()
```

Parameters

Output	Description	Data Type
`Return code`	The hash code for the object.	`int`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.4.27 Object.ToString

Description

Converts the value of the current PrecisionDateTime object to its equivalent string representation using the default “G” format specifier. Refer to section 2.4.20, ToString, for details.

If this instance is NaT (Not a Time), the method returns “NaT”.

.NET Prototype

```
public override string ToString()
```

Parameters

Output	Description	Data Type
`return value`	A string representation of the value of the current PrecisionDateTime object. The string is formatted usings the “G” format specifier for precision date/time strings.	`string`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.5 PrecisionDateTime Operators

The PrecisionDateTime class defines the following operators:

- +
- -
- ==
- !=
- >=
- <=
- >
- <

This section describes the behavior and requirements of each operator.

2.5.1 + (Addition Operator)

Description

Adds a specified time interval to a specified date and time, yielding a new date and time.

If pdt is set to Not a Time, the operation returns Not a Time.

This method does not change the value of this PrecisionDateTime. Instead, a new PrecisionDateTime is returned whose value is the result of this operation.

.NET Prototype

```
public static PrecisionDateTime operator +(PrecisionDateTime pdt,
                                           PrecisionTimeSpan pts)

public static PrecisionDateTime operator +(PrecisionDateTime pdt,
                                           TimeSpan ts)
```

Parameters

Input	Description	Data Type
`pdt`	A PrecisionDateTime that contains the date and time to be added to.	`PrecisionDateTime`
`pts`	A PrecisionTimeSpan that contains the interval to add.	`PrecisionTimeSpan`
`ts`	A TimeSpan that contains the interval to add.	`TimeSpan`

Output	Description	Data Type
`return value`	A PrecisionDateTime whose value is the sum of the date and time represented by pdt and the time interval represented by pts or ts, respectively. If pdt is set to Not a Time, the operation returns Not a Time.	`PrecisionDateTime`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if a operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.5.2 - (Subtraction Operator)

Description

Subtracts (1) a specified PrecisionTimeSpan or TimeSpan from a specified PrecisionDateTime, yielding a new PrecisionDateTime, or (2) a specified PrecisionDateTime or DateTime from a specified PrecisionDateTime, yielding a PrecisionTimeSpan that is the interval between the two.

If pdt or pdt2 is set to Not a Time, the operation returns Not a Time.

.NET Prototype

```
public static PrecisionDateTime operator -(
                                    PrecisionDateTime pdt,
                                    PrecisionTimeSpan pts)

public static PrecisionDateTime operator -(
                                    PrecisionDateTime pdt,
                                    TimeSpan ts)

public static PrecisionTimeSpan operator -(
                                    PrecisionDateTime pdt,
                                    PrecisionDateTime pdt2

public static PrecisionTimeSpan operator -(
                                    PrecisionDateTime pdt,
                                    DateTime dt)
```

Parameters

Input	Description	Data Type
`pdt`	A PrecisionDateTime that contains the date and time to be subtracted from.	`PrecisionDateTime`
`pts`	A PrecisionTimeSpan that contains the interval to subtract.	`PrecisionTimeSpan`
`ts`	A TimeSpan that contains the interval to subtract.	`TimeSpan`
`pdt2`	A PrecisionDateTime that contains the date and time to subtract.	`PrecisionDateTime`
`dt`	A DateTime that contains the date and time to subtract.	`DateTime`

Output	Description	Data Type
`return value`	A PrecisionDateTimewhose value is the sum of the date and time represented by pdt and the time interval represented by pts or ts, respectively. If pdt is set to Not a Time, the operation returns Not a Time.	`PrecisionDateTime`

	A PrecisionTimeSpan whose value is the interval between pdt and pdt2 or dt. This value will be negative if pdt2 > pdt or dt > pdt. If pdt or pdt2 is set to Not a Time, the operation returns Not a Time.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- An exception will be thrown if a operation attempts to construct a date earlier than 1/1/1970 or later than 12/31/9999.

2.5.3 == (Equality Operator)

Description

Determines whether two specified instances of PrecisionDateTime are equal.

.NET Prototype

```
public static bool operator ==(PrecisionDateTime pdt1,
                               PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	A PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	A PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	`true` if `pdt1` and `pdt2` represent the same date and time, or if they are both Not a Time; otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.5.4 != (Equality Operator)

Description

Determines whether two specified instances of PrecisionDateTime are not equal.

.NET Prototype

```
public static bool operator ==(PrecisionDateTime pdt1,
                               PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	A PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	A PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	`false` if `pdt1` and `pdt2` represent the same date and time, or if they are both Not a Time; otherwise, `true`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

2.5.5 >= (Greater Than Or Equal To Operator)

Description

Determines whether one specified PrecisionDateTime is later than or equal to another specified PrecisionDateTime.

.NET Prototype

```
public static bool operator >=(PrecisionDateTime pdt1,
                               PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	A PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	A PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	`true` if `pdt1` and `pdt2` represent the same date and time, or if pdt1 is later than pdt2 otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If either pdt1 or pdt2, but not both, is NotATime, this method throws the Not A Time exception.

2.5.6 <= (Less Than Or Equal To Operator)

Description

Determines whether one specified PrecisionDateTime is earlier than or equal to another specified PrecisionDateTime.

.NET Prototype

```
public static bool operator <=(PrecisionDateTime pdt1,
                               PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	A PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	A PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	`true` if `pdt1` and `pdt2` represent the same date and time, or if pdt1 is earlier than pdt2 otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If either pdt1 or pdt2, but not both, is NotATime, this method throws the Not A Time exception.

2.5.7 > (Greater Than Operator)

Description

Determines whether one specified PrecisionDateTime is later than another specified PrecisionDateTime.

.NET Prototype

```
public static bool operator >(PrecisionDateTime pdt1,
                              PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	A PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	A PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	`true` if pdt1 is later than pdt2 otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If either pdt1 or pdt2 is NotATime, this method throws the Not A Time exception.

2.5.8 < (Less Than Operator)

Description

Determines whether one specified PrecisionDateTime is earlier than another specified PrecisionDateTime.

.NET Prototype

```
public static bool operator <(PrecisionDateTime pdt1,
                              PrecisionDateTime pdt2)
```

Parameters

Input	Description	Data Type
`pdt1`	A PrecisionDateTime.	`PrecisionDateTime`
`pdt2`	A PrecisionDateTime.	`PrecisionDateTime`

Output	Description	Data Type
`return value`	`true` if pdt1 is earlier than pdt2 otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If either pdt1 or pdt2 is NotATime, this method throws the Not A Time exception.

3. PrecisionTimeSpan Class

3.1 Overview

Instruments sometimes require a time interval which exceeds the resolution of the .NET FrameworkTimeSpan struct. To address these cases, IVI.NET provides the PrecisionTimeSpan class, which provides a level of resolution similar to that defined by the IEEE 1588 standard.

3.1.1 Details

PrecisionTimeSpan supports a range of intervals from -1.0e+13 to +1.0e+13 seconds.

The PrecisionTimeSpan SecondsIntegral and SecondsFractional properties always return digits to the left of the decimal point in SecondsIntegral and digits to the right of the decimal point as SecondsFractional according to this convention.

Note that resolution may be lost when converting to other types due to the high precision nature of the PrecisionTimeSpan class. In particular, properties and methods that convert from PrecisionTimeSpan to less precise types may result in a loss of resolution.

For values close to zero, the resolution of a PrecisionTimeSpan is the same as the resolution of a double.

3.1.2 Relationship to .NET Framework TimeSpan Struct

The PrecisionTimeSpan class is modeled on the .NET Framework System.TimeSpan struct. The primary differences between the two are (1) TimeSpan only provides resolution to 100 nanoseconds, while PrecisionTimeSpan provides resolution smaller than one Femtosecond, and (2) TimeSpan can only represent intervals as large as Int64.MaxValue ticks, while PrecisionTimeSpan can represent intervals as large as 1.0e+13 seconds.

Since PrecisionTimeSpan is targeted at a test and measurement market, it does not try to duplicate all of the general purpose features of TimeSpan. For example, PrecisionTimeSpan does not support the full variety of TimeSpan format specifiers, globalization, or serialization.

3.1.3 Relationship to LXISync

IVI 3.15: IviLxiSync Specification includes techniques that allow instrument operation to be triggered at given times and for timestamps to be associated with measured data. *IVI 3.15: IviLxiSync Specification* also specifies a particular data format (a pair of double values) that is used to contain a high-resolution time stamp value. The first double is Time Seconds and the second double is Time Fraction. To allow IVI.NET drivers to interoperate with LXI sync times, the PrecisionTimeSpan class follows a similar model of representing time with two doubles - Seconds Integral and Seconds Fractional.

3.1.4 Inherited Interfaces

The PrecisionTimeSpan class derives from the following interfaces:

- IComparable
- IComparable<PrecisionTimeSpan>
- IEquatable<PrecisionTimeSpan>

IComparable defines “public int CompareTo(object obj)”. Refer to Section 3.4.3, CompareTo, for more details.

IComparable<PrecisionTimeSpan> defines “public int CompareTo(PrecisionTimeSpan other)”. Refer to Section 3.4.3, CompareTo, for more details.

IEquatable<PrecisionTimeSpan> defines “public bool Equals(PrecisionTimeSpan other)”. Refer to Section 3.4.5, Equals, for more details.

3.2 PrecisionTimeSpan Constructors

Description

PrecisionTimeSpan has two types of constructors. TimeSpan-based constructors include a TimeSpan parameter and, optionally, additional parameters to support additional resolution. Seconds-based constructors include one or more parameters that represent the number of seconds in the interval, to femtosecond resolution.

Description – TimeSpan-based Constructors

TimeSpan-based constructors all take a .NET Framework TimeSpan parameter. Since the TimeSpan class only supports a resolution of 100 nanoseconds, there is an overload that allows the user to add fractional seconds. The double allows for femtosecond resolution.

