

Integration Objects'

**Seamless & Secure IT-OT-IIoT Integration
Platform**

Smart IoT Highway

Version 2.4.3 Rev.1

**DATA MODEL
EXPRESSIONS REFERENCE**

Integration Objects' Smart IoT Highway Data Model Expressions Reference Version 2.4.3 Rev 1

Published April 2026.

Copyright © 2018 - 2026 Integration Objects. All rights reserved.

No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Integration Objects.

Windows®, Windows NT® and .NET are registered trademarks of Microsoft Corporation.

SIOTH® is a registered trademark of Integration Objects.

TABLE OF CONTENTS

| | |
|---|-----------|
| PREFACE | 13 |
| INTRODUCTION | 15 |
| GETTING STARTED | 17 |
| DATA MODEL EXPRESSIONS..... | 22 |
| 1. MATHEMATICAL OPERATIONS..... | 22 |
| 1.1. Math Properties | 22 |
| 1.2. Math Methods..... | 22 |
| 1.2.1. Abs..... | 22 |
| 1.2.2. Acos, Acosh, Cos, Cosh | 23 |
| 1.2.3. Asin, Asinh, Sin, Sinh..... | 24 |
| 1.2.4. Atan, Atanh, Atan2, Tan, Tanh..... | 24 |
| 1.2.5. BigMul | 25 |
| 1.2.6. BitDecrement..... | 25 |
| 1.2.7. BitIncrement..... | 25 |
| 1.2.8. Cbrt..... | 26 |
| 1.2.9. Ceiling..... | 26 |
| 1.2.10. Clamp..... | 26 |
| 1.2.11. CopySign | 28 |
| 1.2.12. Exp | 28 |
| 1.2.13. FusedMultiplyAdd..... | 28 |
| 1.2.14. IEEEERemainder..... | 28 |
| 1.2.15. Log..... | 29 |
| 1.2.16. Max..... | 29 |
| 1.2.17. MaxMagnitude..... | 30 |

| | | |
|---------|------------------------------|----|
| 1.2.18. | Min..... | 30 |
| 1.2.19. | MinMagnitude..... | 32 |
| 1.2.20. | Pow..... | 32 |
| 1.2.21. | ReciprocalEstimate | 32 |
| 1.2.22. | Round | 32 |
| 1.2.23. | ScaleB..... | 35 |
| 1.2.24. | Sign..... | 35 |
| 1.2.25. | Sqrt | 36 |
| 1.2.26. | Truncate..... | 36 |
| 1.3. | Math Examples..... | 37 |
| 2. | CONVERT OPERATIONS | 37 |
| 2.1. | Convert Methods..... | 38 |
| 2.1.1. | ToDouble..... | 38 |
| 2.1.2. | ToByte..... | 40 |
| 2.1.3. | ToSByte | 42 |
| 2.1.4. | ToInt16 | 43 |
| 2.1.5. | ToInt32 | 45 |
| 2.1.6. | ToInt64 | 47 |
| 2.1.7. | ToUInt16..... | 49 |
| 2.1.8. | ToUInt32..... | 51 |
| 2.1.9. | ToUInt64..... | 54 |
| 2.1.10. | ToSingle | 56 |
| 2.1.11. | ToChar | 58 |
| 2.1.12. | ToString | 59 |
| 2.1.13. | ToDateTime | 61 |
| 2.1.14. | ToBoolean..... | 62 |
| 2.2. | Convert Examples..... | 64 |
| 3. | BITCONVERTER OPERATIONS..... | 64 |

| | | |
|--------|--|----|
| 3.1. | BitConverter Methods | 64 |
| 3.2. | BitConverter Examples | 65 |
| 4. | BITOPERATIONS METHODS | 66 |
| 4.1. | BitOperations Methods..... | 66 |
| 4.1.1. | IsPow..... | 66 |
| 4.1.2. | LeadingZeroCount..... | 66 |
| 4.1.3. | Log2..... | 67 |
| 4.1.4. | PopCount | 67 |
| 4.1.5. | Rotate | 68 |
| 4.1.6. | RoundUpToPowerOf2 | 68 |
| 4.1.7. | TrailingZeroCount..... | 69 |
| 4.2. | BitOperations Examples..... | 69 |
| 5. | BOOLEAN PROPERTIES, METHODS AND OPERATORS..... | 69 |
| 5.1. | Boolean Properties | 70 |
| 5.2. | Boolean Methods..... | 70 |
| 5.3. | Boolean Operators..... | 71 |
| 5.4. | Boolean Examples | 72 |
| 6. | CHAR PROPERTIES, METHODS AND OPERATORS..... | 72 |
| 6.1. | Char Properties..... | 72 |
| 6.2. | Char Methods..... | 73 |
| 6.3. | Char Operators..... | 79 |
| 6.4. | Char Examples | 82 |
| 7. | BYTE PROPERTIES, METHODS AND OPERATORS | 82 |
| 7.1. | Byte Properties | 82 |
| 7.2. | Byte Methods..... | 83 |
| 7.3. | Byte Operators | 83 |
| 7.4. | Byte Examples..... | 87 |
| 8. | SBYTE PROPERTIES, METHODS AND OPERATORS..... | 87 |

| | | |
|-------|--|-----|
| 8.1. | SByte Properties..... | 87 |
| 8.2. | SByte Methods | 88 |
| 8.3. | SByte Operators..... | 89 |
| 8.4. | SByte Examples | 89 |
| 9. | INT16 PROPERTIES, METHODS AND OPERATORS | 89 |
| 9.1. | Int16 Properties..... | 90 |
| 9.2. | Int16 Methods | 90 |
| 9.3. | Int16 Operators..... | 91 |
| 9.4. | Int16 Examples | 94 |
| 10. | INT32 PROPERTIES, METHODS AND OPERATORS | 94 |
| 10.1. | Int32 Properties..... | 94 |
| 10.2. | Int32 Methods | 95 |
| 10.3. | Int32 Operators..... | 96 |
| 10.4. | Int32 Examples | 96 |
| 11. | INT64 PROPERTIES, METHODS AND OPERATORS | 96 |
| 11.1. | Int64 Properties..... | 97 |
| 11.2. | Int64 Methods | 97 |
| 11.3. | Int64 Operators..... | 98 |
| 11.4. | Int64 Examples | 98 |
| 12. | UINT16 PROPERTIES, METHODS AND OPERATORS | 99 |
| 12.1. | UInt16 Properties | 99 |
| 12.2. | UInt16 Methods | 99 |
| 12.3. | UInt16 Operators..... | 100 |
| 12.4. | UInt16 Examples | 101 |
| 13. | UINT32 PROPERTIES, METHODS AND OPERATORS | 101 |
| 13.1. | UInt32 Properties | 101 |
| 13.2. | UInt32 Methods | 102 |
| 13.3. | UInt32 Operators..... | 103 |

| | |
|--|-----|
| 13.4. UInt32 Examples | 103 |
| 14. UINT64 PROPERTIES, METHODS AND OPERATORS | 104 |
| 14.1. UInt64 Properties | 104 |
| 14.2. UInt64 Methods | 104 |
| 14.3. UInt64 Operators | 105 |
| 14.4. UInt64 Examples | 106 |
| 15. FLOAT PROPERTIES, METHODS AND OPERATORS | 106 |
| 15.1. Float Properties | 106 |
| 15.2. Float Methods | 107 |
| 15.3. Float Operators | 109 |
| 15.4. Float Examples | 111 |
| 16. DOUBLE PROPERTIES, METHODS AND OPERATORS | 111 |
| 16.1. Double Properties | 111 |
| 16.2. Double Methods | 112 |
| 16.3. Double Operators | 114 |
| 16.4. Double Examples | 114 |
| 17. STRING PROPERTIES, METHODS AND OPERATORS | 115 |
| 17.1. String Properties | 115 |
| 17.2. String Methods | 115 |
| 17.3. String Operators | 125 |
| 17.4. String Examples | 126 |
| 18. DATETIME PROPERTIES, METHODS AND OPERATORS | 126 |
| 18.1. DateTime Properties | 126 |
| 18.2. DateTime Methods | 129 |
| 18.3. DateTime Operators | 133 |
| 18.4. DateTime Examples | 134 |
| 19. CUSTOM METHODS | 134 |
| 19.1. Custom Methods Examples | 135 |

TABLE OF FIGURES

| | |
|--|-----|
| FIGURE 1: SIOTH® PLATFORM OVERVIEW..... | 15 |
| FIGURE 2: CREATE NEW INSTANCE BUTTONS | 17 |
| FIGURE 3: EDIT INSTANCE BUTTON | 18 |
| FIGURE 4: EDIT ATTRIBUTE ICON..... | 18 |
| FIGURE 5: CONFIGURING AN ATTRIBUTE WITH AN EXPRESSION..... | 19 |
| FIGURE 6: EXPRESSION WITH REFERENCE TO OTHER ATTRIBUTES..... | 20 |
| FIGURE 7: INSTANCE CURRENT VALUES VIEW..... | 21 |
| FIGURE 8: MATH PROPERTIES EXAMPLE..... | 37 |
| FIGURE 9: MATH METHODS EXAMPLE | 37 |
| FIGURE 10: CONVERT METHODS EXAMPLE..... | 64 |
| FIGURE 11: BITCONVERTER METHODS EXAMPLE | 65 |
| FIGURE 12: BITOPERATIONS METHODS EXAMPLE..... | 69 |
| FIGURE 13: BOOLEAN OPERATIONS EXAMPLE | 72 |
| FIGURE 14: CHAR OPERATIONS EXAMPLE | 82 |
| FIGURE 15: BYTE OPERATIONS EXAMPLE..... | 87 |
| FIGURE 16: SBYTE OPERATIONS EXAMPLE | 89 |
| FIGURE 17: INT16 OPERATIONS EXAMPLE | 94 |
| FIGURE 18: INT32 OPERATIONS EXAMPLE | 96 |
| FIGURE 19: INT64 OPERATIONS EXAMPLE | 98 |
| FIGURE 20: UINT16 OPERATIONS EXAMPLE | 101 |
| FIGURE 21: UINT32 OPERATIONS EXAMPLE | 103 |
| FIGURE 22: UINT64 OPERATIONS EXAMPLE | 106 |
| FIGURE 23: FLOAT OPERATIONS EXAMPLE | 111 |

| | |
|--|-----|
| FIGURE 24: DOUBLE OPERATIONS EXAMPLE..... | 114 |
| FIGURE 25: STRING OPERATIONS EXAMPLE | 126 |
| FIGURE 26: DATETIME OPERATIONS EXAMPLE | 134 |
| FIGURE 27: CUSTOM METHODS EXAMPLE | 135 |

LIST OF TABLES

| | |
|--|----|
| TABLE 1: MATH.PROPERTIES..... | 22 |
| TABLE 2: MATH.ABS METHOD | 23 |
| TABLE 3: MATH.ACOS, MATH.ACOSH, MATH.COS, MATH.COSH METHODS..... | 23 |
| TABLE 4: MATH.ASIN, MATH.ASINH, MATH.SIN, MATH.SINH METHODS | 24 |
| TABLE 5: MATH.ATAN, MATH.ATANH, MATH.ATAN2, MATH.TAN, MATH.TANH METHODS..... | 24 |
| TABLE 6: MATH.BIGMUL METHOD..... | 25 |
| TABLE 7: MATH.BITDECREMENT METHOD | 25 |
| TABLE 8: MATH.BITINCREMENT METHOD | 25 |
| TABLE 9: MATH.CBRT METHOD | 26 |
| TABLE 10: MATH.CEILING METHOD | 26 |
| TABLE 11: MATH.CLAMP METHOD | 27 |
| TABLE 12: MATH.COPYSIGN METHOD | 28 |
| TABLE 13: MATH.EXP METHOD | 28 |
| TABLE 14: MATH.FUSEDMULTIPLYADD METHOD..... | 28 |
| TABLE 15: MATH.IEEEREMAINDER METHOD..... | 28 |
| TABLE 16: MATH.LOG METHODS | 29 |
| TABLE 17: MATH.MAX METHOD | 30 |
| TABLE 18: MATH.MAXMAGNITUDE METHOD..... | 30 |
| TABLE 19: MATH.MIN METHOD..... | 31 |
| TABLE 20: MATH.MINMAGNITUDE METHOD..... | 32 |
| TABLE 21: MATH.POW METHOD..... | 32 |
| TABLE 22: MATH.RECIPROCALESTIMATE METHOD | 32 |
| TABLE 23: MATH.ROUND METHODS | 35 |

| | |
|--|----|
| TABLE 24: MATH.SCALEB METHOD | 35 |
| TABLE 25: MATH.SIGN METHOD | 36 |
| TABLE 26: MATH.SQRT METHOD | 36 |
| TABLE 27: MATH.TRUNCATE METHOD | 36 |
| TABLE 28: CONVERT.TODOUBLE METHOD | 39 |
| TABLE 29: CONVERT.TOBYTE METHOD | 41 |
| TABLE 30: CONVERT.TOSBYTE METHOD | 43 |
| TABLE 31: CONVERT.TOINT16 METHOD | 45 |
| TABLE 32: CONVERT.TOINT32 METHOD | 47 |
| TABLE 33: CONVERT.TOINT64 METHOD | 49 |
| TABLE 34: CONVERT.TOUINT16 METHOD | 51 |
| TABLE 35: CONVERT.TOUINT32 METHOD | 53 |
| TABLE 36: CONVERT.TOUINT64 METHOD | 55 |
| TABLE 37: CONVERT.TOFLOAT METHOD | 57 |
| TABLE 38: CONVERT.TOCHAR METHOD | 59 |
| TABLE 39: CONVERT.TOSTRING METHOD | 61 |
| TABLE 40: CONVERT.TODATETIME METHOD | 62 |
| TABLE 41: CONVERT.TOBOOLEAN METHOD | 63 |
| TABLE 42: BITCONVERTER METHODS | 65 |
| TABLE 43: BITOPERATIONS.ISPOW METHOD | 66 |
| TABLE 44: BITOPERATIONS.LEADINGZEROCOUNT METHOD | 67 |
| TABLE 45: BITOPERATIONS.LOG2 METHOD | 67 |
| TABLE 46: BITOPERATIONS.POPCOUNT METHOD | 67 |
| TABLE 47: BITOPERATIONS.ROTATE METHODS | 68 |
| TABLE 48: BITOPERATIONS.ROUNDUPTOPOWEROF2 METHOD | 68 |
| TABLE 49: BITOPERATIONS.TRAILINGZEROCOUNT METHOD | 69 |
| TABLE 50: BOOLEAN PROPERTIES | 70 |
| TABLE 51: BOOLEAN METHODS | 71 |

| | |
|-----------------------------------|-----|
| TABLE 52: BOOLEAN OPERATORS..... | 72 |
| TABLE 53: CHAR PROPERTIES..... | 73 |
| TABLE 54: CHAR METHODS..... | 79 |
| TABLE 55: CHAR OPERATORS..... | 81 |
| TABLE 56: BYTE PROPERTIES..... | 82 |
| TABLE 57: BYTE METHODS | 83 |
| TABLE 58: BYTE OPERATORS | 86 |
| TABLE 59: SBYTE PROPERTIES | 87 |
| TABLE 60: SBYTE METHODS..... | 88 |
| TABLE 61: INT16 PROPERTIES | 90 |
| TABLE 62: INT16 METHODS..... | 91 |
| TABLE 63: INT16 OPERATORS..... | 93 |
| TABLE 64: INT32 PROPERTIES | 95 |
| TABLE 65: INT32 METHODS..... | 95 |
| TABLE 66: INT64 PROPERTIES | 97 |
| TABLE 67: INT64 METHODS..... | 98 |
| TABLE 68: UInt16 PROPERTIES | 99 |
| TABLE 69: UInt16 METHODS..... | 100 |
| TABLE 70: UInt32 PROPERTIES | 102 |
| TABLE 71: UInt32 METHODS..... | 103 |
| TABLE 72: UInt64 PROPERTIES | 104 |
| TABLE 73: UInt64 METHODS..... | 105 |
| TABLE 74: FLOAT PROPERTIES | 107 |
| TABLE 75: FLOAT METHODS..... | 109 |
| TABLE 76: FLOAT OPERATORS..... | 110 |
| TABLE 77: DOUBLE PROPERTIES..... | 112 |
| TABLE 78: DOUBLE METHODS | 114 |
| TABLE 79: STRING PROPERTIES | 115 |

| | |
|------------------------------------|-----|
| TABLE 80: STRING METHODS..... | 125 |
| TABLE 81: STRING OPERATORS..... | 126 |
| TABLE 82: DATETIME PROPERTIES..... | 128 |
| TABLE 83: DATETIME METHODS..... | 133 |
| TABLE 84: DATETIME OPERATORS..... | 134 |
| TABLE 85: CUSTOM METHODS | 134 |

PREFACE

About This User Guide

This document outlines the available expression engine properties, methods, and operators, and provides their signatures, detailed descriptions, and official references as exposed by Integration Objects' Smart IoT Highway platform.

Target Audience

This document is intended for users, application engineers, and IT/OT integrators who are responsible for configuring Integration Objects' Smart IoT Highway, with a particular focus on Configuring expressions for Data Model attributes.

Document Conventions

| Convention | Description |
|-----------------|---|
| Bold | Bolded text indicates user interface elements, such as buttons, menu items, and dialog names. |
| (!) Note | Information to be noted |

Customer Support Services

| Phone | Email |
|---|---|
| Americas: +1 713 609 9208 | Support: customerservice@integrationobjects.com |
| Europe-Africa-Middle East +216 71 195 360 | Sales: sales@integrationobjects.com Online: www.integrationobjects.com |

INTRODUCTION

Smart IoT Highway (SIOTH®) is an advanced IT-OT integration platform designed to facilitate secure data exchange and transformation. It establishes secure end-to-end pipelines to collect and store data from edge IoT devices and various other sources. SIOTH® enables organizations of all sizes to easily connect applications, systems, and services in a managed, scalable, and secure environment. This comprehensive integration solution allows for seamless connectivity between IT and OT, enabling the conversion of industrial data into actionable intelligence and valuable insights.

The SIOTH® platform operates on robust functional architecture, as illustrated in the figure below:

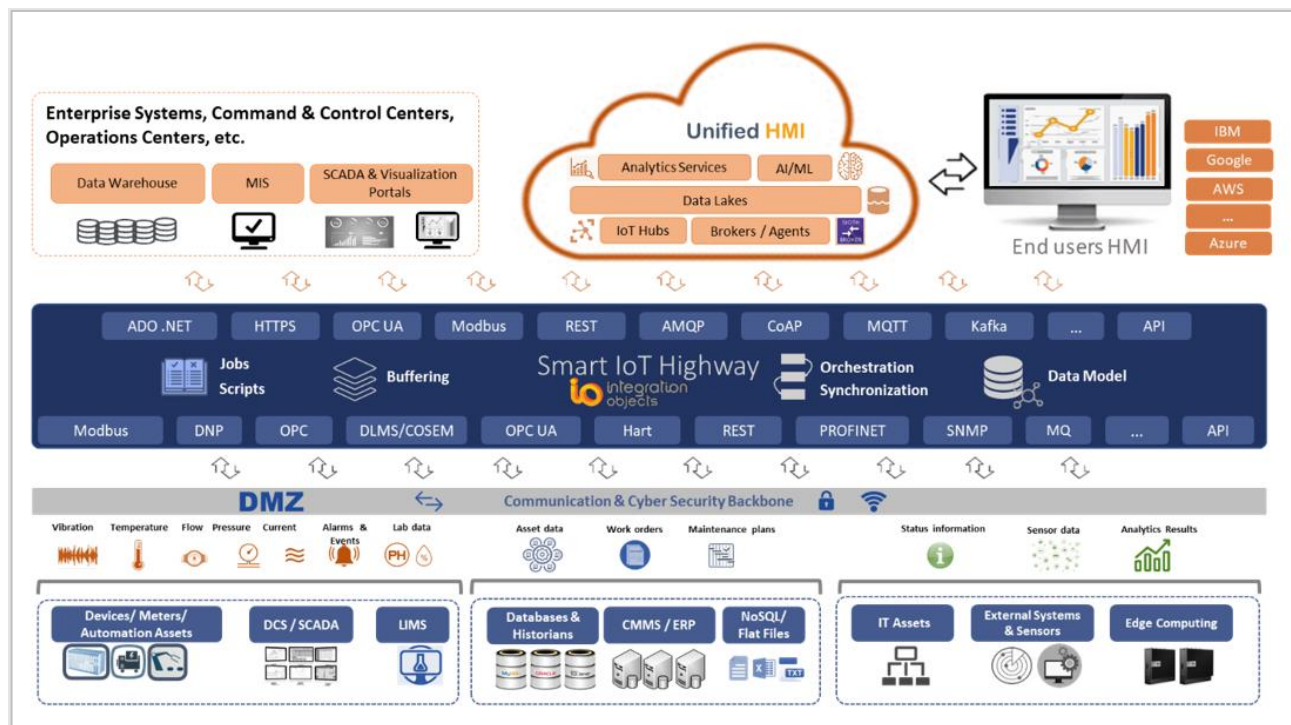


Figure 1: SIOTH® Platform Overview

SIOTH® Data Model provides a user-friendly, meaningful, unified, and hierarchical structure for organizing and managing data. It enables users to organize and interact with data collected from various sources in a structured and consistent way.

- **Hierarchical Organization**

Data elements are arranged in a hierarchical model, making it easier to navigate, interpret, and manage relationships between different data entities.

- **Object-Oriented Design**

The Data Model follows an object-oriented approach based on data element definitions known as classes.

- **Classes** define the attributes that describe the properties of the data elements, along with the methods that can be executed on them.
- **Instances** are concrete representations of classes that reference live data sets. These values can be constant, originate from SIOTH® connectors or be derived using expressions.

- **Integration with SIOTH® Connectors**

Instances of defined classes can directly reference data retrieved by source connectors, making them available across the platform, including the SIOTH® Job Engine rules and workflows.

- **Real-Time and Historical Data**


The model supports:

- **Real-time access** to the latest attribute value.
- **Historical data storage**, which can be leveraged in the SIOTH® Job Engine for analytics, monitoring, and rule execution.

GETTING STARTED

During **Data Model** configuration, attributes can be defined with values calculated using expressions. These expressions may reference other attributes, either within the same instance or across different instances, as well as constants, operators, and method calls.

To configure an attribute using an expression, follow these steps:

1. Navigate to **Instances** from the left-side menu
2. Perform one of the following actions:
 - Click **Create New Instance** in the **Instances Details** view or click the  icon next to either the **Instances Hierarchy** or an existing instance to create a new instance.

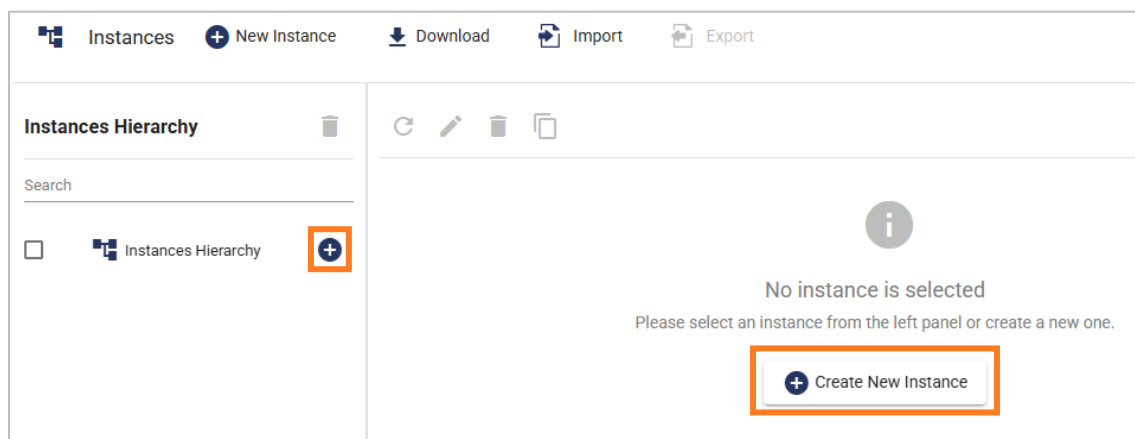


Figure 2: Create New Instance Buttons

- Click the instance and then click the **Edit Instance** icon  from the displayed **Instance Details** to edit an existing instance.

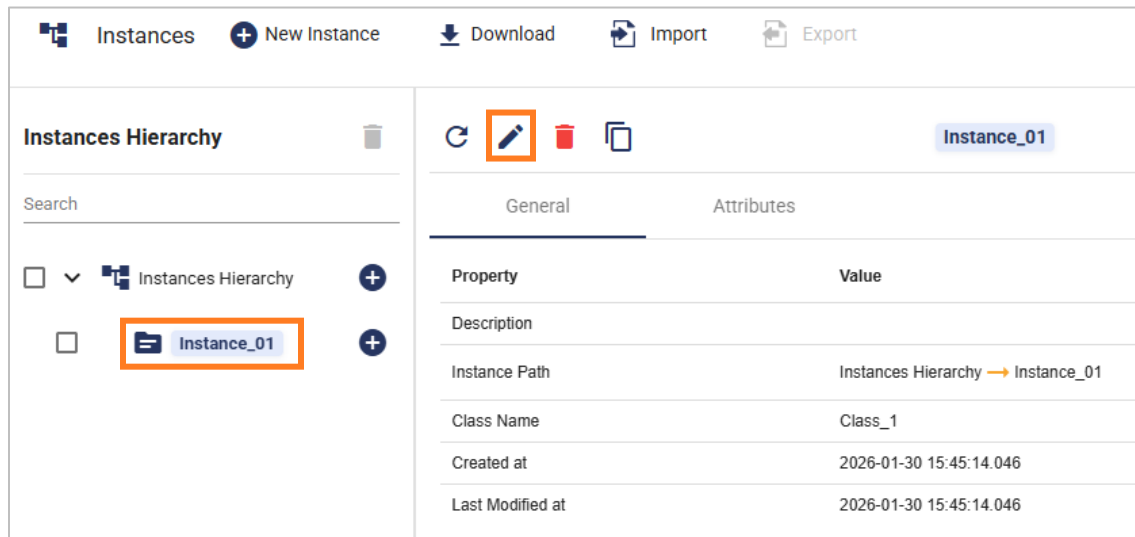



Figure 3: Edit Instance Button

- Click the **Edit** icon  next to the attribute you want to configure with an expression in the attributes section.




Edit Instance

Class Name: Class_1

Instance Name *: Instance_01

Description:
 Max 500 characters 0/500

Attributes

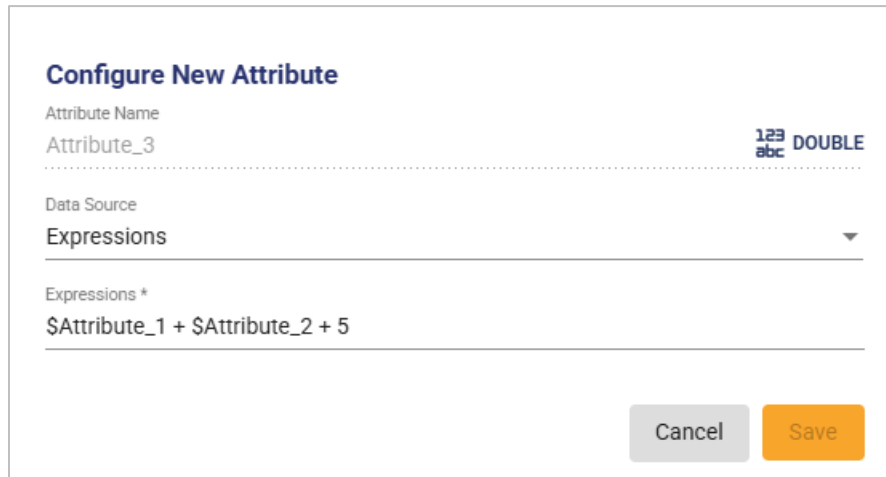
| <input type="checkbox"/> Archive | Name | Data Source | Value | |
|----------------------------------|-------------|-------------|-------|---|
| <input type="checkbox"/> | Attribute_1 | Static | 0 |  |
| <input type="checkbox"/> | Attribute_2 | Static | 0 |  |
| <input type="checkbox"/> | Attribute_3 | Static | 0 |  |

Items per page: 30 1 - 3 of 3 < >

Cancel Save

Figure 4: Edit Attribute Icon

4. Select **Expressions** from the **Data Source** drop-down menu.
5. Enter the required expression and click **Save** to apply the changes.



Configure New Attribute

Attribute Name
Attribute_3

Data Source
Expressions

Expressions *
\$Attribute_1 + \$Attribute_2 + 5

Cancel Save

Figure 5: Configuring an Attribute with an Expression

(!) Note

When configuring an expression, you can use reference attributes within the same instance and attributes from other instances.

- To reference attributes within the same instance, use the "\$" symbol before the attribute name.
Example: \$MyAttribute01 + 5
- To reference attributes from other instances, use the "\$" symbol before the instance name, followed by "." between the instance name and the attribute name.
Example: \$MyInstance01.MyAttribute01 + 5

Configure New Attribute

Attribute Name
Attribute_3

123 abc DOUBLE

Data Source
Expressions

Expressions *
\$Attribute_1 + \$Attribute_2 + \$Instance_01.Attribute1

Cancel Save

Figure 6: Expression with Reference to Other Attributes

This reference provides a comprehensive list of methods that can be used when defining expressions.

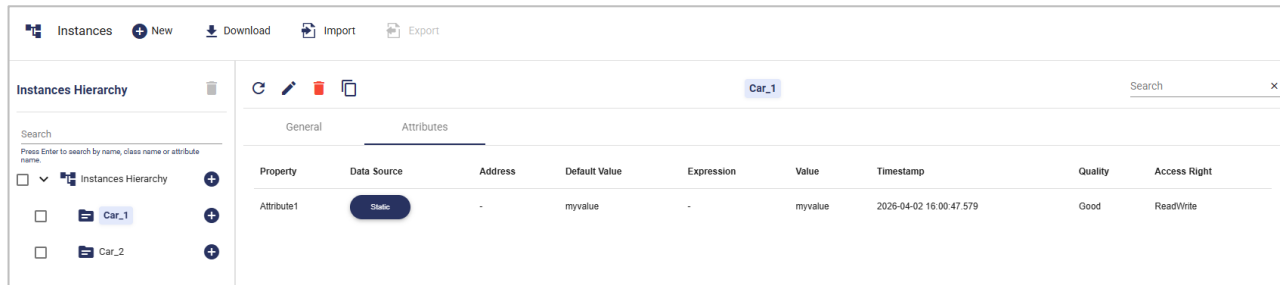
The available methods are grouped into the following categories:

- **Math:** Mathematical operations
- **Convert:** Type conversion operations
- **BitConverter:** Type-to-byte conversion operations
- **BitOperations:** Raw bit-level operations
- **Boolean:** Boolean operations
- **Char:** Unicode character operations
- **Byte:** 8-bit unsigned Byte operations
- **SByte:** 8-bit signed Byte operations
- **Int16:** 16-bit signed integer operations
- **Int32:** 32-bit signed integer operations
- **Int64:** 64-bit signed integer operations
- **UInt16:** 16-bit unsigned integer operations
- **UInt32:** 32-bit unsigned integer operations
- **UInt64:** 64-bit unsigned integer operations
- **Float:** Single-precision floating-point number operations.

- **Double:** Double-precision floating-point number operations
- **String:** Text operations
- **DateTime:** Date/Time operations
- **Custom:** Custom developed operations

Methods can be invoked either directly from their respective category or applied to a specific value or instance.

Select the desired instance and then click **Attributes** from the **Instance Details** section to view the current values of attributes.



The screenshot shows the 'Instances Hierarchy' on the left with 'Car_1' selected. The main panel displays the 'Attributes' tab for 'Car_1'. The table below represents the data shown in the 'Attributes' tab.

| Property | Data Source | Address | Default Value | Expression | Value | Timestamp | Quality | Access Right |
|------------|-------------|---------|---------------|------------|---------|-------------------------|---------|--------------|
| Attribute1 | Static | - | myvalue | - | myvalue | 2026-04-02 16:00:47.579 | Good | ReadWrite |

Figure 7: Instance Current Values View

DATA MODEL EXPRESSIONS

1. Mathematical Operations

The **Math** category provides a set of properties and methods used to perform common mathematical operations.

1.1. Math Properties

| Property | Type | Description |
|-------------------|--------|--|
| <i>E</i> | Double | Represents the base of the natural logarithm (e). |
| <i>PI</i> | Double | Represents the ratio of a circle's circumference to its diameter (π). |
| <i>Tau</i> | Double | Represents the number of radians in one full rotation (τ). |

Table 1: Math Properties

1.2. Math Methods

1.2.1. Abs

| Method | Return Type | Description |
|---------------------------|-------------|---|
| <i>Abs(double)</i> | Double | Returns the absolute value of a double-precision floating-point number. |

| | | |
|--------------------------|-------|---|
| <i>Abs(float)</i> | Float | Returns the absolute value of a single-precision floating-point number. |
| <i>Abs(int16)</i> | Int16 | Returns the absolute value of a 16-bit signed integer. |
| <i>Abs(int32)</i> | Int32 | Returns the absolute value of a 32-bit signed integer. |
| <i>Abs(int64)</i> | Int64 | Returns the absolute value of a 64-bit signed integer. |
| <i>Abs(SByte)</i> | SByte | Returns the absolute value of an 8-bit signed integer. |

Table 2: Math.Abs Method

1.2.2. Acos, Acosh, Cos, Cosh

| Method | Return Type | Description |
|-----------------------------|-------------|---|
| <i>Acos(double)</i> | Double | Returns the angle whose cosine is the specified value. |
| <i>Acosh(double)</i> | Double | Returns the angle whose hyperbolic cosine is the specified value. |
| <i>Cos(double)</i> | Double | Returns the cosine of the specified angle. |
| <i>Cosh(double)</i> | Double | Returns the hyperbolic cosine of the specified angle. |

Table 3: Math.Acos, Math.Acosh, Math.Cos, Math.Cosh Methods

1.2.3. Asin, Asinh, Sin, Sinh

| Method | Return Type | Description |
|-----------------------------|-------------|---|
| <i>Asin(double)</i> | Double | Returns the angle whose sine is the specified value. |
| <i>Asinh(double)</i> | Double | Returns the angle whose hyperbolic sine is the specified value. |
| <i>Sin(double)</i> | Double | Returns the sine of the specified angle. |
| <i>Sinh(double)</i> | Double | Returns the hyperbolic sine of the specified angle. |

Table 4: Math.Asin, Math.Asinh, Math.Sin, Math.Sinh Methods

1.2.4. Atan, Atanh, Atan2, Tan, Tanh

| Method | Return Type | Description |
|--|-------------|--|
| <i>Atan(double)</i> | Double | Returns the angle whose tangent is the specified value. |
| <i>Atanh(double)</i> | Double | Returns the angle whose hyperbolic tangent is the specified value. |
| <i>Math.Atan2(double, double)</i> | Double | Returns the angle whose tangent is the quotient of two specified values. |
| <i>Tan(double)</i> | Double | Returns the tangent of the specified angle. |
| <i>Tanh(double)</i> | Double | Returns the hyperbolic tangent of the specified angle. |

Table 5: Math.Atan, Math.Atanh, Math.Atan2, Math.Tan, Math.Tanh Methods

1.2.5. BigMul

| Method | Return Type | Description |
|-----------------------------|-------------|---|
| BigMul(int32, int32) | Int64 | Returns the full product of two 32-bit signed integers as a 64-bit integer. |