.NET Prototypes – TimeSpan-based Constructors

```
public PrecisionTimeSpan(TimeSpan span)

public PrecisionTimeSpan(TimeSpan span,
                         Double deltaSeconds)
```

Description – Seconds-based Constructors

Seconds-based constructors all take one or two parameters, where the parameter units are seconds. If the total number of seconds exceeds the range of PrecisionTimeSpan, the constructor throws an exception.

For the constructor with two double paramaters (secondsIntegral and secondsFractional), the constructor accepts a secondsIntegral parameter with a fractional part and a secondsFractional parameter (that must be between 0.0 and 1.0), and will add the two together to get the correct interval. While specifying secondsIntegral with a fractional part is not encouraged, this behavior avoids throwing an exception when the data can be interpreted in a meaningful way. The pupose for using two doubles is to allow greater precision than can be expressed with one double.

.NET Prototypes – Seconds-based Constructors

```
public PrecisionTimeSpan(Double secondsIntegral,
                         Double secondsFractional)

public PrecisionTimeSpan(String seconds)
```

Parameters

Input	Description	Data Type
`span`	A .NET Framework TimeSpan object. The number of ticks (100-nanosecond intervals) in the TimeSpan object is used to initialize the Precision TimeSpan object.	`TimeSpan`
`deltaSeconds`	The number of seconds to add to the `span` object used to initialize the precision time span. Must be positive.	`Double`
`seconds`	The total number of seconds in the interval.	`Decimal`
`seconds`	The total number of seconds in the interval, expressed as a string.	`String`

`secondsIntegral`	The number of seconds in the interval. See section 3.1.3, *Details* for more details.	`Double`
`secondsFractional`	A fractional number of seconds (greater than or equal to 0.0, and less than 1.0) added to the time represented by `lxiBaseSeconds`. This parameter provides for femtosecond resolution to the right of the decimal. Resolution finer than femtoseconds will be rounded.	`Double`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

An exception will be thrown if a constructor attempts to construct an interval that is out of range.

An exception will be thrown is the values for the seconds parameters are out of range.

3.3 PrecisionTimeSpan Properties

The PrecisionTimeSpan class defines the following properties:

- Days
- Femtosecond
- Hours
- MaxValue
- Microseconds
- Milliseconds
- Minutes
- MinValue
- Nanoseconds
- Picoseconds
- Seconds
- SecondsFractional
- SecondsTotal
- TotalDays
- TotalHours
- TotalMilliseconds
- TotalMinutes
- TotalSeconds
- Zero

This section describes the behavior and requirements of each property.

3.3.1 Days

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Days;
```

Description

The days component of the time span represented by this instance, expressed as an integer.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.2 Femtoseconds

Data Type	Access
`Int64`	RO

.NET Prototype

```
public Int64 Femtoseconds;
```

Description

The femtosecond component of the time span represented by this instance, expressed as a value between -999,999,999,999,999 and 999,999,999,999,999.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.3 Hours

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Hours;
```

Description

The hours component of the time span represented by this instance, expressed as an integer value between -23 and 23.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.4 Max Value

Data Type	Access
`PrecisionTimeSpan`	RO, static

.NET Prototype

```
public static readonly PrecisionTimeSpan MaxValue;
```

Description

The largest possible value of PrecisionTimeSpan. This property is read-only. The value of this constant is 1.0e+13 seconds, exactly.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.5 Microseconds

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Microseconds;
```

Description

The fractional seconds represented as microseconds for the time span represented by this instance, expressed as a value between -999,999 and 999,999.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.6 Milliseconds

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Milliseconds;
```

Description

The fractional seconds represented as milliseconds for the time span represented by this instance, expressed as a value between -999 and 999.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.7 Minutes

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Minutes;
```

Description

The minutes component of the time span represented by this instance, expressed as an integer value between -59 and 59.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.8 Min Value

Data Type	Access
`PrecisionTimeSpan`	RO, static

.NET Prototype

```
public static readonly PrecisionTimeSpan MinValue;
```

Description

The smallest possible value of PrecisionTimeSpan. This property is read-only. The value of this constant is –1.0e+13 seconds, exactly.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.9 Nanoseconds

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Nanoseconds;
```

Description

The fractional seconds represented as nanoseconds for the time span represented by this instance, expressed as a value between -999,999,999 and 999,999,999.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.10 Picoseconds

Data Type	Access
`Int64`	RO

.NET Prototype

```
public Int64 Picoseconds;
```

Description

The fractional seconds represented as picoseconds for the time span represented by this instance, expressed as a value between -999,999,999,999 and 999,999,999,999.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.11 Seconds

Data Type	Access
`Int32`	RO

.NET Prototype

```
public Int32 Seconds;
```

Description

The seconds component of the time span represented by this instance, expressed as an integer value between -59 and 59.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.12 SecondsFractional

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double SecondsFractional;
```

Description

A value that represents the fractional portion (remainder) of the total number of seconds in the interval, expressed as a value between -1 and 1, exclusive.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.13 SecondsIntegral

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double SecondsIntegral;
```

Description

A value that represents the integer portion of the total number of seconds in the interval. Any digits to the right of the decimal point are truncated.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.14 TotalDays

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double TotalDays;
```

Description

A value that represents the value of the current PrecisionTimeSpan object expressed in whole and fractional days.

One value of type Double connot represent the full resolution of a PrecisionTimeSpan object. The full resolution is represented by the SecondsIntegral and SecondsFractional properties taken together.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.15 TotalHours

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double TotalDays;
```

Description

A value that represents the value of the current PrecisionTimeSpan object expressed in whole and fractional hours.

One value of type Double connot represent the full resolution of a PrecisionTimeSpan object. The full resolution is represented by the SecondsIntegral and SecondsFractional properties taken together.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.16 TotalMilliseconds

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double TotalDays;
```

Description

A value that represents the value of the current PrecisionTimeSpan object expressed in whole and fractional milliseconds.

One value of type Double connot represent the full resolution of a PrecisionTimeSpan object. The full resolution is represented by the SecondsIntegral and SecondsFractional properties taken together.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.17 TotalMinutes

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double TotalDays;
```

Description

A value that represents the value of the current PrecisionTimeSpan object expressed in whole and fractional minutes.

One value of type Double connot represent the full resolution of a PrecisionTimeSpan object. The full resolution is represented by the SecondsIntegral and SecondsFractional properties taken together.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.18 TotalSeconds

Data Type	Access
`Double`	RO

.NET Prototype

```
public Double TotalDays;
```

Description

A value that represents the value of the current PrecisionTimeSpan object expressed in whole and fractional seconds.

One value of type Double connot represent the full resolution of a PrecisionTimeSpan object. The full resolution is represented by the SecondsIntegral and SecondsFractional properties taken together.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.3.19 Zero

Data Type	Access
`PrecisionTimeSpan`	RO, static

.NET Prototype

```
public static PrecisionTimeSpan Zero { get }
```

Description

The PrecisionTimeSpan value that is equivalent to 0.0 seconds.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4 PrecisionTimeSpan Methods

The PrecisionTimeSpan class defines the following methods:

- Add
- Compare
- CompareTo
- Duration
- FromDays
- FromHours
- FromMicroseconds
- FromMilliseconds
- FromMinutes
- FromNanoseconds
- FromPicoseconds
- FromSeconds
- FromTimeSpan
- Multiply
- Negate
- Subtract
- ToTimeSpan

The PrecisionTimeSpan class overrides the following methods:

- Object.Equals
- Object.GetHashCode
- Object.ToString

This section describes the behavior and requirements of each of the above methods.

3.4.1 Add

Description

Adds the value of the specified TimeSpan to the value of this instance.

This method does not change the value of this PrecisionTimeSpan. Instead, a new PrecisionTimeSpan is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionTimeSpan Add(PrecisionTimeSpan pts)

public PrecisionTimeSpan Add(TimeSpan ts)
```

Parameters

Input	Description	Data Type
`pts`	A precision time span to be added to the precision time span of the current object.	`PrecisionTimeSpan`
`ts`	A time span to be added to the precision time span of the current object.	`TimeSpan`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance that is the sum of this instance and pts or ts.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.2 Compare

Description

Compares two instances of PrecisionTimeSpan and returns an indication of their relative values.

.NET Prototype

```
public static int Compare(PrecisionTimeSpan pts1,
                          PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A precision time span.	`PrecisionTimeSpan`
`pts2`	A precision time span.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	A signed number indicating the relative values of pts1 and pts2. If the return value is less than zero, then pts1 falls before pts2. If the return value is equal to zero, then pts1 and pts2 are the same date and time. If the return value is greater than zero, then pts1 falls after pts2.	`Int32`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.3 CompareTo

Description

Compares this instance to a specified PrecisionTimeSpan object and returns an indication of their relative values. Any instance of PrecisionTimeSpan, regardless of its value, is considered greater than a null reference.