Table 6: Math.BigMul Method

1.2.6. BitDecrement

| Method | Return Type | Description |
|------------------------------------|-------------|--|
| <i>BitDecrement(double)</i> | Double | Returns the largest double-precision floating-point value that is less than the specified value. |

Table 7: Math.BitDecrement Method

1.2.7. BitIncrement

| Method | Return Type | Description |
|------------------------------------|-------------|--|
| <i>BitIncrement(double)</i> | Double | Returns the smallest double-precision floating-point value that is greater than the specified value. |

Table 8: Math.BitIncrement Method

1.2.8. Cbrt

| Method | Return Type | Description |
|----------------------------|-------------|--|
| <i>Cbrt(double)</i> | Double | Returns the cube root of the specified number. |

Table 9: Math.Cbrt Method

1.2.9. Ceiling

| Method | Return Type | Description |
|-------------------------------|-------------|--|
| <i>Ceiling(double)</i> | Double | Returns the smallest integral value that is greater than or equal to the specified double-precision floating-point number. |

Table 10: Math.Ceiling Method

1.2.10. Clamp

| Method | Return Type | Description |
|---|-------------|---|
| <i>Clamp(double, double, double)</i> | Double | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(float, float, float)</i> | Float | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(int16, int16, int16)</i> | Int16 | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |

| | | |
|---|--------|---|
| <i>Clamp(int32, int32, int32)</i> | Int32 | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(int64, int64, int64)</i> | Int64 | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(uint16, uint16, uint16)</i> | UInt16 | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(uint32, uint32, uint32)</i> | UInt32 | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(uint64, uint64, uint64)</i> | UInt64 | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(SByte, SByte, SByte)</i> | SByte | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |
| <i>Clamp(Byte, Byte, Byte)</i> | Byte | Returns the value clamped to the inclusive range defined by the specified minimum and maximum values. |

Table 11: Math.Clamp Method

1.2.11. CopySign

| Method | Return Type | Description |
|--|-------------|---|
| <i>CopySign(double, double)</i> | Double | Returns a value with the magnitude of the first argument and the sign of the second argument. |

Table 12: Math.CopySign Method

1.2.12. Exp

| Method | Return Type | Description |
|---------------------------|-------------|--|
| <i>Exp(double)</i> | Double | Returns “e” raised to the specified power. |

Table 13: Math.Exp Method

1.2.13. FusedMultiplyAdd

| Method | Return Type | Description |
|--|-------------|--|
| <i>FusedMultiplyAdd(double, double, double)</i> | Double | Returns $(x \times y) + z$, computed as a single ternary operation with one rounding. |

Table 14: Math.FusedMultiplyAdd Method

1.2.14. IEEERemainder

| Method | Return Type | Description |
|---|-------------|---|
| <i>IEEERemainder(double, double)</i> | Double | Returns the remainder resulting from the division of the first specified value by the second. |

Table 15: Math.IEEERemainder Method

1.2.15. Log

| Method | Return Type | Description |
|-----------------------------------|-------------|--|
| <i>Log(double, double)</i> | Double | Returns the logarithm of the specified value using the specified base. |
| <i>Log(double)</i> | Double | Returns the natural logarithm (base e) of the specified value. |
| <i>Log10(double)</i> | Double | Returns the base 10 logarithm of the specified value. |
| <i>Log2(double)</i> | Double | Returns the base 2 logarithm of the specified value. |
| <i>ILogB(double)</i> | Int32 | Returns the base-2 integer logarithm of the specified value. |

Table 16: Math.Log Methods

1.2.16. Max

| Method | Return Type | Description |
|-----------------------------------|-------------|--|
| <i>Max(double, double)</i> | Double | Returns the larger of two double-precision floating-point numbers. |
| <i>Max(float, float)</i> | Float | Returns the larger of two single-precision floating-point numbers. |
| <i>Max(int16, int16)</i> | Int16 | Returns the larger of two 16-bit signed integers. |
| <i>Max(int32, int32)</i> | Int32 | Returns the larger of two 32-bit signed integers. |

| | | |
|-----------------------------------|--------|---|
| <i>Max(int64 , int64)</i> | Int64 | Returns the larger of two 64-bit signed integers. |
| <i>Max(uint16, uint16)</i> | UInt16 | Returns the larger of two 16-bit unsigned integers. |
| <i>Max(uint32, uint32)</i> | UInt32 | Returns the larger of two 32-bit unsigned integers. |
| <i>Max(uint64, uint64)</i> | UInt64 | Returns the larger of two 64-bit unsigned integers. |
| <i>Max(SByte, SByte)</i> | SByte | Returns the larger of two 8-bit signed integers. |
| <i>Max(Byte, Byte)</i> | Byte | Returns the larger of two 8-bit signed integers. |

Table 17: Math.Max Method

1.2.17. MaxMagnitude

| Method | Return Type | Description |
|--|-------------|--|
| <i>MaxMagnitude(double, double)</i> | Double | Returns the value with the larger absolute magnitude of the two double-precision floating-point numbers. |

Table 18: Math.MaxMagnitude Method

1.2.18. Min

| Method | Return Type | Description |
|-----------------------------------|-------------|---|
| <i>Min(double, double)</i> | Double | Returns the smaller of two double-precision floating-point numbers. |

| | | |
|-----------------------------------|--------|---|
| <i>Min(float, float)</i> | Float | Returns the smaller of two single-precision floating-point numbers. |
| <i>Min(int16, int16)</i> | Int16 | Returns the smaller of two 16-bit signed integers. |
| <i>Min(int32, int32)</i> | Int32 | Returns the smaller of two 32-bit signed integers. |
| <i>Min(int64, int64)</i> | Int64 | Returns the smaller of two 64-bit signed integers. |
| <i>Min(uint16, uint16)</i> | UInt16 | Returns the smaller of two 16-bit unsigned integers. |
| <i>Min(uint32, uint32)</i> | UInt32 | Returns the smaller of two 32-bit unsigned integers. |
| <i>Min(uint64, uint64)</i> | UInt64 | Returns the smaller of two 64-bit unsigned integers. |
| <i>Min(SByte, SByte)</i> | SByte | Returns the smaller of two 8-bit signed integers. |
| <i>Min(Byte, Byte)</i> | Byte | Returns the smaller of two 8-bit signed integers. |

Table 19: Math.Min Method

1.2.19. MinMagnitude

| Method | Return Type | Description |
|--|-------------|---|
| <i>MinMagnitude(double, double)</i> | Double | Returns the value with the smaller absolute magnitude of the two double-precision floating-point numbers. |

Table 20: Math.MinMagnitude Method

1.2.20. Pow

| Method | Return Type | Description |
|-----------------------------------|-------------|--|
| <i>Pow(double, double)</i> | Double | Returns the first argument raised to the power of the second argument. |

Table 21: Math.Pow Method

1.2.21. ReciprocalEstimate

| Method | Return Type | Description |
|--|-------------|--|
| <i>ReciprocalEstimate(double)</i> | Double | Returns an estimated reciprocal (1/x) of the specified number. |

Table 22: Math.ReciprocalEstimate Method

1.2.22. Round

| Method | Return Type | Description |
|------------------------------------|-------------|---|
| <i>Round(double, int32)</i> | Double | Rounds a double-precision value to the specified number of fractional digits. |

| | | |
|--|--------|---|
| | | Midpoint values are rounded to the nearest even number. |
| <i>RoundToEven(double, int32)</i> | Double | Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values to the nearest number, and when a number is halfway between two others, it's rounded toward the nearest even number. |
| <i>RoundAwayFromZero(double, int32)</i> | Double | Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values to the nearest number, and when a number is halfway between two others, it's rounded toward the nearest number that's away from zero. |
| <i>RoundToZero(double, int32)</i> | Double | Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values toward zero, with the result closest to and no greater in magnitude than the infinitely precise result. |
| <i>RoundToNegativeInfinity(double, int32)</i> | Double | Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values using downwards-directed rounding, with the result closest to and no greater than the infinitely precise result. |

| | | |
|--|--------|---|
| <i>RoundToPositiveInfinity(double, int32)</i> | Double | Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values using upwards-directed rounding, with the result closest to and no less than the infinitely precise result. |
| <i>Round(double)</i> | Double | Rounds a decimal value to the nearest integral value and rounds midpoint values to the nearest even number. |
| <i>RoundToEven(double)</i> | Double | Rounds a decimal value to the nearest integral value, and rounds midpoint values to the nearest number, and when a number is halfway between two others, it's rounded toward the nearest even number. |
| <i>RoundAwayFromZero(double)</i> | Double | Rounds a decimal value to the nearest integral value, and rounds midpoint values to the nearest number, and when a number is halfway between two others, it's rounded toward the nearest number that's away from zero. |
| <i>RoundToZero(double)</i> | Double | Rounds a decimal value to the nearest integral value, and rounds midpoint values toward zero, with the result closest to and no greater in magnitude than the infinitely precise result. |

| | | |
|---|--------|---|
| <i>RoundToNegativeInfinity(double)</i> | Double | Rounds a decimal value to the nearest integral value, and rounds midpoint values using downwards-directed rounding, with the result closest to and no greater than the infinitely precise result. |
| <i>RoundToPositiveInfinity(double)</i> | Double | Rounds a decimal value to the nearest integral value, and rounds midpoint values using upwards-directed rounding, with the result closest to and no less than the infinitely precise result. |

Table 23: Math.Round Methods

1.2.23. ScaleB

| Method | Return Type | Description |
|-------------------------------------|-------------|---|
| <i>ScaleB(double, int32)</i> | Double | Returns $x * 2^n$ computed efficiently. |

Table 24: Math.ScaleB Method

1.2.24. Sign

| Method | Return Type | Description |
|---------------------------|-------------|--|
| <i>Sign(SByte)</i> | Int32 | Returns an integer that indicates the sign of an 8-bit signed integer. |
| <i>Sign(int16)</i> | Int32 | Returns an integer that indicates the sign of a 16-bit signed integer. |
| <i>Sign(int32)</i> | Int32 | Returns an integer that indicates the sign of a 32-bit signed integer. |

| | | |
|----------------------------|-------|---|
| <i>Sign(int64)</i> | Int32 | Returns an integer that indicates the sign of a 64-bit signed integer. |
| <i>Sign(float)</i> | Int32 | Returns an integer that indicates the sign of a single-precision floating-point number. |
| <i>Sign(double)</i> | Int32 | Returns an integer that indicates the sign of a double-precision floating-point number. |

Table 25: Math.Sign Method

1.2.25. Sqrt

| Method | Return Type | Description |
|----------------------------|-------------|--|
| <i>Sqrt(double)</i> | Double | Returns the square root of a specified number. |

Table 26: Math.Sqrt Method

1.2.26. Truncate

| Method | Return Type | Description |
|--------------------------------|-------------|--|
| <i>Truncate(double)</i> | Double | Calculates an integral part of a specified double-precision floating-point number. |

Table 27: Math.Truncate Method

1.3. Math Examples

General

Attributes

| Property | Address | Default Value | Regular Expression | Value | Timestamp |
|---------------|--------------------|---------------|--------------------|-------------------|-------------------------|
| MyAttribute01 | Regular Expression | | Math.E | 2.718281828459045 | 2025-11-10 11:45:17.956 |
| MyAttribute02 | Regular Expression | | Math.PI | 3.141592653589793 | 2025-11-10 11:45:17.956 |
| MyAttribute03 | Regular Expression | | Math.Tau | 6.283185307179586 | 2025-11-10 11:45:17.956 |

Figure 8: Math Properties Example

General

Attributes

| Property | Address | Default Value | Regular Expression | Value | Timestamp |
|---------------|--------------------|---------------|-------------------------|--------|-------------------------|
| MyAttribute01 | Regular Expression | | Math.Abs(-5) | 5 | 2025-11-10 12:19:16.709 |
| MyAttribute02 | Regular Expression | | Math.Max(32563, 478625) | 478625 | 2025-11-10 12:19:16.710 |
| MyAttribute03 | Regular Expression | | Math.Clamp(100, 0, 255) | 100 | 2025-11-10 12:19:16.712 |

Figure 9: Math Methods Example

2. Convert Operations

The **Convert** category provides a collection of methods for converting values between different data types.

2.1. Convert Methods

2.1.1. ToDouble

| Method | Return Type | Description |
|-------------------------------|-------------|--|
| <i>ToDouble(bool)</i> | Double | Converts the specified Boolean value to the equivalent double-precision floating-point number. |
| <i>ToDouble(Byte)</i> | Double | Converts the value of the specified 8-bit unsigned integer to the equivalent double-precision floating-point number. |
| <i>ToDouble(SByte)</i> | Double | Converts the value of the specified 8-bit signed integer to the equivalent double-precision floating-point number. |
| <i>ToDouble(int16)</i> | Double | Converts the value of the specified 16-bit signed integer to an equivalent double-precision floating-point number. |
| <i>ToDouble(int32)</i> | Double | Converts the value of the specified 32-bit signed integer to an equivalent double-precision floating-point number. |
| <i>ToDouble(int64)</i> | Double | Converts the value of the specified 64-bit signed integer to an equivalent double-precision floating-point number. |

| | | |
|--------------------------------|--------|---|
| <i>ToDouble(uint16)</i> | Double | Converts the value of the specified 16-bit unsigned integer to an equivalent double-precision floating-point number. |
| <i>ToDouble(uint32)</i> | Double | Converts the value of the specified 32-bit unsigned integer to an equivalent double-precision floating-point number. |
| <i>ToDouble(uint64)</i> | Double | Converts the value of the specified 64-bit unsigned integer to an equivalent double-precision floating-point number. |
| <i>ToDouble(float)</i> | Double | Converts the value of the specified single-precision floating-point number to an equivalent double-precision floating-point number. |
| <i>ToDouble(double)</i> | Double | Returns the specified double-precision floating-point number; no actual conversion is performed. |
| <i>ToDouble(string)</i> | Double | Converts the specified string representation of a number to an equivalent double-precision floating-point number. |

Table 28: Convert.ToDouble Method

2.1.2. ToByte

| Method | Return Type | Description |
|------------------------------------|-------------|--|
| <i>ToByte(bool)</i> | Byte | Converts the specified Boolean value to the equivalent 8-bit unsigned integer. |
| <i>ToByte(Byte)</i> | Byte | Returns the specified 8-bit unsigned integer, no actual conversion is performed. |
| <i>ToByte(SByte)</i> | Byte | Converts the value of the specified 8-bit signed integer to an equivalent 8-bit unsigned integer. |
| <i>ToByte(int16)</i> | Byte | Converts the value of the specified 16-bit signed integer to an equivalent 8-bit unsigned integer. |
| <i>ToByte(int32 value)</i> | Byte | Converts the value of the specified 32-bit signed integer to an equivalent 8-bit unsigned integer. |
| <i>ToByte(int64 value)</i> | Byte | Converts the value of the specified 64-bit signed integer to an equivalent 8-bit unsigned integer. |
| <i>ToByte(uint16 value)</i> | Byte | Converts the value of the specified 16-bit unsigned integer to an equivalent 8-bit unsigned integer. |

| | | |
|---|------|---|
| <i>ToByte(uint32 value)</i> | Byte | Converts the value of the specified 32-bit unsigned integer to an equivalent 8-bit unsigned integer. |
| <i>ToByte(uint64 value)</i> | Byte | Converts the value of the specified 64-bit unsigned integer to an equivalent 8-bit unsigned integer. |
| <i>ToByte(float value)</i> | Byte | Converts the value of the specified single-precision floating-point number to an equivalent 8-bit unsigned integer. |
| <i>ToByte(double value)</i> | Byte | Converts the value of the specified double-precision floating-point number to an equivalent 8-bit unsigned integer. |
| <i>ToByte(string value)</i> | Byte | Converts the specified string representation of a number to an equivalent 8-bit unsigned integer. |
| <i>ToByte(string value, int32)</i> | Byte | Converts the string representation of a number in a specified base to an equivalent 8-bit unsigned integer. |
| <i>ToByte(char value)</i> | Byte | Converts the value of the specified Unicode character to the equivalent 8-bit unsigned integer. |

Table 29: Convert.ToByte Method

2.1.3. ToSByte

| Method | Return Type | Description |
|-------------------------------|-------------|---|
| <i>ToSByte(bool)</i> | SByte | Converts the specified Boolean value to the equivalent 8-bit signed integer. |
| <i>ToSByte(Byte)</i> | SByte | Converts the value of the specified 8-bit unsigned integer to the equivalent 8-bit signed integer. |
| <i>ToSByte(SByte)</i> | SByte | Returns the specified 8-bit signed integer; no actual conversion is performed. |
| <i>ToSByte(int16)</i> | SByte | Converts the value of the specified 16-bit signed integer to the equivalent 8-bit signed integer. |
| <i>ToSByte(int32)</i> | SByte | Converts the value of the specified 32-bit signed integer to the equivalent 8-bit signed integer. |
| <i>ToSByte(int64)</i> | SByte | Converts the value of the specified 64-bit signed integer to the equivalent 8-bit signed integer. |
| <i>ToSByte(uint16)</i> | SByte | Converts the value of the specified 16-bit unsigned integer to the equivalent 8-bit signed integer. |
| <i>ToSByte(uint32)</i> | SByte | Converts the value of the specified 32-bit unsigned integer to the equivalent 8-bit signed integer. |

| | | |
|--------------------------------------|-------|---|
| <i>ToSByte(uint64)</i> | SByte | Converts the value of the specified 64-bit unsigned integer to the equivalent 8-bit signed integer. |
| <i>ToSByte(float)</i> | SByte | Converts the value of the specified single-precision floating-point number to an equivalent 8-bit signed integer. |
| <i>ToSByte(double)</i> | SByte | Converts the value of the specified double-precision floating-point number to an equivalent 8-bit signed integer. |
| <i>ToSByte(string)</i> | SByte | Converts the specified string representation of a number to an equivalent 8-bit signed integer. |
| <i>ToSByte(string, int32)</i> | SByte | Converts the string representation of a number in a specified base to an equivalent 8-bit signed integer. |
| <i>ToSByte(char)</i> | SByte | Converts the value of the specified Unicode character to the equivalent 8-bit signed integer. |

Table 30: Convert.ToSByte Method

2.1.4. ToInt16

| Method | Return Type | Description |
|-----------------------------|-------------|---|
| <i>ToInt16(bool)</i> | Int16 | Converts the specified Boolean value to the equivalent 16-bit signed integer. |

| | | |
|-------------------------------|-------|---|
| <i>ToInt16(Byte)</i> | Int16 | Converts the value of the specified 8-bit unsigned integer to the equivalent 16-bit signed integer. |
| <i>ToInt16(SByte)</i> | Int16 | Converts the value of the specified 8-bit signed integer to the equivalent 16-bit signed integer. |
| <i>ToInt16(int16)</i> | Int16 | Returns the specified 16-bit signed integer, no actual conversion is performed. |
| <i>ToInt16(int32)</i> | Int16 | Converts the value of the specified 32-bit signed integer to an equivalent 16-bit signed integer. |
| <i>ToInt16(int64)</i> | Int16 | Converts the value of the specified 64-bit signed integer to an equivalent 16-bit signed integer. |
| <i>ToInt16(uint16)</i> | Int16 | Converts the value of the specified 16-bit unsigned integer to an equivalent 16-bit signed integer. |
| <i>ToInt16(uint32)</i> | Int16 | Converts the value of the specified 32-bit unsigned integer to an equivalent 16-bit signed integer. |
| <i>ToInt16(uint64)</i> | Int16 | Converts the value of the specified 64-bit unsigned integer to an equivalent 16-bit signed integer. |

| | | |
|--------------------------------------|-------|--|
| <i>ToInt16(float)</i> | Int16 | Converts the value of the specified single-precision floating-point number to an equivalent 16-bit signed integer. |
| <i>ToInt16(double)</i> | Int16 | Converts the value of the specified double-precision floating-point number to an equivalent 16-bit signed integer. |
| <i>ToInt16(string)</i> | Int16 | Converts the specified string representation of a number to an equivalent 16-bit signed integer. |
| <i>ToInt16(string, int32)</i> | Int16 | Converts the string representation of a number in a specified base to an equivalent 16-bit signed integer. |
| <i>ToInt16(char)</i> | Int16 | Converts the value of the specified Unicode character to the equivalent 16-bit signed integer. |

Table 31: Convert.ToInt16 Method

2.1.5. ToInt32

| Method | Return Type | Description |
|-----------------------------|-------------|---|
| <i>ToInt32(bool)</i> | Int32 | Converts the specified Boolean value to the equivalent 32-bit signed integer. |
| <i>ToInt32(Byte)</i> | Int32 | Converts the value of the specified 8-bit unsigned integer to the equivalent 32-bit signed integer. |

| | | |
|-------------------------------|-------|--|
| <i>ToInt32(SByte)</i> | Int32 | Converts the value of the specified 8-bit signed integer to the equivalent 32-bit signed integer. |
| <i>ToInt32(int16)</i> | Int32 | Converts the value of the specified 16-bit signed integer to an equivalent 32-bit signed integer. |
| <i>ToInt32(int32)</i> | Int32 | Returns the specified 32-bit signed integer; no actual conversion is performed. |
| <i>ToInt32(int64)</i> | Int32 | Converts the value of the specified 64-bit signed integer to an equivalent 32-bit signed integer. |
| <i>ToInt32(uint16)</i> | Int32 | Converts the value of the specified 16-bit unsigned integer to an equivalent 32-bit signed integer. |
| <i>ToInt32(uint32)</i> | Int32 | Converts the value of the specified 32-bit unsigned integer to an equivalent 32-bit signed integer. |
| <i>ToInt32(uint64)</i> | Int32 | Converts the value of the specified 64-bit unsigned integer to an equivalent 32-bit signed integer. |
| <i>ToInt32(float)</i> | Int32 | Converts the value of the specified single-precision floating-point number to an equivalent 32-bit signed integer. |

| | | |
|--------------------------------------|-------|--|
| <i>ToInt32(double)</i> | Int32 | Converts the value of the specified double-precision floating-point number to an equivalent 32-bit signed integer. |
| <i>ToInt32(string)</i> | Int32 | Converts the specified string representation of a number to an equivalent 32-bit signed integer. |
| <i>ToInt32(string, int32)</i> | Int32 | Converts the string representation of a number in a specified base to an equivalent 32-bit signed integer. |
| <i>ToInt32(char)</i> | Int32 | Converts the value of the specified Unicode character to the equivalent 32-bit signed integer. |

Table 32: Convert.ToInt32 Method

2.1.6. ToInt64

| Method | Return Type | Description |
|------------------------------|-------------|---|
| <i>ToInt64(bool)</i> | Int64 | Converts the specified Boolean value to the equivalent 64-bit signed integer. |
| <i>ToInt64(Byte)</i> | Int64 | Converts the value of the specified 8-bit unsigned integer to the equivalent 64-bit signed integer. |
| <i>ToInt64(SByte)</i> | Int64 | Converts the value of the specified 8-bit signed integer to the equivalent 64-bit signed integer. |

| | | |
|-------------------------------|-------|--|
| <i>ToInt64(int16)</i> | Int64 | Converts the value of the specified 16-bit signed integer to an equivalent 64-bit signed integer. |
| <i>ToInt64(int32)</i> | Int64 | Converts the value of the specified 32-bit signed integer to an equivalent 64-bit signed integer. |
| <i>ToInt64(int64)</i> | Int64 | Returns the specified 64-bit signed integer; no actual conversion is performed. |
| <i>ToInt64(uint16)</i> | Int64 | Converts the value of the specified 16-bit unsigned integer to an equivalent 64-bit signed integer. |
| <i>ToInt64(uint32)</i> | Int64 | Converts the value of the specified 32-bit unsigned integer to an equivalent 64-bit signed integer. |
| <i>ToInt64(uint64)</i> | Int64 | Converts the value of the specified 64-bit unsigned integer to an equivalent 64-bit signed integer. |
| <i>ToInt64(float)</i> | Int64 | Converts the value of the specified single-precision floating-point number to an equivalent 64-bit signed integer. |

| | | |
|--------------------------------------|-------|--|
| <i>ToInt64(double)</i> | Int64 | Converts the value of the specified double-precision floating-point number to an equivalent 64-bit signed integer. |
| <i>ToInt64(string)</i> | Int64 | Converts the specified string representation of a number to an equivalent 64-bit signed integer. |
| <i>ToInt64(string, int32)</i> | Int64 | Converts the string representation of a number in a specified base to an equivalent 64-bit signed integer. |
| <i>ToInt64(char)</i> | Int64 | Converts the value of the specified Unicode character to the equivalent 64-bit signed integer. |

Table 33: Convert.ToInt64 Method

2.1.7. ToUInt16

| Method | Return Type | Description |
|------------------------------|-------------|---|
| <i>ToUInt16(bool)</i> | UInt16 | Converts the specified Boolean value to the equivalent 16-bit unsigned integer. |
| <i>ToUInt16(Byte)</i> | UInt16 | Converts the value of the specified 8-bit unsigned integer to the equivalent 16-bit unsigned integer. |

| | | |
|--------------------------------|--------|---|
| <i>ToUInt16(SByte)</i> | UInt16 | Converts the value of the specified 8-bit signed integer to the equivalent 16-bit unsigned integer. |
| <i>ToUInt16(int16)</i> | UInt16 | Converts the value of the specified 16-bit signed integer to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(int32)</i> | UInt16 | Converts the value of the specified 32-bit signed integer to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(int64)</i> | UInt16 | Converts the value of the specified 64-bit signed integer to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(uint16)</i> | UInt16 | Returns the specified 16-bit unsigned integer; no actual conversion is performed. |
| <i>ToUInt16(uint32)</i> | UInt16 | Converts the value of the specified 32-bit unsigned integer to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(uint64)</i> | UInt16 | Converts the value of the specified 64-bit unsigned integer to an equivalent 16-bit unsigned integer. |

| | | |
|---------------------------------------|--------|--|
| <i>ToUInt16(float)</i> | UInt16 | Converts the value of the specified single-precision floating-point number to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(double)</i> | UInt16 | Converts the value of the specified double-precision floating-point number to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(string)</i> | UInt16 | Converts the specified string representation of a number to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(string, int32)</i> | UInt16 | Converts the string representation of a number in a specified base to an equivalent 16-bit unsigned integer. |
| <i>ToUInt16(char)</i> | UInt16 | Converts the value of the specified Unicode character to the equivalent 16-bit unsigned integer. |

Table 34: Convert.ToUInt16 Method

2.1.8. ToUInt32

| Method | Return Type | Description |
|------------------------------|-------------|---|
| <i>ToUInt32(bool)</i> | UInt32 | Converts the specified Boolean value to the equivalent 32-bit unsigned integer. |

| | | |
|--------------------------------|--------|---|
| <i>ToUInt32(Byte)</i> | UInt32 | Converts the value of the specified 8-bit unsigned integer to the equivalent 32-bit unsigned integer. |
| <i>ToUInt32(SByte)</i> | UInt32 | Converts the value of the specified 8-bit signed integer to the equivalent 32-bit unsigned integer. |
| <i>ToUInt32(int16)</i> | UInt32 | Converts the value of the specified 16-bit signed integer to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(int32)</i> | UInt32 | Converts the value of the specified 32-bit signed integer to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(int64)</i> | UInt32 | Converts the value of the specified 64-bit signed integer to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(uint16)</i> | UInt32 | Converts the value of the specified 16-bit unsigned integer to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(uint32)</i> | UInt32 | Returns the specified 32-bit unsigned integer, no actual conversion is performed. |

| | | |
|---------------------------------------|--------|--|
| <i>ToUInt32(uint64)</i> | UInt32 | Converts the value of the specified 64-bit unsigned integer to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(float value)</i> | UInt32 | Converts the value of the specified single-precision floating-point number to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(double)</i> | UInt32 | Converts the value of the specified double-precision floating-point number to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(string)</i> | UInt32 | Converts the specified string representation of a number to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(string, int32)</i> | UInt32 | Converts the string representation of a number in a specified base to an equivalent 32-bit unsigned integer. |
| <i>ToUInt32(char)</i> | UInt32 | Converts the value of the specified Unicode character to the equivalent 32-bit unsigned integer. |

Table 35: Convert.ToUInt32 Method

2.1.9. ToUInt64

| Method | Return Type | Description |
|--------------------------------|-------------|---|
| <i>ToUInt64(bool)</i> | UInt64 | Converts the specified Boolean value to the equivalent 64-bit unsigned integer. |
| <i>ToUInt64(Byte)</i> | UInt64 | Converts the value of the specified 8-bit unsigned integer to the equivalent 64-bit unsigned integer. |
| <i>ToUInt64(SByte)</i> | UInt64 | Converts the value of the specified 8-bit signed integer to the equivalent 64-bit unsigned integer. |
| <i>ToUInt64(int16)</i> | UInt64 | Converts the value of the specified 16-bit signed integer to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(int32)</i> | UInt64 | Converts the value of the specified 32-bit signed integer to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(int64)</i> | UInt64 | Converts the value of the specified 64-bit signed integer to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(uint16)</i> | UInt64 | Converts the value of the specified 16-bit unsigned integer to an equivalent 64-bit unsigned integer. |

| | | |
|---------------------------------------|--------|--|
| <i>ToUInt64(uint32)</i> | UInt64 | Converts the value of the specified 32-bit unsigned integer to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(uint64)</i> | UInt64 | Returns the specified 64-bit unsigned integer; no actual conversion is performed. |
| <i>ToUInt64(float)</i> | UInt64 | Converts the value of the specified single-precision floating-point number to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(double)</i> | UInt64 | Converts the value of the specified double-precision floating-point number to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(string)</i> | UInt64 | Converts the specified string representation of a number to an equivalent 64-bit unsigned integer. |
| <i>ToUInt64(string, int32)</i> | UInt64 | Converts the string representation of a number in a specified base to an equivalent 32-bit unsigned integer. |
| <i>ToUInt64(char)</i> | UInt64 | Converts the string representation of a number in a specified base to an equivalent 64-bit unsigned integer. |

Table 36: Convert.ToUInt64 Method

2.1.10. ToSingle

| Method | Return Type | Description |
|-------------------------------|-------------|--|
| <i>ToSingle(bool)</i> | Float | Converts the specified Boolean value to the equivalent single-precision floating-point number. |
| <i>ToSingle(Byte)</i> | Float | Converts the value of the specified 8-bit unsigned integer to the equivalent single-precision floating-point number. |
| <i>ToSingle(SByte)</i> | Float | Converts the value of the specified 8-bit signed integer to the equivalent single-precision floating-point number. |
| <i>ToSingle(int16)</i> | Float | Converts the value of the specified 16-bit signed integer to an equivalent single-precision floating-point number. |
| <i>ToSingle(int32)</i> | Float | Converts the value of the specified 32-bit signed integer to an equivalent single-precision floating-point number. |
| <i>ToSingle(int64)</i> | Float | Converts the value of the specified 64-bit signed integer to an equivalent single-precision floating-point number. |

| | | |
|--------------------------------|-------|---|
| <i>ToSingle(uint16)</i> | Float | Converts the value of the specified 16-bit unsigned integer to an equivalent single-precision floating-point number. |
| <i>ToSingle(uint32)</i> | Float | Converts the value of the specified 32-bit unsigned integer to an equivalent single-precision floating-point number. |
| <i>ToSingle(uint64)</i> | Float | Converts the value of the specified 64-bit unsigned integer to an equivalent single-precision floating-point number. |
| <i>ToSingle(float)</i> | Float | Returns the specified single-precision floating-point number; no actual conversion is performed. |
| <i>ToSingle(double)</i> | Float | Converts the value of the specified double-precision floating-point number to an equivalent single-precision floating-point number. |
| <i>ToSingle(string)</i> | Float | Converts the specified string representation of a number to an equivalent single-precision floating-point number. |

Table 37: Convert.ToFloat Method

2.1.11. ToChar

| Method | Return Type | Description |
|------------------------------|-------------|--|
| <i>ToChar(Byte)</i> | Char | Converts the value of the specified 8-bit unsigned integer to its equivalent Unicode character. |
| <i>ToChar(char)</i> | Char | Returns the specified Unicode character value; no actual conversion is performed. |
| <i>ToChar(SByte)</i> | Char | Converts the value of the specified 8-bit signed integer to its equivalent Unicode character. |
| <i>ToChar(int16)</i> | Char | Converts the value of the specified 16-bit signed integer to its equivalent Unicode character. |
| <i>ToChar(int32)</i> | Char | Converts the value of the specified 32-bit signed integer to its equivalent Unicode character. |
| <i>ToChar(int64)</i> | Char | Converts the value of the specified 64-bit signed integer to its equivalent Unicode character. |
| <i>ToChar(uint16)</i> | Char | Converts the value of the specified 16-bit unsigned integer to its equivalent Unicode character. |
| <i>ToChar(uint32)</i> | Char | Converts the value of the specified 32-bit unsigned integer to its equivalent Unicode character. |

| | | |
|------------------------------|------|--|
| <i>ToChar(uint64)</i> | Char | Converts the value of the specified 64-bit unsigned integer to its equivalent Unicode character. |
| <i>ToChar(string)</i> | Char | Converts the first character of a specified string to a Unicode character. |

Table 38: Convert.ToChar Method

2.1.12. ToString

| Method | Return Type | Description |
|-------------------------------------|-------------|--|
| <i>ToString(Byte)</i> | String | Converts the value of the specified 8-bit unsigned integer to its equivalent string representation. |
| <i>ToString(bool)</i> | String | Converts the specified Boolean value to its equivalent string representation. |
| <i>ToString(Byte, int32)</i> | String | Converts the value of an 8-bit unsigned integer to its equivalent string representation in a specified base. |
| <i>ToString(char)</i> | String | Converts the value of the specified Unicode character to its equivalent string representation. |
| <i>ToString(SByte)</i> | String | Converts the value of the specified 8-bit signed integer to its equivalent string representation. |

| | | |
|--------------------------------------|--------|--|
| <i>ToString(int16)</i> | String | Converts the value of the specified 16-bit signed integer to its equivalent string representation. |
| <i>ToString(int16, int32)</i> | String | Converts the value of a 16-bit signed integer to its equivalent string representation in a specified base. |
| <i>ToString(int32)</i> | String | Converts the value of the specified 32-bit signed integer to its equivalent string representation. |
| <i>ToString(int32, int32)</i> | String | Converts the value of a 32-bit signed integer to its equivalent string representation in a specified base. |
| <i>ToString(int64)</i> | String | Converts the value of the specified 64-bit signed integer to its equivalent string representation. |
| <i>ToString(int64, int32)</i> | String | Converts the value of a 64-bit signed integer to its equivalent string representation in a specified base. |
| <i>ToString(uint16)</i> | String | Converts the value of the specified 16-bit unsigned integer to its equivalent string representation. |

| | | |
|----------------------------------|--------|---|
| <i>ToString(uint32)</i> | String | Converts the value of the specified 32-bit unsigned integer to its equivalent string representation. |
| <i>ToString(uint64)</i> | String | Converts the value of the specified 64-bit unsigned integer to its equivalent string representation. |
| <i>ToString(string)</i> | String | Returns the specified string instance; no actual conversion is performed. |
| <i>ToString(DateTime)</i> | String | Converts the value of the specified DateTime to its equivalent string representation. |
| <i>ToString(double)</i> | String | Converts the value of the specified double-precision floating-point number to its equivalent string representation. |
| <i>ToString(float)</i> | String | Converts the value of the specified single-precision floating-point number to its equivalent string representation. |