.NET Prototype

```
public int CompareTo(PrecisionTimeSpan other)

public int CompareTo(object obj)
```

Parameters

Input	Description	Data Type
`other`	A precision time span.	`PrecisionTimeSpan`
`obj`	A boxed PrecisionTimeSpan object to compare, or a null reference.	`object`

Output	Description	Data Type
`Return value`	A signed number indicating the relative values of pts1 and pts2. If the return value is less than zero, then pts1 falls before pts2. If the return value is equal to zero, then pts1 and pts2 are the same date and time. If the return value is greater than zero, then pts1 falls after pts2.	`Int32`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.4 Duration

Description

Returns the absolute value of this instance. This method does not change the value of this PrecisionTimeSpan. Instead, a new PrecisionTimeSpan is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionTimeSpan Duration()
```

Parameters

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance that is the absolute value of this instance.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.5 Equals

Description

Determines whether two specified instances of PrecisionTimeSpan represent the same precision time span.

.NET Prototype

```
public override bool Equals(PrecisionTimeSpan other)
```

Parameters

Input	Description	Data Type
`other`	A boxed PrecisionTimeSpan object to compare, or a null reference.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	True if this instance and the 'obj' instance represent the same precision time span.	`Boolean`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.6 FromDays

Description

Returns a new PrecisionTimeSpan instance with a length in days equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromDays(Double days)
```

Parameters

Input	Description	Data Type
`days`	A number of days.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in days equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.7 FromHours

Description

Returns a new PrecisionTimeSpan instance with a length in hours equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromHours(Double hours)
```

Parameters

Input	Description	Data Type
`hours`	A number of hours.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in hours equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.8 FromMicroseconds

Description

Returns a new PrecisionTimeSpan instance with a length in microseconds equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromMicroseconds(
                                       Double microseconds)
```

Parameters

Input	Description	Data Type
`microseconds`	A number of microseconds.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in microseconds equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.9 FromMilliseconds

Description

Returns a new PrecisionTimeSpan instance with a length in milliseconds equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromMilliseconds(
                                              Double milliseconds)
```

Parameters

Input	Description	Data Type
`milliseconds`	A number of milliseconds.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in milliseconds equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.10 FromMinutes

Description

Returns a new PrecisionTimeSpan instance with a length in minutes equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromMinutes (Double minutes)
```

Parameters

Input	Description	Data Type
`minutes`	A number of minutes.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in minutes equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.11 FromNanoseconds

Description

Returns a new PrecisionTimeSpan instance with a length in nanoseconds equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromNanoseconds(
                                             Double nanoseconds)
```

Parameters

Input	Description	Data Type
`nanoseconds`	A number of nanoseconds.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in nanoseconds equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.12 FromPicoseconds

Description

Returns a new PrecisionTimeSpan instance with a length in picoseconds equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromPicoseconds(
                                            Double picoseconds)
```

Parameters

Input	Description	Data Type
`picoseconds`	A number of picoseconds.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in picoseconds equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.13 FromSeconds

Description

Returns a new PrecisionTimeSpan instance with a length in seconds equal to the input parameter.

.NET Prototype

```
public static PrecisionTimeSpan FromSeconds(Double seconds)
```

Parameters

Input	Description	Data Type
`seconds`	A number of seconds.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance with a length in seconds equal to the input parameter.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.14 Multiply

Description

Multiplies the value of this TimeSpan by an integer. This method does not change the value of this PrecisionTimeSpan. Instead, a new PrecisionTimeSpan is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionTimeSpan Multiply(Double factor)
```

Parameters

Input	Description	Data Type
`factor`	The integer by which this instance is to be multiplied.	`Double`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance that is the product of this instance and factor.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.15 Negate

Description

Negates this instance. This method does not change the value of this PrecisionTimeSpan. Instead, a new PrecisionTimeSpan is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionTimeSpan Negate()
```

Parameters

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance that is the negative of this instance.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.16 Plus

Description

Unary plus. This method does not change the value of this PrecisionTimeSpan. Instead, a new PrecisionTimeSpan is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionTimeSpan Plus()
```

Parameters

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan with the same value as this instance.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.17 Subtract

Description

Subtracts the value of the specified PrecisionTimeSpan or TimeSpan from the value of this instance. This method does not change the value of this PrecisionTimeSpan. Instead, a new PrecisionTimeSpan is returned whose value is the result of this operation.

.NET Prototype

```
public PrecisionTimeSpan Subtract(PrecisionTimeSpan pts)

public PrecisionTimeSpan Subtract(TimeSpan ts)
```

Parameters

Input	Description	Data Type
`pts`	A PrecisionTimeSpan.	`PrecisionTimeSpan`
`ts`	A TimeSpan.	`TimeSpan`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan instance that is the difference of this instance and pts or ts.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.18 ToString

Description

Converts the value of the current PrecisionTimeSpan object to its equivalent string representation.

In most cases, this string will be equivalent to `this.ToDateTime().ToString()`, with the addition of fractional seconds to femtosecond resolution whenever long times are used.

The format parameter is a PrecisionTimeSpan format string. A format string consists of one or more custom PrecisionTimeSpan format specifiers, and that format string defines the text representation of a PrecisionTimeSpan object that is produced by a formatting operation.

The following subset of PrecisionTimeSpan format specifiers is allowed for PrecisionTimeSpan:

Days	"d"	Represents the number of days in the time span. A single-digit day is formatted with a leading zero.
Hours	"hh"	Represents the hour as a number from 01 through 12, that is, the hour as represented by a 12-hour clock. A single-digit hour is formatted with a leading zero.
Minutes	"mm"	Represents the minutes as a number from 00 through 59. A single-digit minute is formatted with a leading zero.
Seconds	"ss"	Represents the seconds as a number from 00 through 59. A single-digit second is formatted with a leading zero.
	"f " "ff" "f…f"	N 'f' characters, where N is from 1 to 15, represent the N most significant digits of the seconds fraction. Note that this is an extension of the DateTime format specifier, where N cannot be greater than 7.
Special	":"	The time separator defined in the current System.Globalization.DateTimeFormatInfo.TimeSeparator property that is used to differentiate hours, minutes, and seconds.
	"'"	Quoted string (apostrophe). Displays the literal value of any string between two apostrophe (') characters.
	"%c"	Represents the result associated with a custom format specifier "*c*", when the custom DateTime format string consists solely of that custom format specifier. For example, to use the "d" custom format specifier by itself, specify "%d".

For any other character, the literal value of the character is copied to the result string, and does not affect formatting.

.NET Prototype

```
public string ToString(string format)
```

Parameters

Input	Description	Data Type
`format`	A PrecisionTimeSpan format string.	`IFormatProvider`

Output	Description	Data Type
`return value`	A string representation of the value of the current PrecisionTimeSpan object.	`string`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.19 ToTimeSpan

Description

Returns a new TimeSpan with the value of this instance of PrecisionTimeSpan, with femtoseconds rounded to the nearest 100 nanoseconds.

.NET Prototype

```
public TimeSpan ToTimeSpan()
```

Parameters

Output	Description	Data Type
`Return value`	A TimeSpan whose value is the date and time represented by this instance with femtoseconds rounded to the nearest 100 nanoseconds.	`TimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.20 IConvertible.ToDouble

Description

Returns a Double with the value returned by the TotalSeconds property of this instance of PrecisionTimeSpan.

.NET Prototype

```
string IConvertible.ToDouble(IFormatProvider provider)
```

Parameters

Input	Description	Data Type
`provider`	A format provider.	`IFormatProvider`

Output	Description	Data Type
`return value`	A Double with the value returned by the TotalSeconds property of this instance of PrecisionTimeSpan.	`Double`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.21 IConvertible.ToString

Description

Converts the value of the current PrecisionTimeSpan object to its equivalent string representation using the default "d.hh:mm:ss.fffffffffffffff" format specifier. Refer to section 3.4.18, ToString, for a description of the PrecisionTimeSpan format specifiers.

.NET Prototype

```
string IConvertible.ToString(IFormatProvider provider)
```

Parameters

Input	Description	Data Type
`provider`	A format provider.	`IFormatProvider`

Output	Description	Data Type
`return value`	The string representation of this instance, formatted using the default "d.hh:mm:ss.fffffffffffffff" format specifier.	`string`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.22 Object.Equals

Description

Determines whether two specified instances of PrecisionTimeSpan represent the same precision time span.

.NET Prototype

```
public override bool Equals(object obj)
```

Parameters

Output	Description	Data Type
`obj`	A boxed PrecisionTimeSpan object to compare, or a null reference.	`Object`
`Return value`	True if this instance and the 'obj' instance represent the same precision time span.	`Boolean`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.23 Object.GetHashCode

Description

Returns the hash code for this instance.

.NET Prototype

```
public override int GetHashCode()
```

Parameters

Output	Description	Data Type
`Return value`	A 32-bit signed integer hash code.	`Int32`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.4.24 Object.ToString

Description

Converts the value of the current PrecisionTimeSpan object to its equivalent string representation using the default "d.hh:mm:ss.fffffffffffffff" format specifier. Refer to section 3.4.18, ToString, for a description of the PrecisionTimeSpan format specifiers.

.NET Prototype

```
public override string ToString()
```

Parameters

Output	Description	Data Type
`Return value`	The formatted time span string.	`string`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5 PrecisionTimeSpan Operators

The PrecisionTimeSpan class defines the following operators:

- + (Unary)
- - (Unary)
- +
- -
- *
- ==
- !=
- >=
- <=
- >
- <

This section describes the behavior and requirements of each operator.

3.5.1 + (Unary Addition Operator)

Description

Returns the same instance of PrecisionTimeSpan, unchanged.

.NET Prototype

```
public static PrecisionTimeSpan operator +(PrecisionTimeSpan pts)
```

Parameters

Input	Description	Data Type
`pts`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	The same instance of PrecisionTimeSpan as pts.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.2 - (Unary Subtraction Operator)

Description

Returns a new PrecisionTimeSpan with the same numeric value as pts, but the opposite sign.

.NET Prototype

```
public static PrecisionTimeSpan operator -(PrecisionTimeSpan pts)
```

Parameters

Input	Description	Data Type
`pts`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	A new PrecisionTimeSpan with the same numeric value as pts, but the opposite sign.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.3 + (Addition Operator)

Description

Adds a PrecisionTimeSpan or TimeSpan to another PrecisionTimeSpan or Time Span, yielding a new PrecisionTimeSpan.

.NET Prototype

```
public static PrecisionTimeSpan operator +(
                                    PrecisionTimeSpan pts1,
                                    PrecisionTimeSpan pts2)

public static PrecisionTimeSpan operator +(
                                    PrecisionTimeSpan pts,
                                    TimeSpan ts)

public static PrecisionTimeSpan operator +(
                                    TimeSpan ts,
                                    PrecisionTimeSpan pts)
```

Parameters

Input	Description	Data Type
`pts, pts1, pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`ts`	A TimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	The sum of the two operands.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.4 - (Subtraction Operator)

Description

Subtracts a PrecisionTimeSpan or TimeSpan from another PrecisionTimeSpan or TimeSpan, yielding a new PrecisionTimeSpan.

.NET Prototype

```
public static PrecisionTimeSpan operator -(
                                    PrecisionTimeSpan pts1,
                                    PrecisionTimeSpan pts2)

public static PrecisionTimeSpan operator -(
                                    PrecisionTimeSpan pts,
                                    TimeSpan ts)

public static PrecisionTimeSpan operator -(
                                    TimeSpan ts,
                                    PrecisionTimeSpan pts)
```

Parameters

Input	Description	Data Type
`pts, pts1, pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`ts`	A TimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	The difference between the first and second operands.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.5 * (Multiplication Operator)

Description

Multiplies a PrecisionTimeSpan by a numeric multiplier, yielding a new PrecisionTimeSpan whose length is the product of the multiplier and the number of seconds in the original time span.

If necessary, the result is rounded to the nearest Femtosecond. Results that are exactly exactly .5 femtoseconds from a valid whole femtosecond are rounded up.

.NET Prototype

```
public static PrecisionTimeSpan operator *(PrecisionTimeSpan pts,
                                            Double factor)
```

Parameters

Input	Description	Data Type
`pts`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`factor`	The numeric multiplier.	`Double`

Output	Description	Data Type
`Return value`	A PrecisionTimeSpan whose length is the product of the multiplier and the number of seconds in the original time span.	`PrecisionTimeSpan`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.6 == (Equality Operator)

Description

Determines whether two specified instances of PrecisionTimeSpan are equal.

.NET Prototype

```
public static bool operator ==(PrecisionTimeSpan pts1,
                               PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	`true` if pts1 and pts2 represent the same time span; otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.7 != (Equality Operator)

Description

Determines whether two specified instances of PrecisionTimeSpan are not equal.

.NET Prototype

```
public static bool operator !=(PrecisionTimeSpan pts1,
                               PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	`true` if pts1 and pts2 represent different time spans; otherwise, `false`.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.8 >= (Greater Than Or Equal To Operator)

Description

Determines whether one specified PrecisionTimeSpan is greater than or equal to another specified PrecisionTimeSpan.

.NET Prototype

```
public static bool operator >=(PrecisionTimeSpan pts1,
                               PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	`true if pts1 and pts2 represent the same time span, or if pts1 is greater than pts2, otherwise false.`	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.9 <= (Less Than Or Equal To Operator)

Description

Determines whether one specified PrecisionTimeSpan is less than or equal to another specified PrecisionTimeSpan.

.NET Prototype

```
public static bool operator <=(PrecisionTimeSpan pts1,
                               PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	true if pts1 and pts2 represent the same time span, or if pts1 is less than pts2, otherwise false.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.10 > (Greater Than Operator)

Description

Determines whether one specified PrecisionTimeSpan is greater than another specified PrecisionTimeSpan.

.NET Prototype

```
public static bool operator >(PrecisionTimeSpan pts1,
                              PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	true if pts1 is greater than pts2, otherwise false.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

3.5.11 < (Less Than Operator)

Description

Determines whether one specified PrecisionTimeSpan is less than another specified PrecisionTimeSpan.

.NET Prototype

```
public static bool operator < (PrecisionTimeSpan pts1,
                               PrecisionTimeSpan pts2)
```

Parameters

Input	Description	Data Type
`pts1`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`
`pts2`	A PrecisionTimeSpan operand.	`PrecisionTimeSpan`

Output	Description	Data Type
`Return value`	true if pts1 is less than pts2, otherwise false.	`bool`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4. Common Properties and Methods of Waveform and Spectrum Interfaces

4.1 Overview

This section describes methods and properties that are common to the spectrum and waveform interfaces:

Spectrum interfaces

ISpectrum

ISpectrumMemory

Waveform interfaces

IWaveform

IWaveformMemory

Drivers that produce or consume a waveform or spectrum type use these common APIs to simplify working with those objects.

A waveform is used for any time-varying value. Therefore, the data array in a time waveform may refer to power, voltage, wavelength, or other sundry values. A spectrum is used for any frequency-varying value. Therefore, the data array in a spectrum may refer to power, voltage, wavelength, or other sundry values.

Unless documented otherwise, all methods that return a waveform/spectrum shall implement the appropriate waveform or spectrum interface defined here.

4.2 How to use Waveform and Spectrum Types

Waveforms and spectrums are composed of:

Data array This is an array that contains the explicit data. Waveform/Spectrum objects are generic so that the type of the data array can be chosen appropriately for the application. The data array can be composed of compound types such as a MinMax struct with two values.

Implicit Axis The information about the index of the explicit data. For a waveform this describes the time corresponding to the waveform data. For a spectrum, this describes the frequency corresponding to the spectrum data.

Utility Methods The interfaces have methods that are generally useful for working with the data including a timestamp, scaling information for integers and other conveniences.

4.2.1 The Location of the Waveform or Spectrum in the data array

Where there are not performance penalties, the first element in the data array should be the first element in the Waveform/Spectrum. However, for certain hardware implementations, the alignment required by the DMA hardware may differ from the memory allocation alignment. In circumstances such as this, Waveform/Spectrum producers are permitted to start the actual waveform/spectrum data within the data array at an offset indicated by FirstValidPoint property.

In any case, the number of data points within the data array is indicated by the ValidPointCount property.

4.2.2 Methods that return a Waveform or Spectrum

With the exception of factory methods, methods that return a waveform/spectrum shall also be passed a waveform/spectrum and will have the following behaviors based on the input waveform:

If the method receives a null reference: The method will consult the type of the null reference and allocate a new waveform/spectrum of the same type with an appropriate extent for the current configuration of the driver. The new waveform/spectrum is returned to the client. The driver may allocate more memory than necessary for the data array if the larger size has the potential to provide some present or future efficiency benefit, that is, the Capacity may exceed the ValidPointCount.

If the method receives a non-null reference with zero sized data: This permits the client to choose the concrete class that implements the Waveform/Spectrum but defer to the driver for the size and creation of the data array. This may result in sub-optimal performance if the waveform/spectrum was not of the class that the driver prefers. If the data array is not of a supported size or type, the driver shall throw the Invalid Waveform Data Type or Invalid Spectrum Data Type exception. The driver is permitted to allocate new memory or use memory from an existing source for the data array.

If the method receives a previous instantiated waveform/spectrum object: The driver shall fill the data into the object passed. The driver is required to use the allocated memory and is not permitted to use an existing or allocate a new block of memory. If the measured data exceeds the available space the driver will throw the Data Array Too Small exception. If the measured data matches or is less than the available space the data is filled in and the ValidPointCount shall return an appropriate value. The Capacity shall remain the same. Note that if the alignment requirements dictate that the array be filled from some firstValidPoint other than zero, then a block of memory that appears to be large enough just based on the Capacity may be too small.

If the method does not support the data array type specified in this API, it shall throw Invalid Waveform Data Type or Invalid Spectrum Data Type exception.

Methods that return a waveform or spectrum shall not alter the driver configuration based on the implicit axis information or the pre-existing data values. That is, these values shall be filled in by the method and the values in the input object shall not change the driver configuration.

In addition to setting the data array, the returned object shall set up the waveform/spectrum object so that is has appropriate responses for the properties:

StartTime	May be set to Zero if not meaningful for this object.
IntervalPerPoint	Set to zero if not meaningful for this object
TotalTime	Set to zero if not meaningful for this object
EndTime	May be set to Zero if not meaningful for this object.
TriggerTime	May be set to <NotATime> if not meaningful for this object.

If time has no meaning for a waveform, then TriggerTime shall be set to <NotATime>, IntervalPerPoint is set to zero, and StartTime is set to zero.

4.2.3 Methods or properties that receive a Waveform or Spectrum

Methods that receive a waveform/spectrum as an input value are required to evaluate the full extent of the object, including the implicit axis information and the full data array (up to the valid point count, not the capacity). If the object is inappropriate to the API the driver shall throw an appropriate exception.

In some cases, values such as the StartTime or TriggerTime may not be applicable to the API (for instance, if the waveform represents a periodic waveform to be generated by an arbitrary waveform generator). However, if there are reasonable interpretations of the implicit axis data in the waveform (such as the IntervalPerPoint when used with an arbitrary waveform generator) they shall be used by the driver or an appropriate exception thrown. An exception shall not be thrown in cases where the supplied values are not applicable if the applicable values are all implementable.

4.2.4 Scaled array data

If the data array is based on an integer template type, it does not directly represent a physical quantity. If the data array is based on the Double template type, it does directly represent a physical quantity.

To obtain the physical quantity from a data array based on an integer template type, the user must scale the data using the following formula:

Physical Quantity = ArrayElement * Scale + Offset

For example:

```
Double value = (Double)aWaveform[i] * aWaveform.Scale + aWaveform.Offset;
```

If the Scale and Offset are not used they shall return the values 1 and 0 respectively.

The Scale and Offset shall not be used for floating point data.

4.2.5 General Requirements regarding IWaveform and ISpectrum interfaces

Although required to accept any object that implements the IviDriver.IWaveform<T> or IviDriver.ISpectrum<T> interfaces with an appropriate template type, individual drivers are permitted to provide their own factory methods that create objects optimized for the operation of that driver. An example of this would be a driver that allocates shared memory for the data array. Such an object would not necessarily work correctly if passed to a driver other than the one that created it.

If a client allocates an Ivi.Driver.Waveform and passes it to an instrument interface, they are assured that no ties remain to it from the driver when it is returned from the driver. However, if the object was allocated by the driver, further interaction with the driver could result in the data in the object changing. Clients worried about this should either copy the result to an Ivi.Driver.Waveform before additional driver access, or pass an Ivi.Driver.Waveform to the driver to get the data.

4.2.6 Data Array Types

IVI specifications, classes or specific drivers that produce or consume objects that implement or extend IWaveform<T> or ISpectrum<T> interfaces shall document the type parameters they support.