Table 39: Convert.ToString Method

2.1.13. ToDateTime

| Method | Return Type | Description |
|------------------------------------|-------------|---|
| <i>ToDateTime(DateTime)</i> | DateTime | Returns the specified DateTime object; no actual conversion is performed. |

| | | |
|----------------------------------|----------|---|
| <i>ToDateTime(string)</i> | DateTime | Converts the specified string representation of a date and time to an equivalent date and time value. |
|----------------------------------|----------|---|

Table 40: Convert.ToDateTime Method

2.1.14. ToBoolean

| Method | Return Type | Description |
|--------------------------------|-------------|--|
| <i>ToBoolean(bool)</i> | Boolean | Returns the specified Boolean value; no actual conversion is performed. |
| <i>ToBoolean(Byte)</i> | Boolean | Converts the value of the specified 8-bit unsigned integer to an equivalent Boolean value. |
| <i>ToBoolean(SByte)</i> | Boolean | Converts the value of the specified 8-bit signed integer to an equivalent Boolean value. |
| <i>ToBoolean(int16)</i> | Boolean | Converts the value of the specified 16-bit signed integer to an equivalent Boolean value. |
| <i>ToBoolean(int32)</i> | Boolean | Converts the value of the specified 32-bit signed integer to an equivalent Boolean value. |

| | | |
|---------------------------------|---------|--|
| <i>ToBoolean(int64)</i> | Boolean | Converts the value of the specified 64-bit signed integer to an equivalent Boolean value. |
| <i>ToBoolean(uint16)</i> | Boolean | Converts the value of the specified 16-bit unsigned integer to an equivalent Boolean value. |
| <i>ToBoolean(uint32)</i> | Boolean | Converts the value of the specified 32-bit unsigned integer to an equivalent Boolean value. |
| <i>ToBoolean(uint64)</i> | Boolean | Converts the value of the specified 64-bit unsigned integer to an equivalent Boolean value. |
| <i>ToBoolean(float)</i> | Boolean | Converts the value of the specified single-precision floating-point number to an equivalent Boolean value. |
| <i>ToBoolean(double)</i> | Boolean | Converts the value of the specified double-precision floating-point number to an equivalent Boolean value. |
| <i>ToBoolean(string)</i> | Boolean | Converts the specified string representation of a logical value to its Boolean equivalent. |

Table 41: Convert.ToBoolean Method

2.2. Convert Examples





| <div>     </div> | | | | | |
|--|--------------------|---------------|---|-------------------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute_Double | Regular Expression | | Convert.ToDouble(true) | 1 | 2025-11-10 12:27:59.266 |
| MyAttribute_Boolean | Regular Expression | | Convert.ToBoolean(1) | true | 2025-11-10 12:27:59.267 |
| MyAttribute_String | Regular Expression | | Convert.ToString(531.5) | 531.5 | 2025-11-10 12:27:59.268 |
| MyAttribute_DateTime | Regular Expression | | Convert.ToDateTime("2025-11-11 11:44:53") | 2025-11-11 12:44:53.000 | 2025-11-10 12:27:59.269 |
| MyAttribute_SByte | Regular Expression | | Convert.ToSByte(-33.6) | -34 | 2025-11-10 12:27:59.269 |
| MyAttribute_Byte | Regular Expression | | Convert.ToByte(55.3) | 55 | 2025-11-10 12:27:59.269 |
| MyAttribute_Int16 | Regular Expression | | Convert.ToInt16("-341") | -341 | 2025-11-10 12:27:59.270 |

Figure 10: Convert Methods Example

3. BitConverter Operations

The **BitConverter** category provides a set of methods for converting data between various types and Byte arrays, enabling low-level data manipulation.

3.1. BitConverter Methods

| Method | Return Type | Description |
|--|-------------|---|
| <i>DoubleToInt64Bits(double)</i> | Int64 | Converts the specified double-precision floating point number to a 64-bit signed integer. |
| <i>DoubleToUInt64Bits(double)</i> | UInt64 | Converts the specified double-precision floating point number to a 64-bit unsigned integer. |
| <i>Int32BitsToSingle(int32)</i> | Float | Reinterprets the specified 32-bit integer as a single-precision floating-point value. |

| | | |
|--|--------|---|
| <i>Int64BitsToDouble(int64)</i> | Double | Reinterprets the specified 64-bit signed integer to a double-precision floating point number. |
| <i>SingleToInt32Bits(float)</i> | Int32 | Converts a single-precision floating-point value into an integer. |
| <i>SingleToUInt32Bits(float)</i> | UInt32 | Converts the specified single-precision floating point number to a 32-bit unsigned integer. |
| <i>UInt32BitsToSingle(uint32)</i> | Float | Converts the specified 32-bit unsigned integer to a single-precision floating point number. |

Table 42: BitConverter Methods

3.2. BitConverter Examples





|     | | | | | |
|---|--------------------|---------------|--|---------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | BitConverter.Int32BitsToSingle(21474836) | 3.6675422E-38 | 2025-11-10 14:00:02.756 |
| MyAttribute02 | Regular Expression | | BitConverter.UInt32BitsToSingle(42949672u) | 2.1064857E-37 | 2025-11-10 14:00:02.756 |

Figure 11: BitConverter Methods Example

(!) Note

As mentioned in the previous example for the **BitConverter.UInt32BitsToSingle** method, we used the suffix **"u"** or **"U"**, which denotes an unsigned integer literal.

4. BitOperations Methods

The **BitOperations** category provides a collection of methods for performing bitwise operations and bit-level manipulation on numeric data types.

4.1. BitOperations Methods

4.1.1. IsPow

| Method | Return Type | Description |
|------------------------------|-------------|--|
| <i>IsPow2(int32)</i> | Boolean | Evaluates whether the specified Int32 value is a power of two or not. |
| <i>IsPow2(uint32)</i> | Boolean | Evaluates whether the specified UInt32 value is a power of two or not. |
| <i>IsPow2(int64)</i> | Boolean | Evaluates whether the specified Int64 value is a power of two or not. |
| <i>IsPow2(uint64)</i> | Boolean | Evaluates whether the specified UInt64 value is a power of two or not. |

Table 43: BitOperations.IsPow Method

4.1.2. LeadingZeroCount

| Method | Return Type | Description |
|--|-------------|--|
| <i>LeadingZeroCount(uint32)</i> | Int32 | Counts the number of leading zero bits in an unsigned 32-bit integer mask. |

| | | |
|--|-------|--|
| <i>LeadingZeroCount(uint64)</i> | Int32 | Counts the number of leading zero bits in an unsigned 64-bit integer mask. |
|--|-------|--|

Table 44: BitOperations.LeadingZeroCount Method

4.1.3. Log2

| Method | Return Type | Description |
|----------------------------|-------------|---|
| <i>Log2(uint32)</i> | Int32 | Returns the integer (floor) log of the specified value, base 2. |
| <i>Log2(uint64)</i> | Int32 | Returns the integer (floor) log of the specified value, base 2. |

Table 45: BitOperations.Log2 Method

4.1.4. PopCount

| Method | Return Type | Description |
|--------------------------------|-------------|--|
| <i>PopCount(uint32)</i> | Int32 | Returns the population count (number of bits set) of a mask. |
| <i>PopCount(uint64)</i> | Int32 | Returns the population count (number of bits set) of a mask. |

Table 46: BitOperations.PopCount Method

4.1.5. Rotate

| Method | Return Type | Description |
|--|-------------|--|
| <i>RotateLeft(uint32, int32)</i> | UInt32 | Rotates the specified value left by the specified number of bits. |
| <i>RotateRight(uint32, int32)</i> | UInt32 | Rotates the specified value right by the specified number of bits. |
| <i>RotateLeft(uint64, int32)</i> | UInt64 | Rotates the specified value left by the specified number of bits. |
| <i>RotateRight(uint64, int32)</i> | UInt64 | Rotates the specified value right by the specified number of bits. |

Table 47: BitOperations Rotate Methods

4.1.6. RoundUpToPowerOf2

| Method | Return Type | Description |
|---|-------------|---|
| <i>RoundUpToPowerOf2(uint32)</i> | UInt32 | Rounds the specified UInt32 value up to a power of two. |

Table 48: BitOperations.RoundUpToPowerOf2 Method

4.1.7. TrailingZeroCount

| Method | Return Type | Description |
|--|-------------|---|
| <i>TrailingZeroCount(int64 value)</i> | Int64 | Counts the number of trailing zero bits in a 64-bit integer value mask. |
| <i>TrailingZeroCount(uint32)</i> | UInt32 | Counts the number of trailing zero bits in an unsigned 32-bit integer value mask. |
| <i>TrailingZeroCount(uint64)</i> | UInt64 | Counts the number of trailing zero bits in an unsigned 64-bit integer value mask. |

Table 49: BitOperations.TrailingZeroCount Method

4.2. BitOperations Examples



|     | | | | | |
|---|--------------------|---------------|--|-------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | BitOperations.IsPow2(8) | true | 2025-11-10 14:08:36.777 |
| MyAttribute02 | Regular Expression | | BitOperations.LeadingZeroCount(4294967295) | 0 | 2025-11-10 14:08:36.778 |
| MyAttribute03 | Regular Expression | | BitOperations.Log2(4294967295) | 31 | 2025-11-10 14:08:36.778 |

Figure 12: BitOperations Methods Example

5. Boolean Properties, Methods and Operators

The **Boolean** category provides a set of properties and methods for managing, manipulating, and formatting Boolean values.

5.1. Boolean Properties

| Property | Type | Description |
|---------------------------|--------|---|
| <i>TrueString</i> | string | Represents the Boolean value true as a string. |
| <i>FalseString</i> | string | Represents the Boolean value false as a string. |

Table 50: Boolean Properties

5.2. Boolean Methods

| Method | Return Type | Description |
|-------------------------------|-------------|---|
| <i>Parse(string)</i> | Boolean | Converts the specified string representation of a logical value to its Boolean equivalent. |
| <i>CompareTo(bool)</i> | Int32 | Compares this instance to a specified Boolean object and returns an integer that indicates their relationship to one another. |
| <i>Equals(bool)</i> | Boolean | Returns a value indicating whether this instance is equal to a specified object or not. |

| | | |
|--------------------------|--------|---|
| <i>ToString()</i> | String | Converts the value of this instance to its equivalent string representation (either "True" or "False"). |
|--------------------------|--------|---|

Table 51: Boolean Methods

5.3. Boolean Operators

| Operator | Return Type | Description |
|---------------------------------------|-------------|---|
| <i>Equality (==)</i> | Boolean | The equality operator = returns true if its operands are equal, false otherwise. |
| <i>Inequality (!=)</i> | Boolean | The inequality operator <> returns true if its operands aren't equal, false otherwise. |
| <i>LogicalOR ()</i> | Boolean | The conditional logical OR operator , computes the logical OR of its operands. The result of x y is true if either x or y evaluates to true. Otherwise, the result is false. If x evaluates to true, y isn't evaluated. |
| <i>LogicalAND (&&)</i> | Boolean | The conditional logical AND operator && computes the logical AND of its operands. The result of x && y is true if both x and y are evaluated to true. Otherwise, the result is false. If x evaluates to false, y isn't evaluated. |

| | | |
|----------------------------------|---------|--|
| <i>LogicalNegation(!)</i> | Boolean | The unary prefix ! operator computes logical negation of its operand. That is, it produces true, if the operand evaluates to false, and false, if the operand evaluates to true. |
|----------------------------------|---------|--|

Table 52: Boolean Operators

5.4. Boolean Examples

General

Attributes

| Property | Address | Default Value | Regular Expression | Value | Timestamp |
|---------------|--------------------|---------------|---------------------------------|-------|-------------------------|
| MyAttribute01 | Regular Expression | | Boolean.TrueString | True | 2025-11-10 14:20:59.434 |
| MyAttribute02 | Regular Expression | | Boolean.FalseString | False | 2025-11-10 14:20:59.435 |
| MyAttribute03 | Regular Expression | | Boolean.Parse("False") | false | 2025-11-10 14:20:59.436 |
| MyAttribute04 | Regular Expression | | (\$MyAttribute03).Equals(false) | true | 2025-11-10 14:20:59.436 |
| MyAttribute05 | Regular Expression | | \$MyAttribute04 && True | true | 2025-11-10 14:20:59.436 |

Figure 13: Boolean Operations Example

6. Char Properties, Methods and Operators

The **Char** category provides a collection of properties and methods for managing, manipulating, and formatting Unicode characters.

6.1. Char Properties

| Property | Type | Description |
|------------------------|------|--|
| <i>MaxValue</i> | Char | Represents the largest possible value of a Char. |

| | | |
|------------------------|------|---|
| <i>MinValue</i> | Char | Represents the smallest possible value of a Char. |
|------------------------|------|---|

Table 53: Char Properties

6.2. Char Methods

| Method | Type | Description |
|--|--------|---|
| <i>CompareTo(char)</i> | Int32 | Compares this instance to a specified Char object and indicates whether this instance precedes, follows, or appears in the same position in the sort order as the specified Char object or not. |
| <i>ConvertFromUtf32(int)</i> | String | Converts the specified Unicode code point into a UTF-16 encoded string. |
| <i>ConvertToUtf32(string, int)</i> | Int32 | Converts the value of a UTF-16 encoded character or surrogate pair at a specified position in a string into a Unicode code point. |
| <i>ConvertToUtf32(char, char)</i> | Int32 | Converts the value of a UTF-16 encoded surrogate pair into a Unicode code point. |
| <i>GetNumericValue(char)</i> | Double | Converts the specified numeric Unicode character to a double-precision floating point number. |
| <i>GetNumericValue(string, int)</i> | Double | Converts the numeric Unicode character at the specified position in a specified |

| | | |
|---|---------|--|
| | | string to a double-precision floating point number. |
| <i>GetUnicodeCategory(char)</i> | String | Categorizes a specified Unicode character into a group identified by one of the Unicode Category values. |
| <i>GetUnicodeCategory(string, int)</i> | String | Categorizes the character at the specified position in a specified string into a group identified by one of the Unicode Category values. |
| <i>IsAscii(char)</i> | Boolean | Returns true if c is an ASCII character ([U+0000..U+007F]). |
| <i>IsControl(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a control character or not. |
| <i>IsControl(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a control character or not. |
| <i>IsDigit(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a decimal digit or not. |
| <i>IsDigit(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a decimal digit or not. |

| | | |
|--|---------|--|
| <i>IsHighSurrogate(char)</i> | Boolean | Indicates whether the specified Char object is a high surrogate or not. |
| <i>IsHighSurrogate(string, int)</i> | Boolean | Indicates whether the Char object at the specified position in a string is a high surrogate or not. |
| <i>IsLetter(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a Unicode letter or not. |
| <i>IsLetter(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a Unicode letter or not. |
| <i>IsLetterOrDigit(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a letter or a decimal digit or not. |
| <i>IsLetterOrDigit(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a letter or a decimal digit. |
| <i>IsLower(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a lowercase letter or not. |
| <i>IsLower(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a lowercase letter or not. |

| | | |
|---|---------|--|
| <i>IsLowSurrogate(char)</i> | Boolean | Indicates whether the specified Char object is a low surrogate or not. |
| <i>IsLowSurrogate(string, int)</i> | Boolean | Indicates whether the Char object at the specified position in a string is a low surrogate or not. |
| <i>IsNumber(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a number or not. |
| <i>IsNumber(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a number or not. |
| <i>IsPunctuation(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a punctuation mark or not. |
| <i>IsPunctuation(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a punctuation mark or not. |
| <i>IsSeparator(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a separator character or not. |
| <i>IsSeparator(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is |

| | | |
|--|---------|--|
| | | categorized as a separator character or not. |
| <i>IsSurrogate(char)</i> | Boolean | Indicates whether the specified character has a surrogate code unit or not. |
| <i>IsSurrogate(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string has a surrogate code unit or not. |
| <i>IsSurrogatePair(string, int)</i> | Boolean | Indicates whether two adjacent Char objects are at a specified position in a string form a surrogate pair or not. |
| <i>IsSurrogatePair(char, char)</i> | Boolean | Indicates whether the two specified Char objects form a surrogate pair or not. |
| <i>IsSymbol(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as a symbol character or not. |
| <i>IsSymbol(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as a symbol character or not. |
| <i>IsUpper(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as an uppercase letter or not. |

| | | |
|---|---------|---|
| <i>IsUpper(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as an uppercase letter or not. |
| <i>IsWhiteSpace(char)</i> | Boolean | Indicates whether the specified Unicode character is categorized as white space or not. |
| <i>IsWhiteSpace(string, int)</i> | Boolean | Indicates whether the character at the specified position in a specified string is categorized as white space or not. |
| <i>Parse(string)</i> | Char | Converts the value of the specified string to its equivalent Unicode character. |
| <i>ToLower(char)</i> | Char | Converts the value of a Unicode character to its lowercase equivalent. |
| <i>ToLowerInvariant(char)</i> | Char | Converts the value of a Unicode character to its lowercase equivalent using the casing rules of the invariant culture. |
| <i>ToString(char)</i> | Char | Converts the specified Unicode character to its equivalent string representation. |
| <i>ToUpper(char)</i> | Char | Converts the value of a Unicode character to its uppercase equivalent. |

| | | |
|--------------------------------------|--------|--|
| <i>ToUpperInvariant(char)</i> | Char | Converts the value of a Unicode character to its uppercase equivalent using the casing rules of the invariant culture. |
| <i>ToString()</i> | String | Converts the value of this instance to its equivalent string representation. |

Table 54: Char Methods

6.3. Char Operators

| Operator | Return Type | Description |
|----------------------------------|---|---|
| <i>Addition (+)</i> | Numeric, same or promoted numeric type of operands (e.g., int, double, float) | The addition operator + computes the sum of its operands. |
| <i>Subtraction (-)</i> | Numeric, same or promoted numeric type of operands (e.g., int, double, float) | The subtraction operator - subtracts its right-hand operand from its left-hand operand. |
| <i>Multiplication (*)</i> | Numeric, same or promoted numeric type of operands (e.g., int, double, float) | The multiplication operator * computes the product of its operands. |
| <i>Division (/)</i> | Numeric, same or promoted numeric type of operands (e.g., int, double, float) | The division operator / divides its left-hand operand by its right-hand operand. |

| | | |
|--|---|--|
| <i>Remainder (%)</i> | Numeric, same or promoted numeric type of operands (e.g., int, double, float) | The remainder operator % computes the remainder after dividing its left-hand operand by its right-hand operand. |
| <i>Equality (==)</i> | Boolean | The equality operator == returns true if its operands are equal, false otherwise. |
| <i>Inequality (!=)</i> | Boolean | The inequality operator <> returns true if its operands aren't equal, false otherwise. |
| <i>GreaterThan (>)</i> | Boolean | The > operator returns true if its left-hand operand is greater than its right-hand operand, false otherwise. |
| <i>GreaterThanOrEqual (>=)</i> | Boolean | The >= operator returns true if its left-hand operand is greater than or equal to its right-hand operand, false otherwise. |
| <i>LessThan (<)</i> | Boolean | The < operator returns true if its left-hand operand is |

| | | |
|---------------------------------------|--|--|
| | | less than its right-hand operand, false otherwise. |
| <i>LessThanOrEqual (<=)</i> | Boolean | The <= operator returns true if its left-hand operand is less than or equal to its right-hand operand, false otherwise. |
| <i>BitwiseAND (&)</i> | Integral numeric type, same or promoted type of operands | The & operator computes the bitwise logical AND of its integral operands. |
| <i>BitwiseOR ()</i> | Integral numeric type, same or promoted type of operands | The operator computes the bitwise logical OR of its integral operands. |
| <i>BitwiseXOR (^)</i> | Integral numeric type, same or promoted type of operands | The ^ operator computes the bitwise logical exclusive OR, also known as the bitwise logical XOR, of its integral operands. |
| <i>BitwiseComplement (~)</i> | Integral numeric type, same or promoted type of operands | The ~ operator produces a bitwise complement of its operand by reversing each bit. |

Table 55: Char Operators

6.4. Char Examples





|     | | | | | |
|---|--------------------|---------------|------------------------------|-----------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | Char.ConvertFromUtf32(253) | ý | 2025-11-10 14:45:05.947 |
| MyAttribute02 | Regular Expression | | Char.GetUnicodeCategory('A') | UppercaseLetter | 2025-11-10 14:45:05.948 |
| MyAttribute03 | Regular Expression | | Char.GetNumericValue('A') | -1 | 2025-11-10 14:45:05.949 |
| MyAttribute04 | Regular Expression | | Char.IsDigit('B') | False | 2025-11-10 14:45:05.950 |

Figure 14: Char Operations Example

7. Byte Properties, Methods and Operators

The **Byte** category provides a collection of properties and methods for managing, manipulating, and formatting 8-bit unsigned integer values.

7.1. Byte Properties

| Property | Type | Description |
|------------------------|------|---|
| <i>MaxValue</i> | Byte | Represents the largest possible value of Byte. |
| <i>MinValue</i> | Byte | Represents the smallest possible value of Byte. |

Table 56: Byte Properties

7.2. Byte Methods

| Method | Return Type | Description |
|--------------------------------|-------------|---|
| <i>CompareTo(Byte)</i> | Int32 | Compares this instance to a specified 8-bit unsigned integer and returns an indication of their relative values. |
| <i>Equals(Byte)</i> | Boolean | Returns a value indicating whether this instance and a specified 8-bit unsigned integer object represent the same value or not. |
| <i>Parse(string)</i> | Byte | Converts the string representation of a number to its 8-bit unsigned integer equivalent. |
| <i>ToString()</i> | String | Converts the value of the current 8-bit unsigned integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 8-bit unsigned integer object to its equivalent string representation using the specified format. |

Table 57: Byte Methods

7.3. Byte Operators

| Operator | Return Type | Description |
|----------|-------------|-------------|
|----------|-------------|-------------|

| | | |
|----------------------------------|---------|---|
| <i>Increment (++)</i> | Byte | Increases the value of a Byte by 1. |
| <i>Decrement (--)</i> | Byte | Decreases the value of a Byte by 1. |
| <i>Addition (+)</i> | Int32 | The addition operator + computes the sum of its operands. |
| <i>Subtraction (-)</i> | Int32 | The subtraction operator - subtracts its right-hand operand from its left-hand operand. |
| <i>Multiplication (*)</i> | Int32 | The multiplication operator * computes the product of its operands. |
| <i>Division (/)</i> | Int32 | The division operator / divides its left-hand operand by its right-hand operand. |
| <i>Remainder (%)</i> | Int32 | The remainder operator % computes the remainder after dividing its left-hand operand by its right-hand operand. |
| <i>Equality (==)</i> | Boolean | The equality operator == returns true if its operands are equal, false otherwise. |
| <i>Inequality (!=)</i> | Boolean | The inequality operator <> returns true if its operands aren't equal, false otherwise. |
| <i>GreaterThan (>)</i> | Boolean | The > operator returns true if its left-hand operand is greater than its right-hand operand, false otherwise. |

| | | |
|--|---------|--|
| <i>GreaterThanEqual (>=)</i> | Boolean | The >= operator returns true if its left-hand operand is greater than or equal to its right-hand operand, false otherwise. |
| <i>LessThan (<)</i> | Boolean | The < operator returns true if its left-hand operand is less than its right-hand operand, false otherwise. |
| <i>LessThanOrEqual (<=)</i> | Boolean | The <= operator returns true if its left-hand operand is less than or equal to its right-hand operand, false otherwise. |
| <i>LeftShift (<<)</i> | Int32 | The << operator shifts its left-hand operand left by the number of bits defined by its right-hand operand. The left-shift operation discards the high-order bits that are outside the range of the result type and sets the low-order empty bit positions to zero. |
| <i>RightShift (>>)</i> | Int32 | The >> operator shifts its left-hand operand right by the number of bits defined by its right-hand operand. The right-shift operation discards the low-order bits. |
| <i>BitwiseAND (&)</i> | Int32 | The & operator computes the bitwise logical AND of its integral operands. |

| | | |
|-------------------------------------|-------|--|
| <i>BitwiseOR ()</i> | Int32 | The operator computes the bitwise logical OR of its integral operands. |
| <i>BitwiseXOR (^)</i> | Int32 | The ^ operator computes the bitwise logical exclusive OR, also known as the bitwise logical XOR, of its integral operands. |
| <i>BitwiseComplement (~)</i> | Int32 | The ~ operator produces a bitwise complement of its operand by reversing each bit. |

Table 58: Byte Operators
(!) Note

```
Byte a = 0b_0000_1101; // Decimal 13
```

```
Byte b = 0b_0000_0110; // Decimal 6
```

```
int resultAnd = a & b; // resultAnd will be 4 (0b_0000_0100)
```

```
int resultOr = a | b; // resultOr will be 15 (0b_0000_1111)
```

```
int resultXor = a ^ b; // resultXor will be 11 (0b_0000_1011)
```

```
int resultNot = ~a; // resultNot will be -14 (due to int promotion and two's complement)
```

```
Byte shiftedLeft = (Byte)(a << 2); // shiftedLeft will be 52 (0b_0011_0100)
```

```
Byte shiftedRight = (Byte)(a >> 2); // shiftedRight will be 3 (0b_0000_0011)
```

7.4. Byte Examples





|     | | | | | |
|---|--------------------|---------------|----------------------------------|-------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | Byte.Parse("255") | 255 | 2025-11-10 14:53:43.461 |
| MyAttribute02 | Regular Expression | | Byte.MinValue | 0 | 2025-11-10 14:53:43.462 |
| MyAttribute03 | Regular Expression | | Byte.MaxValue | 255 | 2025-11-10 14:53:43.462 |
| MyAttribute04 | Regular Expression | | (\$MyAttribute01).ToString("D4") | 0255 | 2025-11-10 14:53:43.461 |
| MyAttribute05 | Regular Expression | | (\$MyAttribute01).Equals(256) | false | 2025-11-10 14:53:43.461 |

Figure 15: Byte Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000."

8. SByte Properties, Methods and Operators

The **SByte** category provides a collection of properties and methods for managing, manipulating, and formatting 8-bit signed integer values.

8.1. SByte Properties

| Property | Type | Description |
|------------------------|-------|--|
| <i>MaxValue</i> | SByte | Represents the largest possible value of SByte. |
| <i>MinValue</i> | SByte | Represents the smallest possible value of SByte. |

Table 59: SByte Properties

8.2. SByte Methods

| Method | Return Type | Description |
|---------------------------------------|-------------|---|
| <i>CompareTo(Byte)</i> | Int32 | Compares this instance to a specified 8-bit signed integer and returns an indication of their relative values. |
| <i>Equals(SByte)</i> | Boolean | Returns a value indicating whether this instance and a specified 8-bit signed integer object represent the same value or not. |
| <i>Parse(string)</i> | SByte | Converts the string representation of a number to its 8-bit signed integer equivalent. |
| <i>ToString()</i> | String | Converts the value of the current 8-bit signed integer object to its equivalent string representation. |
| <i>ToString(string format)</i> | String | Converts the value of the current 8-bit signed integer object to its equivalent string representation using the specified format. |

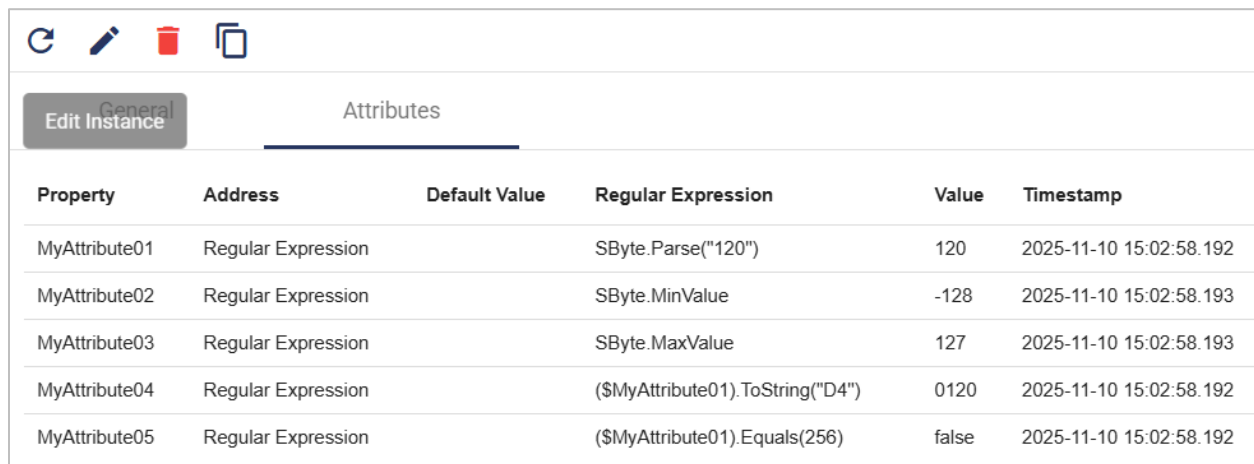
Table 60: SByte Methods

8.3. SByte Operators

(!) Note

The **SByte** category uses the same operators defined for the **Byte** type. However, the **Increment** (++) and **Decrement** (--) operators return an **SByte** value.

8.4. SByte Examples



| Property | Address | Default Value | Regular Expression | Value | Timestamp |
|---------------|--------------------|---------------|----------------------------------|-------|-------------------------|
| MyAttribute01 | Regular Expression | | SByte.Parse("120") | 120 | 2025-11-10 15:02:58.192 |
| MyAttribute02 | Regular Expression | | SByte.MinValue | -128 | 2025-11-10 15:02:58.193 |
| MyAttribute03 | Regular Expression | | SByte.MaxValue | 127 | 2025-11-10 15:02:58.193 |
| MyAttribute04 | Regular Expression | | (\$MyAttribute01).ToString("D4") | 0120 | 2025-11-10 15:02:58.192 |
| MyAttribute05 | Regular Expression | | (\$MyAttribute01).Equals(256) | false | 2025-11-10 15:02:58.192 |

Figure 16: SByte Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000."

9. Int16 Properties, Methods and Operators

The **Int16** category provides a collection of properties and methods for managing, manipulating, and formatting 16-bit signed integer values.

9.1. Int16 Properties

| Property | Type | Description |
|------------------------|-------|--|
| <i>MaxValue</i> | Int16 | Represents the largest possible value of Int16. |
| <i>MinValue</i> | Int16 | Represents the smallest possible value of Int16. |

Table 61: Int16 Properties

9.2. Int16 Methods

| Method | Return Type | Description |
|--------------------------------|-------------|--|
| <i>CompareTo(int16)</i> | Int32 | Compares this instance to a specified 16-bit signed integer and returns an indication of their relative values. |
| <i>Equals(int16)</i> | Boolean | Returns a value indicating whether this instance and a specified 16-bit signed integer object represent the same value or not. |
| <i>Parse(string)</i> | Int16 | Converts the string representation of a number to its 16-bit signed integer equivalent. |

| | | |
|--------------------------------|--------|--|
| <i>ToString()</i> | String | Converts the value of the current 16-bit signed integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 16-bit signed integer object to its equivalent string representation using the specified format. |

Table 62: Int16 Methods

9.3. Int16 Operators

| Operator | Return Type | Description |
|----------------------------------|-------------|---|
| <i>Addition (+)</i> | Int32 | The addition operator + computes the sum of its operands. |
| <i>Subtraction (-)</i> | Int32 | The subtraction operator - subtracts its right-hand operand from its left-hand operand. |
| <i>Multiplication (*)</i> | Int32 | The multiplication operator * computes the product of its operands. |
| <i>Division (/)</i> | Int32 | The division operator / divides its left-hand operand by its right-hand operand. |

| | | |
|--|---------|--|
| <i>Remainder (%)</i> | Int32 | The remainder operator % computes the remainder after dividing its left-hand operand by its right-hand operand. |
| <i>Equality (==)</i> | Boolean | Returns true if the two Int16 operands are equal; otherwise, false. |
| <i>Inequality (!=)</i> | Boolean | Returns true if the two Int16 operands are not equal; otherwise, false. |
| <i>GreaterThan (>)</i> | Boolean | Returns true if the first Int16 operand is greater than the second; otherwise, false. |
| <i>GreaterThanOrEqual (>=)</i> | Boolean | Returns true if the first Int16 operand is greater than or equal to the second; otherwise, false. |
| <i>LessThan (<)</i> | Boolean | Returns true if the first Int16 operand is less than the second; otherwise, false. |
| <i>LessThanOrEqual (<=)</i> | Boolean | Returns true if the first Int16 operand is less than or equal to the second; otherwise, false. |
| <i>LeftShift (<<)</i> | Int32 | Shifts all bits in the Int16 value to the left by the specified number of bits; zero bits are inserted on the right. |

| | | |
|-------------------------------------|-------|---|
| <i>RightShift (>>)</i> | Int32 | Shifts all bits in Int16 value to the right by the specified number of bits; the sign bit is preserved. |
| <i>BitwiseAND (&)</i> | Int32 | Performs a bitwise AND operation on each bit pair of the Int16 operands. |
| <i>BitwiseOR ()</i> | Int32 | Performs a bitwise OR operation on each bit pair of the Int16 operands. |
| <i>BitwiseXOR (^)</i> | Int32 | Performs a bitwise exclusive OR (XOR) operation on each bit pair of the Int16 operands. |
| <i>BitwiseComplement (~)</i> | Int32 | Inverts all bits of the Int16 operand (bitwise NOT). |

Table 63: Int16 Operators

9.4. Int16 Examples





| <div>     </div> | | | | | |
|--|--------------------|---------------|--|----------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | Int16.MinValue | -32768 | 2025-11-10 15:26:16.565 |
| MyAttribute02 | Regular Expression | | Int16.MaxValue | 32767 | 2025-11-10 15:26:16.565 |
| MyAttribute03 | Regular Expression | | Int16.Parse("556") | 556 | 2025-11-10 15:26:16.566 |
| MyAttribute04 | Regular Expression | | (\$MyAttribute01).ToString("C3") | (\$32,768.000) | 2025-11-10 15:26:16.565 |
| MyAttribute05 | Regular Expression | | \$MyAttribute01 + \$MyAttribute02 | -1 | 2025-11-10 15:26:16.565 |
| MyAttribute06 | Regular Expression | | \$MyAttribute01 - \$MyAttribute02 | -65535 | 2025-11-10 15:26:16.565 |
| MyAttribute07 | Regular Expression | | (\$MyAttribute01).Equals(Int16.MinValue) | true | 2025-11-10 15:26:16.565 |
| MyAttribute08 | Regular Expression | | \$MyAttribute03 << 2 | 2224 | 2025-11-10 15:26:16.566 |
| MyAttribute09 | Regular Expression | | \$MyAttribute03 >> 2 | 139 | 2025-11-10 15:26:16.566 |

Figure 17: Int16 Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000."

10. Int32 Properties, Methods and Operators

The **Int32** category provides a collection of properties and methods for managing, manipulating, and formatting 32-bit signed integer values.

10.1. Int32 Properties

| Property | Type | Description |
|------------------------|-------|---|
| <i>MaxValue</i> | Int32 | Represents the largest possible value of Int32. |

| | | |
|------------------------|-------|--|
| <i>MinValue</i> | Int32 | Represents the smallest possible value of Int32. |
|------------------------|-------|--|

Table 64: Int32 Properties

10.2. Int32 Methods

| Method | Type | Description |
|--------------------------------|---------|--|
| <i>CompareTo(int32)</i> | Int32 | Compares this instance to a specified 32-bit signed integer and returns an indication of their relative values. |
| <i>Equals(int32)</i> | Boolean | Returns a value indicating whether this instance and a specified 32-bit signed integer object represent the same value or not. |
| <i>Parse(string)</i> | Int32 | Converts the string representation of a number to its 32-bit signed integer equivalent. |
| <i>ToString()</i> | String | Converts the value of the current 32-bit signed integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 32-bit signed integer object to its equivalent string representation using the specified format. |

Table 65: Int32 Methods

10.3. Int32 Operators

(!) Note

The **Int32** category uses the same operators defined for the **Int16** type. However, The return type for the operators is **Int32** for operators such as addition, subtraction, multiplication, division, or modulo.