4.3 Waveform and Spectrum Common Properties

The waveform and spectrum interfaces use the following properties:

- Item
- Capacity
- EndTime
- IntervalPerPoint
- Offset
- Scale
- StartTime
- TotalTime
- TriggerTime
- ValidPointCount

This section describes the common behavior and requirements of each property.

4.3.1 Item

Data Type	Access
`T`	RW

.NET Prototype

In .NET the property name is Item. The syntax for this property is:

```
T this[Int64 Index]
```

That is, the array access operator can be directly applied to a waveform to access elements of the data array. T is the type of the data element in the waveform.

Description

This returns the data element at the specified index. Note that for scaled (that is, integer) types, the scaling must be applied to the returned data element to convert it to a physical value.

Drivers shall document the template types that they support.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.2 Capacity

Data Type	Access
`Int64`	RW

.NET Prototype

```
Int64 Capacity { get; set;  }
```

Description

Capacity is the number of elements that the waveform/spectrum array can store. Note that Valid Point Count may be used to get the actual number of elements in this waveform/spectrum. When the value of Capacity is set explicitly, the internal array is also reallocated to accommodate the specified capacity, and all the elements are copied.

In some cases, the data array may be stored using a method that does not support dynamic allocation, such as a memory mapped acquisition buffer. In these cases, attempts to explicitly set Capacity will throw an exception.

If a reduction in capacity would cause FirstValidPoint to be invalid, it will set both ValidPointCount and FirstValidPoint to zero. If a reduction in capacity would cause only ValidPointCount to be invalid, ValidPointCount will be reduced to fit within the capacity.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

If the implementation cannot reallocate the internal storage array, the method shall throw a System.InsufficientMemory exception.

4.3.3 ContainsInvalidElement

Data Type	Access
`Boolean`	RW

.NET Prototype

```
Boolean ContainsInvalidElement { get; set; }
```

Description

ContainsInvalidElement indicates that one or more points in the waveform/spectrum array are not valid. For instance, the signal was not sampled at this point.

If the elements are composed of a floating point type, ContainsInvalidElement shall be true if and only if at least one element within the valid range of elements is set to NaN.

If valid point count is zero, the value shall be false.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If the elements are not composed of floating point types, setting and reading ContainsInvalidElement are permitted to throw an Invalid Operation exception.
- If the implementation does not support changing ContainsInvalidElement (i.e. it is determined automatically), setting ContainsInvalidElement is permitted to throw an Operation Not Supported exception.

4.3.4 ContainsOutOfRangeElement

Data Type	Access
`Boolean`	RW

.NET Prototype

```
Boolean ContainsOutOfRangeElement { get; set; }
```

Description

ContainsOutOfRangeElement indicates that one or more points in the waveform/spectrum array are out of range. That is, a value that is too large or to small to represent (for instance a large positive number). This is not intended to represent numbers that are too close to zero to represent.

If the elements are composed of a floating point type, ContainsOutOfRangeElement shall be true if and only if at least one element within the valid range of elements is set to +Inf or -Inf.

If valid point count is zero, the value shall be false.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If the elements are not composed of floating point types, setting and reading ContainsOutOfRangeElement are permitted to throw an Invalid Operation exception.
- If the implementation does not support changing ContainsOutOfRangeElement (i.e. it is determined automatically), setting ContainsOutOfRangeElement is permitted to throw an Operation Not Supported exception.

4.3.5 EndTime (waveform types)

Data Type	Access
`PrecisionTimeSpan`	R

.NET Prototype

```
PrecisionTimeSpan EndTime { get; }
```

Description

`EndTime` is the time between the last valid data point in the waveform and the `TriggerTime`. Positive values of `EndTime` indicate that it occurred after the trigger.

This value is set by the waveform `Configure` method.

If start time is `NotATime`, or the valid point count is zero, the value shall be `NotATime`.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.6 FirstValidPoint

Data Type	Access
`Int64`	RW

.NET Prototype

```
Int64 FirstValidPoint { get; set; }
```

Description

For waveforms/spectrums that contain invalid padding data at the beginning of the data array, FirstValidPoint indicates the first element in the data array with valid data. If there is no padding data at the beginning of the data array, FirstValidPoint will be zero.

If valid point count is 0, the value shall be zero. This value must not exceed capacity.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If the FirstValidPoint exceeds the capacity, it will throw a Operation Not Supported exception.

4.3.7 FrequencyStep (spectrum types)

Data Type	Access
`Double`	R

.NET Prototype

```
Double FrequencyStep { get; }
```

Description

Frequency step is the frequency difference in Hertz between subsequent points in the data array.

This value is set by the spectrum Configure method, and is defined as (StopFrequency - StartFrequency) / (ValidPointCount – 1).

If valid point count is 0 or 1, the value shall be zero..NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.8 Start Frequency (spectrum types)

Data Type	Access
`Double`	R

.NET Prototype

```
Double StartFrequency { get; }
```

Description

Start frequency is the frequency in Hertz of the first valid data point (that is the data point at index FirstValidPoint) in the data array.

This value is set by the spectrum Configure method.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.9 Stop Frequency (spectrum types)

Data Type	Access
`Double`	R

.NET Prototype

```
Double StopFrequency { get; }
```

Description

Stop frequency is the frequency in Hertz of the final valid data point in the data array (that is the data point at index FirstValidPoint+ValidPointCount-1).

This value is set by the spectrum Configure method.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.10 IntervalPerPoint (waveform types)

Data Type	Access
`PrecisionTimeSpan`	R

.NET Prototype

```
PrecisionTimeSpan IntervalPerPoint { get; }
```

Description

Interval per point is the amount of time between data points in the data array.

This value is set by the waveform `Configure` method, and cannot be a negative value. A value of Zero indicates that it is not meaningful for this object..NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.11 Offset

Data Type	Access
`Double`	RW

.NET Prototype

```
Double Offset { get; set; }
```

Description

Offset is the offset to apply to scale integer values. To convert an integer data array element to a physical value first it is multiplied by the scale, and then the offset is added. The Scale and Offset properties are used to map the range and resolution of integers to physical values.

If the integers in the data array do not have an offset, the offset property is 0.

If the contents of the data array are floating point scalars, the offset property is set to 0.

If the contents of the data array are some other type the use of the offset is dependent on that driver and data type.

The value cannot be positive or negative infinity, or Not a Number.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If the data type is a floating point value, setting Offset is permitted to throw an Invalid Operation exception.

4.3.12 Scale

Data Type	Access
`Double`	RW

.NET Prototype

```
Double Scale { get; set; }
```

Description

Scale is the factor to apply to scale integer values. To convert an integer data array element to a physical value the element is multiplied by scale, and then the offset is added. The Scale and Offset properties are used to map the range and resolution of integers to physical values.

If the integers in the data array do not have an offset, the scale property is set to 1.

If the contents of the data array are floating point scalars, the scale property is set to 1.

If the contents of the data array are some other type the use of the scale is dependent on that driver and data type.

The value cannot be positive or negative infinity, or Not a Number.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If the data type is a floating point value, setting Offset is permitted to throw an Invalid Operation exception.

4.3.13 Start Time (waveform types)

Data Type	Access
`PrecisionTimeSpan`	R

.NET Prototype

```
PrecisionTimeSpan StartTime { get; }
```

Description

`StartTime` is the time between the first valid data point (that is the data point at index `FirstValidPoint`) in the waveform and the trigger. Positive values indicate that the `StartTime` occurred after the trigger.

This value is set by the waveform `Configure` method.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.14 TotalTime (waveform types)

Data Type	Access
`PrecisionTimeSpan`	R

.NET Prototype

```
PrecisionTimeSpan TotalTime { get; }
```

Description

`TotalTime` is the timespan represented by the valid points in the waveform. Numerically, it is equivalent to the `IntervalPerPoint * (ValidPointCount - 1)`. It is also numerically the `EndTime – StartTime`.

If valid point count is 0 or 1, the value shall be zero.

This value is set by the waveform `Configure` method.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.15 TriggerTime

Data Type	Access
`PrecisionDateTime`	RW

.NET Prototype

```
PrecisionDateTime TriggerTime { get; set; }
```

Description

`TriggerTime` is the time that this measurement was triggered.

Note that this differs from `StartTime` in that the trigger may have occurred at some time other than when the first data point was captured, as in pre-trigger or post-trigger applications.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3.16 ValidPointCount

Data Type	Access
`Int64`	RW

.NET Prototype

```
Int64 ValidPointCount { get; set; }
```

Description

`ValidPointCount` is the actual number of elements in the waveform/spectrum. Note that `Capacity` may be used to get the number of elements that the waveform/spectrum can store.

The `ValidPointCount` is the number of valid points starting with the element identified by `FirstValidPoint`.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

- If the validPointCount exceeds the data array size, the Valid Point Count Exceeds Capacity exception is thrown.

4.4 Waveform and Spectrum Common Methods

The waveform and spectrum interfaces use the following methods:

- Configure
- GetAllElements
- GetElements
- GetScaled
- PutElements

This section describes the common behavior and requirements of each of the above methods.

4.4.1 Configure (waveform types)

Description

The Configure method defines the time (implicit) axis and number of data points in the waveform.

Because of the interaction between these values, they are set as a group with this method or when the waveform is initially created.

The Configure call does not change any of the explicit data in the Waveform.

If the validPointCount is specified and it is different from the current value of Valid Point Count, the mechanism by which the array is extended or contracted is depends on the waveform class. Classes may optimize memory use by maintaining a validPointCount different from the capacity of the data array. The capacity of the waveform shall not change as a side effect of the Configure method.

If the validPointCount exceeds the data array size, the Valid Point Count Exceeds Capacity exception is thrown.