10.4. Int32 Examples





| <div>     </div> | | | | | |
|--|--------------------|-----------------------------------|--------------------|-------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | Int32.MinValue | | -2147483648 | 2025-11-10 15:56:19.856 |
| MyAttribute02 | Regular Expression | Int32.MaxValue | | 2147483647 | 2025-11-10 15:56:19.856 |
| MyAttribute03 | Regular Expression | Int32.Parse("500") | | 500 | 2025-11-10 15:56:19.857 |
| MyAttribute04 | Regular Expression | (\$MyAttribute01).ToString("X4") | | 80000000 | 2025-11-10 15:56:19.856 |
| MyAttribute05 | Regular Expression | 300 * 5 | | 1500 | 2025-11-10 15:56:19.858 |
| MyAttribute06 | Regular Expression | 500 / 2 | | 250 | 2025-11-10 15:56:19.858 |
| MyAttribute07 | Regular Expression | \$MyAttribute01 == Int32.MinValue | | true | 2025-11-10 15:56:19.856 |
| MyAttribute08 | Regular Expression | \$MyAttribute03 & 2 | | 0 | 2025-11-10 15:56:19.857 |
| MyAttribute09 | Regular Expression | \$MyAttribute03 2 | | 502 | 2025-11-10 15:56:19.857 |

Figure 18: Int32 Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000.

11. Int64 Properties, Methods and Operators

The **Int64** category provides a collection of properties and methods for managing, manipulating, and formatting 64-bit signed integer values.

11.1. Int64 Properties

| Property | Type | Description |
|------------------------|-------|--|
| <i>MaxValue</i> | Int64 | Represents the largest possible value of Int64. |
| <i>MinValue</i> | Int64 | Represents the smallest possible value of Int64. |

Table 66: Int64 Properties

11.2. Int64 Methods

| Method | Return Type | Description |
|--------------------------------|-------------|--|
| <i>Equals(int64)</i> | Boolean | Returns a value indicating whether this instance and a specified 64-bit signed integer object represent the same value or not. |
| <i>CompareTo(int64)</i> | Int32 | Compares this instance to a specified 64-bit signed integer and returns an indication of their relative values. |
| <i>Parse(string)</i> | Int64 | Converts the string representation of a number to its 64-bit signed integer equivalent. |

| | | |
|--------------------------------|--------|--|
| <i>ToString()</i> | String | Converts the value of the current 64-bit signed integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 64-bit signed integer object to its equivalent string representation using the specified format. |

Table 67: Int64 Methods

11.3. Int64 Operators

(!) Note

The **Int64** category uses the same operators defined for the **Int16** type. However, The return type for the operators is **Int64** for operators such as addition, subtraction, multiplication, division, or modulo.

11.4. Int64 Examples





|     | | | | | |
|---|--------------------|-----------------------------------|--------------------|----------------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | Int64.MinValue | | -9223372036854775808 | 2025-11-10 16:10:42.628 |
| MyAttribute02 | Regular Expression | Int64.MaxValue | | 9223372036854775807 | 2025-11-10 16:10:42.629 |
| MyAttribute03 | Regular Expression | Int64.Parse("500") | | 500 | 2025-11-10 16:10:42.629 |
| MyAttribute04 | Regular Expression | (\$MyAttribute01).ToString("X4") | | 8000000000000000 | 2025-11-10 16:10:42.628 |
| MyAttribute05 | Regular Expression | 300 * 5 | | 1500 | 2025-11-10 16:10:42.630 |
| MyAttribute06 | Regular Expression | 500 - 2 | | 498 | 2025-11-10 16:10:42.630 |
| MyAttribute07 | Regular Expression | \$MyAttribute01 != Int64.MinValue | | false | 2025-11-10 16:10:42.628 |
| MyAttribute08 | Regular Expression | \$MyAttribute03 << 2 | | 2000 | 2025-11-10 16:10:42.629 |
| MyAttribute09 | Regular Expression | \$MyAttribute03 >> 2 | | 125 | 2025-11-10 16:10:42.629 |

Figure 19: Int64 Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000.

12. UInt16 Properties, Methods and Operators

The **UInt16** category provides a collection of properties and methods for managing, manipulating, and formatting 16-bit unsigned integer values.

12.1. UInt16 Properties

| Property | Type | Description |
|-----------------|--------|---|
| MaxValue | UInt16 | Represents the largest possible value of UInt16. |
| MinValue | UInt16 | Represents the smallest possible value of UInt16. |

Table 68: UInt16 Properties

12.2. UInt16 Methods

| Method | Return Type | Description |
|--------------------------|-------------|---|
| CompareTo(uint16) | Int32 | Compares this instance to a specified 16-bit unsigned integer and returns an indication of their relative values. |
| Equals(uint16) | Boolean | Returns a value indicating whether this instance and a specified 16-bit unsigned |

| | | |
|--------------------------------|--------|--|
| | | integer object represent the same value or not. |
| <i>Parse(string)</i> | UInt16 | Converts the string representation of a number to its 16-bit unsigned integer equivalent. |
| <i>ToString()</i> | String | Converts the value of the current 16-bit unsigned integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 16-bit unsigned integer object to its equivalent string representation using the specified format. |

Table 69: UInt16 Methods

12.3. UInt16 Operators

(!) Note

The **UInt16** category uses the same operators defined for the **Int16** type. However, The return type for the operators is **Int32** for operators such as addition, subtraction, multiplication, division, or modulo.

12.4. UInt16 Examples





| <div>     </div> | | | | | |
|--|--------------------|------------------------------------|--------------------|-------|-------------------------|
| <div> <div>sh Instance</div> <div>General</div> <div>Attributes</div> </div> | | | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | UInt16.MinValue | | 0 | 2025-11-10 16:17:41.583 |
| MyAttribute02 | Regular Expression | UInt16.MaxValue | | 65535 | 2025-11-10 16:17:41.584 |
| MyAttribute03 | Regular Expression | UInt16.Parse("500") | | 500 | 2025-11-10 16:17:41.585 |
| MyAttribute04 | Regular Expression | (\$MyAttribute01).ToString("X4") | | 0000 | 2025-11-10 16:17:41.583 |
| MyAttribute05 | Regular Expression | 300 * 5 | | 1500 | 2025-11-10 16:17:41.587 |
| MyAttribute06 | Regular Expression | 500 - 2 | | 498 | 2025-11-10 16:17:41.587 |
| MyAttribute07 | Regular Expression | \$MyAttribute01 != UInt16.MinValue | | false | 2025-11-10 16:17:41.583 |
| MyAttribute08 | Regular Expression | \$MyAttribute03 << 2 | | 2000 | 2025-11-10 16:17:41.585 |
| MyAttribute09 | Regular Expression | \$MyAttribute03 >> 2 | | 125 | 2025-11-10 16:17:41.585 |

Figure 20: UInt16 Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000."

13. UInt32 Properties, Methods and Operators

The **UInt32** category provides a collection of properties and methods for managing, manipulating, and formatting 32-bit unsigned integer values.

13.1. UInt32 Properties

| Property | Type | Description |
|------------------------|--------|--|
| <i>MaxValue</i> | UInt32 | Represents the largest possible value of UInt32. |

| | | |
|------------------------|--------|---|
| <i>MinValue</i> | UInt32 | Represents the smallest possible value of UInt32. |
|------------------------|--------|---|

Table 70: UInt32 Properties

13.2. UInt32 Methods

| Method | Return Type | Description |
|---------------------------------|-------------|--|
| <i>CompareTo(uint32)</i> | Int32 | Compares this instance to a specified 32-bit unsigned integer and returns an indication of their relative values. |
| <i>Equals(uint32)</i> | Boolean | Returns a value indicating whether this instance and a specified 32-bit unsigned integer object represent the same value or not. |
| <i>Parse(string)</i> | UInt32 | Converts the string representation of a number to its 32-bit unsigned integer equivalent. |
| <i>ToString()</i> | String | Converts the value of the current 32-bit unsigned integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 32-bit unsigned integer object to its equivalent string representation using the specified format. |

Table 71: UInt32 Methods

13.3. UInt32 Operators

(!) Note

The **UInt32** category uses the same operators defined for the **Int16** type. However, The return type for the operators is **UInt32** for operators such as addition, subtraction, multiplication, division, or modulo.

13.4. UInt32 Examples





| <div>     </div> | | | | | |
|--|--------------------|-----------------------------------|--------------------|------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | UInt32.MinValue | | 0 | 2025-11-10 16:24:48.898 |
| MyAttribute02 | Regular Expression | UInt32.MaxValue | | 4294967295 | 2025-11-10 16:24:48.899 |
| MyAttribute03 | Regular Expression | UInt32.Parse("350") | | 350 | 2025-11-10 16:24:48.900 |
| MyAttribute04 | Regular Expression | (\$MyAttribute01).ToString("D4") | | 0000 | 2025-11-10 16:24:48.898 |
| MyAttribute05 | Regular Expression | 786489 * 5 | | 3932445 | 2025-11-10 16:24:48.901 |
| MyAttribute06 | Regular Expression | 548963 - 7845 | | 541118 | 2025-11-10 16:24:48.902 |
| MyAttribute07 | Regular Expression | \$MyAttribute01 > UInt32.MinValue | | false | 2025-11-10 16:24:48.898 |
| MyAttribute08 | Regular Expression | \$MyAttribute03 << 2 | | 1400 | 2025-11-10 16:24:48.900 |
| MyAttribute09 | Regular Expression | \$MyAttribute03 >> 2 | | 87 | 2025-11-10 16:24:48.900 |

Figure 21: UInt32 Operations Example

(!) Note

As shown in the figure above, the format specifiers for the ToString(string format) method are: C3, D4, E1, E2, F1, G, N1, P0, X4, 0000.0000."

14. UInt64 Properties, Methods and Operators

The **UInt64** category provides a collection of properties and methods for managing, manipulating, and formatting 64-bit unsigned integer values.

14.1. UInt64 Properties

| Property | Type | Description |
|------------------------|--------|---|
| <i>MaxValue</i> | UInt64 | Represents the largest possible value of UInt64. |
| <i>MinValue</i> | UInt64 | Represents the smallest possible value of UInt64. |

Table 72: UInt64 Properties

14.2. UInt64 Methods

| Method | Return Type | Description |
|---------------------------------|-------------|--|
| <i>CompareTo(uint64)</i> | Int32 | Compares this instance to a specified 64-bit unsigned integer and returns an indication of their relative values. |
| <i>Equals(uint64)</i> | Boolean | Returns a value indicating whether this instance and a specified 64-bit unsigned integer object represent the same value or not. |

| | | |
|--------------------------------|--------|--|
| <i>Parse(string)</i> | UInt64 | Converts the string representation of a number to its 64-bit unsigned integer equivalent. |
| <i>ToString()</i> | String | Converts the value of the current 64-bit unsigned integer object to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the value of the current 64-bit unsigned integer object to its equivalent string representation using the specified format. |

Table 73: UInt64 Methods

14.3. UInt64 Operators

(!) Note

The **UInt64** category uses the same operators defined for the **Int16** type. However, The return type for the operators is **UInt64** for operators such as addition, subtraction, multiplication, division, or modulo.

14.4. UInt64 Examples





| <div>     </div> | | | | | |
|--|--------------------|------------------------------------|--------------------|----------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | UInt64.MinValue | | 0 | 2025-11-10 16:36:09.688 |
| MyAttribute02 | Regular Expression | 965874238 / 50 | | 19317484 | 2025-11-10 16:36:09.688 |
| MyAttribute03 | Regular Expression | UInt64.Parse("350") | | 350 | 2025-11-10 16:36:09.689 |
| MyAttribute04 | Regular Expression | (\$MyAttribute01).ToString("D4") | | 0000 | 2025-11-10 16:36:09.688 |
| MyAttribute05 | Regular Expression | 786489 * 5 | | 3932445 | 2025-11-10 16:36:09.690 |
| MyAttribute06 | Regular Expression | 548963 - 7845 | | 541118 | 2025-11-10 16:36:09.690 |
| MyAttribute07 | Regular Expression | \$MyAttribute01 >= UInt64.MinValue | | true | 2025-11-10 16:36:09.688 |
| MyAttribute08 | Regular Expression | \$MyAttribute03 << 2 | | 1400 | 2025-11-10 16:36:09.689 |
| MyAttribute09 | Regular Expression | \$MyAttribute03 >> 2 | | 87 | 2025-11-10 16:36:09.689 |

Figure 22: UInt64 Operations Example

15. Float Properties, Methods and Operators

The **Float** category provides a collection of properties and methods to manage, manipulate, and format single-precision floating-point numbers.

15.1. Float Properties

| Property | Type | Description |
|------------------------|-------|---|
| <i>MaxValue</i> | Float | Represents the largest possible value of Float. |
| <i>MinValue</i> | Float | Represents the smallest possible value of Float . |

| | | |
|--------------------------------|-------|---|
| <i>Epsilon</i> | Float | Represents the smallest positive Float value that is greater than zero. |
| <i>NaN</i> | Float | Represents not a number (NaN). |
| <i>PositiveInfinity</i> | Float | Represents positive infinity. |
| <i>NegativeInfinity</i> | Float | Represents negative infinity. |

Table 74: Float Properties

15.2. Float Methods

| Method | Return Type | Description |
|--------------------------------|-------------|--|
| <i>CompareTo(float)</i> | Int32 | Compares this instance to a specified single-precision floating-point number and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the specified single-precision floating-point number. |
| <i>Equals(float)</i> | Boolean | Returns a value indicating whether this instance and a specified Float object represent the same value or not. |
| <i>IsFinite(float)</i> | Boolean | Determines whether the specified value is finite (zero, subnormal or normal) or not. |

| | | |
|---|---------|--|
| <i>IsInfinity(float)</i> | Boolean | Returns a value indicating whether the specified number evaluates to negative or positive infinity. |
| <i>IsNaN(float)</i> | Boolean | Returns a value that indicates whether the specified value is not a number (NaN) or not. |
| <i>IsNegative(float)</i> | Boolean | Determines whether the specified value is negative or not. |
| <i>IsNegativeInfinity(float)</i> | Boolean | Returns a value indicating whether the specified number evaluates to negative infinity or not. |
| <i>IsNormal(float)</i> | Boolean | Determines whether the specified value is normal or not. |
| <i>IsPositiveInfinity(float)</i> | Boolean | Returns a value indicating whether the specified number evaluates to positive infinity or not. |
| <i>IsSubnormal(float)</i> | Boolean | Determines whether the specified value is subnormal or not. |
| <i>Parse(string)</i> | Float | Converts the string representation of a number to its single-precision floating-point number equivalent. |

| | | |
|--------------------------------|--------|--|
| <i>ToString()</i> | String | Converts the numeric value of this instance to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the numeric value of this instance to its equivalent string representation, using the specified format. |

Table 75: Float Methods

15.3. Float Operators

| Operator | Return Type | Description |
|----------------------------------|-------------|---|
| <i>Addition (+)</i> | Float | The addition operator + computes the sum of its operands. |
| <i>Subtraction (-)</i> | Float | The subtraction operator - subtracts its right-hand operand from its left-hand operand. |
| <i>Multiplication (*)</i> | Float | The multiplication operator * computes the product of its operands. |
| <i>Division (/)</i> | Float | The division operator / divides its left-hand operand by its right-hand operand. |
| <i>Remainder (%)</i> | Float | The remainder operator % computes the remainder after dividing its left-hand operand by its right-hand operand. |

| | | |
|--|---------|--|
| <i>Equality (==)</i> | Boolean | The equality operator == returns true if its operands are equal, false otherwise. |
| <i>Inequality (!=)</i> | Boolean | The inequality operator <> returns true if its operands aren't equal, false otherwise. |
| <i>GreaterThan (>)</i> | Boolean | The > operator returns true if its left-hand operand is greater than its right-hand operand, false otherwise. |
| <i>GreaterThanOrEqual (>=)</i> | Boolean | The >= operator returns true if its left-hand operand is greater than or equal to its right-hand operand, false otherwise. |
| <i>LessThan (<)</i> | Boolean | The < operator returns true if its left-hand operand is less than its right-hand operand, false otherwise. |
| <i>LessThanOrEqual (<=)</i> | Boolean | The <= operator returns true if its left-hand operand is less than or equal to its right-hand operand, false otherwise. |

Table 76: Float Operators

15.4. Float Examples





|     | | | | | |
|---|--------------------|---------------|------------------------------------|---------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | Float.MaxValue | 3.4028235E+38 | 2025-11-10 17:00:25.166 |
| MyAttribute02 | Regular Expression | | Float.Epsilon | 1E-45 | 2025-11-10 17:00:25.166 |
| MyAttribute03 | Regular Expression | | Float.NaN | NaN | 2025-11-10 17:00:25.167 |
| MyAttribute04 | Regular Expression | | (\$MyAttribute01).ToString() | 3.4028235E+38 | 2025-11-10 17:00:25.166 |
| MyAttribute05 | Regular Expression | | Float.IsInfinity(\$MyAttribute01) | false | 2025-11-10 17:00:25.166 |
| MyAttribute06 | Regular Expression | | Float.IsNaN(\$MyAttribute03) | true | 2025-11-10 17:00:25.167 |
| MyAttribute07 | Regular Expression | | \$MyAttribute01 == \$MyAttribute02 | false | 2025-11-10 17:00:25.166 |
| MyAttribute08 | Regular Expression | | \$MyAttribute01 % 965874 | 357630 | 2025-11-10 17:00:25.166 |

Figure 23: Float Operations Example

16. Double Properties, Methods and Operators

The **Double** category provides a collection of properties and methods to manage, manipulate, and format double-precision floating-point numbers.