.NET Prototype

```
void Configure(PrecisionTimeSpan startTime,
               PrecisionTimeSpan intervalPerPoint,
               Int64 validPointCount,
               PrecisionDateTime triggerTime);

void Configure(PrecisionTimeSpan startTime,
               PrecisionTimeSpan intervalPerPoint,
               PrecisionDateTime triggerTime);

void Configure(PrecisionTimeSpan startTime,
               PrecisionTimeSpan intervalPerPoint,
               Int64 validPointCount);

void Configure(PrecisionTimeSpan startTime,
               PrecisionTimeSpan intervalPerPoint);

void Configure(PrecisionTimeSpan intervalPerPoint,
               Int64 validPointCount);

void Configure(PrecisionTimeSpan intervalPerPoint);
```

Parameters

Input	Description	Data Type
`startTime`	startTime is the time of the first data point in the waveform relative to the trigger time. See the StartTime property for more information. Default value is PrecisionTimeSpan.Zero.	`PrecisionTimeSpan`

`intervalPerPoint`	intervalPerPoint is the amount of time between data points in the waveform, and cannot be a negative. A value of zero indicates that it is not meaningful for this object.. See the InterverPerPoint property for more information. This value is required.	`PrecisionTimeSpan`
`triggerTime`	triggerTime is the time that this measurement was triggered. See the TriggerTime property for more information. Default value is NotATime	`PrecisionDateTime`
`validPointCount`	validPointCount is the number of elements in the waveform, and cannot be negative. See the ValidPointCount property for more information. Default behavior is to not change the current setting.	`Int64`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If the value of validPointCount is greater than the capacity of the waveform, the method shall throw the Valid Point Count Exceeds Capacity exception.

4.4.2 Configure (spectrum types)

Description

The Configure method fully defines the frequency (implicit) axis and number of data points in the spectrum.

Because of the interaction between these values, they are set as a group with this method or when the spectrum is initially created.

Ths Configure call does not change any of the explicit data in the spectrum if the extent of the array is not changed.

No changes to the underlying data array are made if the extent of the array is not changed by specifying a *validPointCount* that is different from the array currently in the spectrum.

If the validPointCount is specified and it is different from the current value of Valid Point Count, the mechanism by which the array is extended or contracted is driver-dependent. Regardless, the capacity of the spectrum shall not change as a side effect of this method. Classes may optimize memory use by maintaining a validPointCount different from the capacity of the data array.

If the validPointCount exceeds the data array size, the Valid Point Count Exceeds Capacity exception is thrown.

NET Prototype

```
void Configure(Double startFrequency,
               Double stopFrequency,
               PrecisionDateTime triggerTime,
               Int64 validPointCount);


void Configure(Double startFrequency,
               Double stopFrequency,
               PrecisionDateTime triggerTime);


void Configure(Double startFrequency,
               Double stopFrequency,
               Int64 validPointCount);


void Configure(Double startFrequency,
               Double stopFrequency);
```

Parameters

Input	Description	Data Type
`startFrequency`	startFrequency is the frequency of the first data point in the spectrum. See the StartFrequency property for more information.	`Double`
`stopFrequency`	stopFrequency is the frequency fo the last data pointin the spectrum. See the StopFrequency property for more information.	`Double`

`triggerTime`	triggerTime is the time that this measurement was triggered. See the TriggerTime property for more information. The default value for this property is <NotATime>	`PrecisionDateTime`
`validPointCount`	validPointCount is the number of elements in the waveform, and cannot be negative. See the ValidPointCount property for more information. Default behavior is to not change the current setting.	`Int64`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

- If the value of validPointCount is greater than the capacity of the waveform, the method shall throw the Valid Point Count Exceeds Capacity exception.

4.4.3 GetAllElements

Description

The GetAllElements method returns a copy of the entire data array in the template data type.

If the template data type is an integer, the returned data will not be scaled. That is, the data will not represent physical units until the scale and offset are applied.

.NET Prototype

```
T[] GetAllElements();
```

Parameters

Output	Description	Data Type
`Return value`	An array of the template type that contains the same values as the data in data array	`T[]`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.4.4 GetElements

Description

The GetElements method returns a copy of either all or part of the data array in the template type starting at the specified index and with the specified length.

If the template data type is an integer, the data will not be scaled. That is, the data will not represent physical units until the scale and offset are applied.

.NET Prototype

```
T[] GetElements(Int64 index, Int64 count);
```

Parameters

Input	Description	Data Type
`index`	The index in the Waveform that will be the first element in the returned array. That is, element zero in the returned array is at this index in the Waveform.	`Int64`
`count`	The number of elements to be returned.	`Int64`

Output	Description	Data Type
`Return value`	The data array taken from the Waveform, of the type of the Waveform data array.	`T[]`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.4.5 GetScaled

Description

Returns all or part of the data array as a Double. If the internal data array is an integer, the scaling is applied to the values in the returned array.

If only an index is provided, the data value at that point is returned instead of an array.

.NET Prototype

```
Double GetScaled(Int64 index);

Double[] GetScaled(Int64 index, Int64 count);
```

Parameters

Input	Description	Data Type
`index`	Either the index of the data element to return, or the index of the first element in the array to return.	`Int64`
`count`	If provided, the number of data points to include in the returned array. If not provided, a scalar value is returned.	`Int64`

Output	Description	Data Type
`Return value`	The specified data element from the array.	`Double`
	Data from the array.	`Double[]`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.4.6 PutElements

Description

PutElements copies data elements into either all or part of the data array.

Index is the first element of the data array to receive the new data.

The implicit axis of the waveform or spectrum is not changed by PutElements.

If the data array does not have sufficient capacity for the data, the capacity will be increased to accommodate the new data.

.NET Prototype

```
void PutElements(T[] data);

void PutElements(Int64 index, T[] data);

void PutElements(Int64 index, ArraySegment<T> segment);
```

Parameters

Input	Description	Data Type
`index`	The index of the first element of the data array to change.	`Int64`
`data`	The data to be placed into the array.	`T[]`
`segment`	The data to be placed into the array.	`ArraySegment<T>`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

If the array passed extends beyond the end of the Waveform and the implementation does not reallocate a larger capacity, the method shall throw an Argument Out Of Range exception.

5. IWaveform<T> Interface

5.1 Overview

IVI provides standard definitions for Waveforms and Spectrums. Drivers that produce or consume a waveform/spectrum use these common types to simplify working with those objects.

IWaveform<T> is the most basic template interface for a waveform. The requirements of the common properties and methods are described in section 4 (Common Properties and Methods of Waveform and Spectrum Interfaces).

IWaveform<T> does not provide direct access to the data array so it can be used where the data array is not in conventional memory. IWaveform is also the basis for other waveform types.

5.2 IWaveform<T> Properties

The IWaveform<T> interface defines the following properties:

- Item
- Capacity
- ContainsInvalidElement
- ContainsOutOfRangeElement
- EndTime
- FirstValidPoint
- IntervalPerPoint
- Offset
- Scale
- StartTime
- TotalTime
- TriggerTime
- ValidPointCount

Each of these is described in section 4, *Common Properties and Methods of Waveform and Spectrum Interfaces*.

5.3 IWaveform <T> Methods

The IWaveform <T> interface defines the following methods:

- Configure
- GetAllElements
- GetElements
- GetScaled
- PutElements

Each of these is described in section 4 (Common Properties and Methods of Waveform and Spectrum Interfaces).

6. IMemoryWaveform<T> Interface

6.1 Overview

The IMemoryWaveform interface extends from the IWaveform interface and includes all of the methods and properties of IWaveform and has the same requirements and capabilities. In addition, the IMemoryWaveform provides direct access to the explicit data with an in-memory array.

6.1.1 Type Parameters

IVI specifications, classes or specific drivers that produce or consume objects that implement or extend IMemoryWaveform<T> shall document the type parameters they support.

6.1.2 Base Interface

The IMemoryWaveform<T> interface extends the IWaveform<T> interface.

6.2 IMemoryWaveform<T> Properties

The IMemoryWaveform<T> interface defines the following properties:

- Data

This section describes the behavior and requirements of the Data property.

6.2.1 Data

Data Type	Access
`T[]`	RW

.NET Prototype

```
T[] Data { get; set; }
```

Description

A public, in-memory array containing elements of type T than contains the explicit waveform data. Clients can use the Data property to directly access the in-memory data without copying.

The Data property returns the entire original array, not a copy of the array. Changes made to the array returned by the Data property are made to the original array. If the template data type is an integer, the data will not be scaled.

To acquire a copy of all or part of an array, use the GetAllElements or GetElements methods.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

7. ISpectrum<T> Interface

A spectrum is used to represent a frequency-varying value. Therefore, the data array in a spectrum may refer to power, voltage, or other sundry values.

7.1 Overview

IVI provides standard definitions for Spectrums. Drivers that produce or consume spectra use these common types to simplify working with those objects.

ISpectrum<T> is the most basic template interface for a spectrum. The requirements of the common properties and methods are described in section 4 (Common Properties and Methods of Waveform and Spectrum Interfaces).

ISpectrum<T> does not provide direct access to the data array so it can be used where the data array is not in conventional memory. ISpectrum is also the basis for other spectrum types.

7.2 ISpectrum<T> Properties

The ISpectrum<T> interface defines the following properties:

- Item
- Capacity
- ContainsInvalidElement
- ContainsOutOfRangeElement
- FirstValidPoint
- FrequencyStep
- StartFrequency
- StopFrequency
- Offset
- Scale
- TriggerTime
- ValidPointCount

Each of these is described in section 4, *Common Properties and Methods of Waveform and Spectrum Interfaces*.

7.3 ISpectrum<T> Methods

The ISpectrum<T> interface defines the following methods:

- Configure
- GetAllElements
- GetElements
- GetScaled
- PutElements

Each of these is described in section 4 (Common Properties and Methods of Waveform and Spectrum Interfaces

8. IMemorySpectrum<T> Interface

8.1 Overview

The IMemorySpectrum<T> interface extends the ISpectrum<T> interface and includes all of the methods and properties of ISpectrum<T> and has the same requirements and capabilities. In addition, the IMemorySpectrum<T> provides direct access to the explicit data with an in-memory array.

8.1.1 Base Interface

The IMemorySpectrum<T> interface extends the ISpectrum<T> interface.

8.2 IMemorySpectrum Properties

The IMemorySpectrum<T> interface defines the following properties:

- Data

This section describes the behavior and requirements of the Data property.

8.2.1 Data

Data Type	Access
`Double[]`	RW

.NET Prototype

```
Double[] Data { get; set; }
```

Description

A public, in-memory array containing elements of type T that contains the explicit spectrum data. Clients can use the Data property to directly access the in-memory data without copying.

The Data property returns the entire original array, not a copy of the array. Changes made to the array returned by the Data property are made to the original array. If the template data type is an integer, the data will not be scaled.

To acquire a copy of all or part of an array, use the GetAllElements or GetElements methods.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

9. Waveform<T> Class

The Waveform<T> class is a concrete class provided by IVI to contain waveforms. The IviDriver.Waveform class implements the IMemoryWaveform<T> interface.

9.1 Overview

9.1.1 Type Parameter Types

The IviDriver.Waveform class supports struct template types of:

- Byte
- Int16
- Int32
- Int64
- Single
- Double

9.1.2 Implemented Interfaces

The Waveform<T> class implements the IMemoryWaveform<T> interface.

9.1.3 Implemention Limitations

The Waveform class is subject to the .NET 2GB limitation, and cannot be used to represent a waveform exceeding 2GB. It is possible to manage waveforms larger than this by creating a class that implements the IWaveform<T> interface using multiple memory allocations to present a single waveform greater than 2GBs in size. Attempts to set the capacity to a value that would exceed this limit will cause the method to throw a System.InsufficientMemory exception.

9.2 Waveform Constructors

Description

Waveform is concrete class that implements the IMemoryWaveform and IWaveform interfaces. It has no specific semantics other than containing the data.

Note the waveform allocates memory if the Capacity is specified or if it is initialized with another waveform.

.NET Prototype

```
Waveform(IWaveform waveform);

Waveform(PrecisionTimeSpan startTime,
         PrecisionTimeSpan intervalPerPoint,
         PrecisionDateTime triggerTime);

Waveform(PrecisionTimeSpan startTime,
         PrecisionTimeSpan intervalPerPoint);

Waveform(PrecisionTimeSpan intervalPerPoint);

Waveform(PrecisionTimeSpan startTime,
         PrecisionTimeSpan intervalPerPoint,
         PrecisionDateTime triggerTime,
         Int64 capacity);

Waveform(PrecisionTimeSpan startTime,
         PrecisionTimeSpan intervalPerPoint,
         Int64 capacity);

Waveform(PrecisionTimeSpan intervalPerPoint,
         Int64 capacity);
```

Parameters

Input	Description	Data Type
`startTime`	Set the Waveforms startTime property per the IWaveform definition. The default value is PrecisionTimeSpan.Zero.	`PrecisionTimeSpan`
`intervalPerPoint`	Set the Waveforms intervalPerPoint property per the IWaveform definition. There is no default value, and it cannot be zero.	`PrecisionTimeSpan`
`triggerTime`	Sets the Waveforms triggerTime property per the IWaveform definition The default value is Not a Time.	`PrecisionDateTime`
`capacity`	The capacity of the waveform data array. The default is 0. If this value is greater than 0, the memory is allocated in the constructor.	`Int64`

`waveform`	A waveform whose implicit axis information will be copied into this waveform.	`IWaveform`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by these constructors.

10. Spectrum<T> Class

10.1 Overview

The Spectrum<T> class is a concrete class provided by IVI to contain spectrums. The Spectrum<T> class implements the IMemorySpectrum<T> interface.

10.1.1 Type Parameter Types

The IviDriver.Spectrum class supports struct template types of:

- Byte
- Int16
- Int32
- Int64
- Single
- Double

10.1.2 Implemented Interfaces

The Spectrum class implements the IMemorySpectrum<T> interface.

10.1.3 Implemention Limitations

The Spectrum class is subject to the .NET 2GB limitation, and cannot be used to represent a spectrum exceeding 2GB. It is possible to manage spectra larger than this by creating a class that implements the ISpectrum<T> interface using multiple memory allocations to present a single spectrum greater than 2GBs in size.

10.2 Spectrum Constructors

Description

Spectrum is concrete class that implements the ISpectrumWaveform and ISpectrum interfaces. It has no specific semantics other than containing the data.

.NET Prototype

```
Spectrum(ISpectrum spectrum);

Spectrum(Double startFrequency,
         Double stopFrequency,
         PrecisionDateTime triggerTime,
         Int64 capacity);

Spectrum(Double startFrequency,
         Double stopFrequency,
         Int64 capacity);

Spectrum(Double startFrequency,
         Double stopFrequency,
         PrecisionDateTime triggerTime);

Spectrum(Double startFrequency,
         Double stopFrequency);
```

Parameters

Input	Description	Data Type
`startFrequency`	The frequency in Hertz corresponding to the first point in the data array.	`Double`
`stopFrequency`	The frequency in Hertz corresponding to the final point in the data array.	`Double`
`triggerTime`	Sets the Spectrum triggerTime property per the ISpectrum definition The default value is Not a Time.	`PrecisionDateTime`
`capacity`	The capacity of the spectrum data array. The default is 0. If this value is greater than 0, the memory is allocated in the constructor.	`Int64`
`spectrum`	A spectrum whose implicit axis information will be copied into this spectrum.	`ISpectrum`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by these constructors.

11. Repeated Capability Collection Base Interfaces

11.1 Overview

IVI.NET provides two base interfaces that are used in the implementation of IVI.NET repeated capability collections. For more information on the use of these interfaces, refer to section 4, *Repeated Capability Group*, of *IVI-3.3: Standard Cross-Class Capabilities Specification*, and section 12, *Repeated Capabilities*, of *IVI-3.4: API Style Guide*.

All IVI.NET repeated capability collection interfaces extend IIviRepeatedCapabilityCollection, either directly or indirectly.

All IVI.NET interfaces that represent instances of a repeated capability collection extend IIviRepeatedCapabilityIdentification, either directly or indirectly.

11.2 IIviRepeatedCapabilityCollection<T>

<T> is the interface that represents instances of a repeated capability collection. If there are multiple interfaces that represent a collection instance, <T> is the one that is returned by the collection.

The IIviRepeatedCapabilityCollection interface extends the following interface:

- IEnumerable<T>

The IIviRepeatedCapabilityCollection interface defines the following properties:

- Count
- Item Indexer

11.2.1 Count

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	<RC>s	None	

.NET Property Name

`Count`

Description

Specifies the number of repeated capabilities in the collection.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

11.2.2 Item Indexer

Data Type	Access	Applies to	Coercion	High Level Functions
T	RO	IIviRepeatedCapabilityCollection	None	

.NET Property Name

```
T this[String name];
```

Description

Item uniquely identifies an instance of a repeated capability in the repeated capability collection. It returns an interface reference which can be used to control the attributes and other functionality of that repeated capability.

The .NET indexer takes a repeated capability name If the user passes an invalid value for the name parameter, the indexer returns an error.

Valid names include physical repeated capability identifiers and virtual repeated capability identifiers.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this indexer.

11.3 IIviRepeatedCapabilityIdentification

The IIviRepeatedCapabilityIdentification interface defines the following properties:

- Name

11.3.1 Name

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	R	<RC>	None	

.NET Property Name

`Name`

Description

Returns the physical repeated capability identifier defined by the specific driver for the repeated capability that corresponds to the index that the user specifies. If the driver defines a qualified repeated capability name, this property returns the qualified name.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

Compliance Notes

For an instrument with only one repeated capability, i.e. the Count attribute is one, the driver may return an empty string.

12. LockManager Class

The LockManager is a concrete class provided by the IVI Foundation to aid driver developers in implementing multithread locking within an IVI.NET driver. IVI.NET drivers are not required to use the LockManager, but it does provide all three levels of multithread locking as described in Section 4.3.11, *Multithread Safety*, of *IVI-3.1: Driver Architecture Specification*.

The LockManager exposes one public constructor and two public methods.

12.1 LockManager Constructor

The LockManager.class exposes the following public constructor:

```
LockManager(IIviDriver driver,
            LockType requestedLockType,
            String accessKey)
```

An instance of an IVI.NET driver instantiates a single instance of the LockManager and uses that LockManager instance to: a) obtain a multithread lock within a single driver method, and b) to implement the two overloads of IIviDriverUtility.Lock.The first parameter of the constructor represents the instance of the driver that will use the LockManager. The second two parameters match the lockType and accessKey parameters of the IVI.NET driver constructor. Refer to Section 8, *IVI.NET Constuctors*, of *IVI-3.2: Inherent Capabilities Specification*, for an explanation of these IVI.NET driver constructor parameters.

If requested lock type is LockType.Process, and the access key is the empty string, then the lock requested is an instance-wide lock.

If requested lock type is LockType.Process, and the access key is not the empty string, then the lock requested is a process-wide lock.

If requested lock type is LockType.Machine, the access key may not be the empty string, and the lock requested is a machine-wide lock.

12.2 LockManager Lock method

The LockManager class exposes the following two public methods:

```
IIviDriverLock Lock()
IIviDriverLock Lock(PrecisionTimeSpan maximumTime)
```

These methods are used to implement the IIviDriverUtility.Lock methods. The IVI.NET driver implementation delegates to these two methods on the LockManager class. Refer to Section 6.18, *Lock Session*, of IVI-*3.2: Inherent Capabilities Specification*, for details on the IIviDriverUtility.Lock methods.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

12.3 Example Usage

The following C# code demonstrates the intended usage of the LockManager class within a fictious driver named "Acme4301".

```
public class Acme4301
{
  private LockManager _lockManager;

  public Acme4301(String resourceName, Boolean idQuery, Boolean reset,
                  LockType lockType, String accessKey, String options)
  {
    // Cache a single instance of the LockManager for the entire driver
    _lockManager = new LockManager(this, lockType, accessKey);
  }

  public IIviDriverLock Lock()
  {
    // Use LockManager to implement external locking across method calls
    return _lockManager.Lock();
  }

  public void Configure(Double Range, Double Resolution)
  {
    // Use LockManager to implement internal locking within a method call
    using (this.Lock())
    {
      // ... send device command
    }
  }
}
```

13. Enumerations

13.1 Auto

The Auto enumeration provides the value need for automatic setting properties that implement a "Once" option. Automatic setting properties that only use "On" and "Off" should be implemented as Booleans.

Name	***Description***			
		Language	***Identifier***	***Value***
Auto Off	Disables auto-<Name>. The instrument sets the <Name> attribute to the value it most recently calculated. Further queries of the <Name> attribute return the actual value.			
		.NET	`Auto.Off`	0
Auto On	Sets the instrument to calculate the auto setting's primary attribute automatically. When On, the actual value for auto setting's primary attribute as automatically determined by the instrument is returned by the auto setting's primary attribute attribute Setting the auto setting's primary attribute attribute also sets the automatic setting attribute to Auto Off.			
		.NET	`Auto.On`	1
Auto Once	Sets the instrument to calculate <Name> exactly once, before its next use. After its next use, the instrument uses the calculated value of <Name> for subsequent uses. Further queries of the <Name> attribute return the actual value.			
		.NET	`Auto.Once`	2

13.2 Slope

The Slope enumeration provides the standard "Positive" and "Negative" values for slope.

Name	***Description***			
		Language	***Identifier***	***Value***
Negative	Sets slope to negative.			
		.NET	`Slope.Negative`	0
Positive	Sets slope to positive.			
		.NET	`Slope.Positive`	1

14. Standard TriggerSource Class

The `TriggerSource` class provides drivers and clients with a standard way to use the standard trigger source values. The standard trigger source values are defined in *IVI-3.3, Standard Cross Class Capabilities Specification*, Section 3, *Standard Trigger Source Values*.

The `TriggerSource` class is a static class with static properties. Each property is named for a trigger source and returns the standard string value for that source. Property names and returned values are shown in the table below. All properties are read-only.

Property Name	***String Return Value***
None	`String.Empty`
Immediate	`"Immediate"`
External	`"External"`
Internal	`"Internal"`
Software	`"Software"`
Get	`"GET"`
ACLine	`"ACLine"`
Interval	`"Interval"`
Lan0	`"LAN0"`
Lan1	`"LAN1"`
Lan2	`"LAN2"`
Lan3	`"LAN3"`
Lan4	`"LAN4"`
Lan5	`"LAN5"`
Lan6	`"LAN6"`
Lan7	`"LAN7"`
Lxi0	`"LXI0"`
Lxi1	`"LXI1"`
Lxi2	`"LXI2"`
Lxi3	`"LXI3"`
Lxi4	`"LXI4"`
Lxi5	`"LXI5"`
Lxi6	`"LXI6"`
Lxi7	`"LXI7"`
Ttl0	`"TTL0"`
Ttl1	`"TTL1"`
Ttl2	`"TTL2"`
Ttl3	`"TTL3"`
Ttl4	`"TTL4"`
Ttl5	`"TTL5"`

Ttl6	“TTL6”
Ttl7	“TTL7”
Ecl0	“ECL0”
Ecl1	“ECL1”
PxiClk10	“PXI_CLK10”
PxiStar	“PXI_STAR”
PxiTrigger0	“PXI_TRIG0”
PxiTrigger1	“PXI_TRIG1”
PxiTrigger2	“PXI_TRIG2”
PxiTrigger3	“PXI_TRIG3”
PxiTrigger4	“PXI_TRIG4”
PxiTrigger5	“PXI_TRIG5”
PxiTrigger6	“PXI_TRIG6”
PxiTrigger7	“PXI_TRIG7”
PxiExpressClk100	“PXIe_CLK100
PxiExpressDStarA	“PXIe_DSTARA”
PxiExpressDStarB	“PXIe_DSTARB”
PxiExpressDStarC	“PXIe_DSTARC”
Rtsi0	“RTSI0”
Rtsi1	“RTSI1”
Rtsi2	“RTSI2”
Rtsi3	“RTSI3”
Rtsi4	“RTSI4”
Rtsi5	“RTSI5”
Rtsi6	“RTSI6”
Rtsi7	“RTSI7”

The `TriggerSource` class also overloads the Equals method. Note that the Equals method will validate either String.Empty or the string “None” as a standard value.

Method Name
Equals

15. IVI.NET Utility Classes and Interfaces Exceptions

This section defines the list of IVI.NET exceptions specific to the IVI.NET utility classes and interfaces. For general information on IVI.NET exceptions and warnings, refer to *IVI-3.1: Driver Architecture Specification* and section 12, *Common IVI.NET Exceptions and Warnings*, of *IVI-3.2: Inherent Capabilities Specification*.

15.1 IVI.NET Exceptions

The following exceptions defined in the .NET Framework are used by properties and methods defined in this specfciation.

- DataArrayTooSmallException
- InvalidSpectrumDataTypeException
- InvalidWaveformDataTypeException
- NotATimeException
- ValidPointCountExceedsCapacityException

Note that other exceptions defined by the .NET Framework may be thrown by IVI.NET drivers, but only if there is not an IVI defined exception that can be used instead.

15.1.1 ValidPointCountExceedsCapacityException

Description

The specified valid point count exceeds the capacity of the waveform or spectrum.

Exception

```
Ivi.Driver.ValidPointCountExceedsCapacityException
```

Constructors

```
Ivi.Driver.ValidPointCountExceedsCapacityException(
                                    String validPointCount,
                                    String capacity);

Ivi.Driver.ValidPointCountExceedsCapacityException();

Ivi.Driver.ValidPointCountExceedsCapacityException(
                                    String message);

Ivi.Driver.ValidPointCountExceedsCapacityException(
                                    String message,
                                    System.Exception innerException);
```

Parameters

Inputs	Description	Base Type
`validPointCount`	The valid point count specified in the waveform or spectrum configure method.	`String`
`capacity`	The capacity of the waveform or spectrum.	`String`

Default Message String

```
The specified valid point count exceeds the capacity of the waveform or
spectrum object's data array.
Valid point count: <validPointCount>
Destination object's capacity: <capacity>
```

Usage

Avoid using this exception to relay another exception. As a general rule, just let the original exception propagate up.

If driver developers specify the message string, they are responsible for message string localization.

15.1.2 DataArrayTooSmallException

Description

The measured waveform or spectrum exceeds the capacity of the waveform or spectrum.

Exception

```
Ivi.Driver.DataArrayTooSmallException
```

Constructors

```
Ivi.Driver.DataArrayTooSmallException(String measuredElements,
                                      String capacity);

Ivi.Driver.DataArrayTooSmallException();

Ivi.Driver.DataArrayTooSmallException(String message);

Ivi.Driver.DataArrayTooSmallException(String message,
                                      System.Exception innerException);
```

Default Message String

```
The mesured waveform or spectrum exceeds the capacity of the waveform or
spectrum object's data array.
Measured elements: <measuredElements>
Destination object's capacity: <capacity>
```

Parameters

Inputs	Description	Base Type
`measuredElements`	The measured number of elements.	`String`
`capacity`	The capacity of the waveform or spectrum.	`String`

Usage

Avoid using this exception to relay another exception. As a general rule, just let the original exception propagate up.

If driver developers specify the message string, they are responsible for message string localization.

15.1.3 InvalidSpectrumDataTypeException

Description

The spectrum class does support data arrays of the specified type.

Exception

```
Ivi.Driver.InvalidSpectrumDataTypeException
```

Constructors

```
Ivi.Driver.InvalidSpectrumDataTypeException(String message
                                            String type);

Ivi.Driver.InvalidSpectrumDataTypeException();

Ivi.Driver.InvalidSpectrumDataTypeException(String message);

Ivi.Driver.InvalidSpectrumDataTypeException(String message,
                                            System.Exception
innerException);
```

Default Message String

```
The spectrum class does not support data arrays of the specified type.
Type: <type>
```

Parameters

Inputs	Description	Base Type
`type`	The specified waveform or spectrum type.	`String`

Usage

Avoid using this exception to relay another exception. As a general rule, just let the original exception propagate up.

15.1.4 InvalidWaveformDataTypeException

Description

The waveform class does support data arrays of the specified type.

Exception

```
Ivi.Driver.InvalidWaveformDataTypeException
```

Constructors

```
Ivi.Driver.InvalidSpectrumDataTypeException(String message
                                            String type);

Ivi.Driver.InvalidWaveformDataTypeException();

Ivi.Driver.InvalidWaveformDataTypeException(String message);

Ivi.Driver.InvalidWaveformDataTypeException(String message,
                                            System.Exception
innerException);
```

Default Message String

```
The waveform class does not support data arrays of the specified type.
Type: <type>
```

Parameters

Inputs	Description	Base Type
`type`	The specified waveform or spectrum type.	`String`

Usage

Avoid using this exception to relay another exception. As a general rule, just let the original exception propagate up.

15.1.5 NotATimeException

Description

The PrecisionDateTime value is Not a Time (NaT).

Exception

```
Ivi.Driver.NotATimeException
```

Constructors

```
Ivi.Driver.NotATimeException(String message,
                             String paramName);

Ivi.Driver.NotATimeException();

Ivi.Driver.NotATimeException(String message);

Ivi.Driver.NotATimeException(String message,
                             System.Exception innerException);
```

Default Message String

```
The PrecisionDateTime value is Not a Time (NaT).
Parameter name: <paramName>
```

Parameters

Inputs	Description	Base Type
`paramName`	The name of the PrecisionDateTime parameter whose value is Not a Time (NaT).	`String`

Usage

This exception will not generally be used to relay another exception.

This exception is thrown extensively by IVI.NET PrecisionDateTime class distributed by the IVI Foundation as part of the IVI.NET Shared Components. It may also be thrown by drivers.