16.1. Double Properties

| Property | Type | Description |
|------------------------|--------|---|
| <i>MaxValue</i> | Double | Represents the largest possible value of Double. |
| <i>MinValue</i> | Double | Represents the smallest possible value of Double. |

| | | |
|--------------------------------|--------|--|
| <i>Epsilon</i> | Double | Represents the smallest positive Double value that is greater than zero. |
| <i>NaN</i> | Double | Represents not a number (NaN). |
| <i>PositiveInfinity</i> | Double | Represents positive infinity. |
| <i>NegativeInfinity</i> | Double | Represents negative infinity. |

Table 77: Double Properties

16.2. Double Methods

| Method | Return Type | Description |
|---------------------------------|-------------|--|
| <i>CompareTo(double)</i> | Int32 | Compares this instance to a specified double-precision floating-point number and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the specified double-precision floating-point number. |
| <i>Equals(double)</i> | Boolean | Returns a value indicating whether this instance and a specified Float object represent the same value or not. |
| <i>IsFinite(double)</i> | Boolean | Determines whether the specified value is finite (zero, subnormal or normal) or not. |

| | | |
|--|---------|--|
| <i>IsInfinity(double)</i> | Boolean | Returns a value indicating whether the specified number evaluates to negative or positive infinity or not. |
| <i>IsNaN(double)</i> | Boolean | Returns a value that indicates whether the specified value is not a number (NaN) or not. |
| <i>IsNegative(double)</i> | Boolean | Determines whether the specified value is negative or not. |
| <i>IsNegativeInfinity(double)</i> | Boolean | Returns a value indicating whether the specified number evaluates to negative infinity or not. |
| <i>IsNormal(double)</i> | Boolean | Determines whether the specified value is normal or not. |
| <i>IsPositiveInfinity(double)</i> | Boolean | Returns a value indicating whether the specified number evaluates to positive infinity or not. |
| <i>IsSubnormal(double)</i> | Boolean | Determines whether the specified value is subnormal or not. |
| <i>Parse(string)</i> | Double | Converts the string representation of a number to its double-precision floating-point number equivalent. |

| | | |
|--------------------------------|--------|--|
| <i>ToString()</i> | String | Converts the numeric value of this instance to its equivalent string representation. |
| <i>ToString(string)</i> | String | Converts the numeric value of this instance to its equivalent string representation, using the specified format. |

Table 78: Double Methods

16.3. Double Operators

(!) Note

The **Double** category uses the same operators defined for the **Float** type. However, The return type for the operators is **Double** for operators such as addition, subtraction, multiplication, division, or modulo.

16.4. Double Examples





|     | | | | | |
|---|--------------------|------------------------------------|--------------------|-------------------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | Double.MaxValue | | 1.7976931348623157E+308 | 2025-11-10 17:11:51.572 |
| MyAttribute02 | Regular Expression | Double.NegativeInfinity | | -Infinity | 2025-11-10 17:11:51.573 |
| MyAttribute03 | Regular Expression | Double.NaN | | NaN | 2025-11-10 17:11:51.573 |
| MyAttribute04 | Regular Expression | (\$MyAttribute01).ToString() | | 1.7976931348623157E+308 | 2025-11-10 17:11:51.572 |
| MyAttribute05 | Regular Expression | Double.IsInfinity(\$MyAttribute02) | | true | 2025-11-10 17:11:51.573 |
| MyAttribute06 | Regular Expression | Double.IsNaN(\$MyAttribute03) | | true | 2025-11-10 17:11:51.573 |
| MyAttribute07 | Regular Expression | \$MyAttribute01 != \$MyAttribute02 | | true | 2025-11-10 17:11:51.573 |
| MyAttribute08 | Regular Expression | \$MyAttribute01 / 86247139 | | 2.0843510355309475E+300 | 2025-11-10 17:11:51.572 |

Figure 24: Double Operations Example

17. String Properties, Methods and Operators

The **String** category provides a collection of properties, methods, and operators to manipulate, compare, search, and format text.

17.1. String Properties

| Property | Type | Description |
|----------------------|--------|---|
| <i>Empty</i> | String | Represents the empty string. |
| <i>Length</i> | Int32 | Gets the number of characters in the current string instance. |

Table 79: String Properties

17.2. String Methods

| Method | Return Type | Description |
|--|-------------|--|
| <i>Compare(string, int32, string, int32, int32)</i> | Int32 | Compares substrings of two specified String objects and returns an integer that indicates their relative position in the sort order. |
| <i>Compare(string, int32, string, int32, int32, bool)</i> | Int32 | Compares substrings of two specified String objects, ignoring or honoring their case, and returns an integer that indicates their relative position in the sort order. |

| | | |
|---|--------|--|
| <i>Compare(string, string)</i> | Int32 | Compares two specified String objects and returns an integer that indicates their relative position in the sort order. |
| <i>Compare(string, string, bool)</i> | Int32 | Compares two specified String objects, ignoring or honoring their case, and returns an integer that indicates their relative position in the sort order. |
| <i>CompareOrdinal(string, string)</i> | Int32 | Compares two specified String objects by evaluating the numeric values of the corresponding Char objects in each String. |
| <i>CompareOrdinal(string, int32, string, int32, int32)</i> | Int32 | Compares substrings of two specified String objects by evaluating the numeric values of the corresponding Char objects in each substring. |
| <i>CompareTo(string)</i> | Int32 | Compares this instance with a specified String object and indicates whether this instance precedes, follows, or appears in the same position in the sort order as the specified String or not. |
| <i>Concat(string, string)</i> | String | Concatenates two specified instances of String. |

| | | |
|---|---------|--|
| <i>Concat(string, string, string)</i> | String | Concatenates three specified instances of String. |
| <i>Concat(string, string, string str2, string)</i> | String | Concatenates four specified instances of String. |
| <i>Concat(string, string, string, string, ...)</i> | String | Concatenates multiple specified instances of String. |
| <i>Contains(char)</i> | Boolean | Returns a value indicating whether a specified character occurs within this String or not. |
| <i>Contains(string)</i> | Boolean | Returns a value indicating whether a specified substring occurs within this String or not. |
| <i>Equals(string)</i> | Boolean | Determines whether this instance and another specified String object have the same value or not. |
| <i>Equals(string, string)</i> | Boolean | Determines whether two specified String objects have the same value or not. |
| <i>EndsWith(char)</i> | Boolean | Determines whether the end of this String instance matches the specified character or not. |

| | | |
|---|---------|--|
| <i>EndsWith(string)</i> | Boolean | Determines whether the end of this String instance matches the specified String or not. |
| <i>Format(string, object)</i> | String | Replaces the format items in a String with the String representation of a specified object. |
| <i>Format(string, object, object)</i> | String | Replaces the format items in a String with the String representation of two specified objects. |
| <i>Format(string, object, object, object)</i> | String | Replaces the format items in a String with the String representation of three specified objects. |
| <i>Format(string, object, object, object, ...)</i> | String | Replaces the format items in a String with the String representation of multiple specified objects. |
| <i>IndexOf(char)</i> | Int32 | Reports the zero-based index of the first occurrence of the specified Unicode character in this String. |
| <i>IndexOf(char, int32)</i> | Int32 | Reports the zero-based index of the first occurrence of the specified Unicode character in this String. The search starts at a specified character position. |

| | | |
|---|---------|---|
| <i>IndexOf(char, int32, int32)</i> | Int32 | Reports the zero-based index of the first occurrence of the specified character in this instance. The search starts at a specified character position and examines a specified number of character positions. |
| <i>IndexOf(string)</i> | Int32 | Reports the zero-based index of the first occurrence of the specified String in this instance. |
| <i>IndexOf(string, int32)</i> | Int32 | Reports the zero-based index of the first occurrence of the specified String in this instance. The search starts at a specified character position. |
| <i>IndexOf(string, int32, int32)</i> | Int32 | Reports the zero-based index of the first occurrence of the specified String in this instance. The search starts at a specified character position and examines a specified number of character positions. |
| <i>Insert(int, string)</i> | String | Returns a new String in which a specified String is inserted at a specified index position in this instance. |
| <i>IsNormalized()</i> | Boolean | Indicates whether this String is in Unicode normalization form C or not. |

| | | |
|---|---------|--|
| <i>IsNullOrEmpty(string)</i> | Boolean | Indicates whether the specified String is null or an empty String ("") or not. |
| <i>IsNullOrWhiteSpace(string)</i> | Boolean | Indicates whether a specified String is null, empty, or consists only of white-space characters or not. |
| <i>Join(char, string, string, string, ...)</i> | String | Concatenates multiple strings, using the specified separator between each member. |
| <i>LastIndexOf(char)</i> | Int32 | Reports the zero-based index position of the last occurrence of a specified Unicode character within this instance. |
| <i>LastIndexOf(char, int32)</i> | Int32 | Reports the zero-based index position of the last occurrence of a specified Unicode character within this instance. The search starts at a specified character position and proceeds backward toward the beginning of the String. |
| <i>LastIndexOf(char, int32, int32)</i> | Int32 | Reports the zero-based index position of the last occurrence of the specified Unicode character in a substring within this instance. The search starts at a specified character position and proceeds backward toward the beginning of the |

| | | |
|---|--------|---|
| | | String for a specified number of character positions. |
| <i>LastIndexOf(string)</i> | Int32 | Reports the zero-based index position of the last occurrence of a specified String within this instance. |
| <i>LastIndexOf(string, int32)</i> | Int32 | Reports the zero-based index position of the last occurrence of a specified String within this instance. The search starts at a specified character position and proceeds backward toward the beginning of the String. |
| <i>LastIndexOf(string, int32, int32)</i> | Int32 | Reports the zero-based index position of the last occurrence of the specified String in a substring within this instance. The search starts at a specified character position and proceeds backward toward the beginning of the String for a specified number of character positions. |
| <i>PadLeft(int32)</i> | String | Returns a new String that right-aligns the characters in this instance by padding them with spaces on the left, for a specified total length. |
| <i>PadLeft(int32, char)</i> | String | Returns a new String that right-aligns the characters in this instance by padding |

| | | |
|-------------------------------------|--------|--|
| | | them on the left with a specified Unicode character, for a specified total length. |
| <i>PadRight(int32)</i> | String | Returns a new String that left-aligns the characters in this instance by padding them with spaces on the right, for a specified total length. |
| <i>PadRight(int32, char)</i> | String | Returns a new String that left-aligns the characters in this instance by padding them on the right with a specified Unicode character, for a specified total length. |
| <i>Remove(int32)</i> | String | Returns a new String in which all the characters in the current instance, beginning at a specified position and continuing through the last position, have been deleted. |
| <i>Remove(int32, int32)</i> | String | Returns a new String in which a specified number of characters in the current instance beginning at a specified position have been deleted. |
| <i>Replace(char, char)</i> | String | Returns a new String in which all occurrences of a specified Unicode character in this instance are replaced with another specified Unicode character. |

| | | |
|--|---------|---|
| <i>Replace(string, string)</i> | String | Returns a new String in which all occurrences of a specified String in the current instance are replaced with another specified String. |
| <i>ReplaceLineEndings()</i> | String | Replaces all newline sequences in the current String with the newline String defined for the current environment (\r\n for non-Unix platforms, or \n for Unix platforms). |
| <i>ReplaceLineEndings(string)</i> | String | Replaces all newline sequences in the current String with replacementText. |
| <i>StartsWith(char)</i> | Boolean | Determines whether this String instance starts with the specified character or not. |
| <i>StartsWith(string)</i> | Boolean | Determines whether the beginning of this String instance matches the specified String or not. |
| <i>Substring(int32)</i> | String | Retrieves a substring from this instance. The substring starts at a specified character position and continues to the end of the String. |
| <i>Substring(int32, int32)</i> | String | Retrieves a substring from this instance. The substring starts at a specified |

| | | |
|---|--------|--|
| | | character position and has a specified length. |
| <i>ToLower()</i> | String | Returns a copy of this String converted to lowercase. |
| <i>ToLowerInvariant()</i> | String | Returns a copy of this String object converted to lowercase using the casing rules of the invariant culture. |
| <i>ToString()</i> | String | Returns this instance of String, no actual conversion is performed. |
| <i>ToUpper()</i> | String | Returns a copy of this String converted to uppercase. |
| <i>ToUpperInvariant()</i> | String | Returns a copy of this String object converted to uppercase using the casing rules of the invariant culture. |
| <i>Trim()</i> | String | Removes all leading and trailing white-space characters from the current String. |
| <i>Trim(char)</i> | String | Removes all leading and trailing instances of a character from the current String. |
| <i>Trim(char, char, char, ...)</i> | String | Removes all leading and trailing occurrences of multiple specified characters from the current String. |

| | | |
|--|--------|--|
| <i>TrimEnd()</i> | String | Removes all the trailing white-space characters from the current String. |
| <i>TrimEnd(char)</i> | String | Removes all the trailing occurrences of a character from the current String. |
| <i>TrimEnd(char, char, char, ...)</i> | String | Removes all trailing occurrences of multiple specified characters from the current String. |
| <i>TrimStart()</i> | String | Removes all the leading white-space characters from the current String. |
| <i>TrimStart(char)</i> | String | Removes all the leading occurrences of a character from the current String. |
| <i>TrimStart(char, char, char, ...)</i> | String | Removes all leading occurrences of multiple specified characters from the current String. |

Table 80: String Methods

17.3. String Operators

| Operator | Return Type | Description |
|-----------------------------|-------------|--|
| <i>Equality (==)</i> | Boolean | Determines whether two specified strings have the same value or not. |

| | | |
|-------------------------------|---------|--|
| <i>Inequality (!=)</i> | Boolean | Determines whether two specified strings have different values or not. |
|-------------------------------|---------|--|

Table 81: String Operators

17.4. String Examples





|     | | | | | |
|---|--------------------|---------------|---|-----------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Static | Hello | | Hello | 2025-11-11 12:08:30.674 |
| MyAttribute02 | Regular Expression | | (\$MyAttribute01).Contains("World") | false | 2025-11-11 12:08:30.674 |
| MyAttribute03 | Regular Expression | | String.Concat(\$MyAttribute01, " World") | Hello World | 2025-11-11 12:08:30.674 |
| MyAttribute04 | Regular Expression | | (\$MyAttribute03).Contains("World") | true | 2025-11-11 12:08:30.674 |
| MyAttribute05 | Regular Expression | | "Hello World".IndexOf("World", 0) | 6 | 2025-11-11 12:08:30.675 |
| MyAttribute06 | Regular Expression | | (\$MyAttribute03).IsNormalized() | true | 2025-11-11 12:08:30.674 |
| MyAttribute07 | Regular Expression | | (\$MyAttribute03).Remove(0, 5) | World | 2025-11-11 12:08:30.674 |
| MyAttribute08 | Regular Expression | | (\$MyAttribute03).Replace("World", "- World -") | Hello - World - | 2025-11-11 12:08:30.674 |
| MyAttribute09 | Regular Expression | | \$MyAttribute03 == "Hello World" | true | 2025-11-11 12:08:30.674 |

Figure 25: String Operations Example

18. DateTime Properties, Methods and Operators

The **DateTime** category provides a collection of properties, methods, and operators to manage, manipulate, and format dates and times.

18.1. DateTime Properties

| Property | Type | Description |
|------------------------|----------|--|
| <i>MaxValue</i> | DateTime | Represents the largest possible value of DateTime. |

| | | |
|-------------------------|----------|---|
| <i>MinValue</i> | DateTime | Represents the smallest possible value of DateTime. |
| <i>UnixEpoch</i> | DateTime | The value of this constant is equivalent to 00:00:00.0000000 UTC, January 1, 1970, in the Gregorian calendar. UnixEpoch defines the point in time when Unix time is equal to 0. |
| <i>Now</i> | DateTime | Gets a DateTime object that is set to the current date and time on this computer, expressed as the local time. |
| <i>Today</i> | DateTime | Gets the current date. |
| <i>UtcNow</i> | DateTime | Gets a DateTime object that is set to the current date and time on this computer, expressed as Coordinated Universal Time (UTC). |
| <i>Date</i> | DateTime | Gets the date component of this instance. |
| <i>DayOfWeek</i> | String | Gets the day of the week represented by this instance. |
| <i>Kind</i> | String | Specifies whether a DateTime object represents a local time, a Coordinated Universal Time (UTC), or is not specified as either local time or UTC. |

| | | |
|---------------------------|-------|---|
| <i>Ticks</i> | Int64 | Gets the number of ticks that represent the date and time of this instance. |
| <i>Millisecond</i> | Int32 | Gets the milliseconds component of the date represented by this instance. |
| <i>Second</i> | Int32 | Gets the seconds component of the date represented by this instance. |
| <i>Minute</i> | Int32 | Gets the minute component of the date represented by this instance. |
| <i>Hour</i> | Int32 | Gets the hour component of the date represented by this instance. |
| <i>Day</i> | Int32 | Gets the day of the month represented by this instance. |
| <i>Month</i> | Int32 | Gets the month component of the date represented by this instance. |
| <i>Year</i> | Int32 | Gets the year component of the date represented by this instance. |
| <i>DayOfYear</i> | Int32 | Gets the day of the year represented by this instance. |

Table 82: DateTime Properties

18.2. DateTime Methods

| Method | Return Type | Description |
|---|-------------|---|
| <i>Compare(DateTime, DateTime)</i> | Int32 | Compares two instances of DateTime and returns an integer that indicates whether the first instance is earlier than, the same as, or later than the second instance. |
| <i>DaysInMonth(int32, int32)</i> | Int32 | Returns the number of days in the specified month and year. |
| <i>CompareTo(DateTime)</i> | Int32 | Compares the value of this instance to a specified DateTime value and returns an integer that indicates whether this instance is earlier than, the same as, or later than the specified DateTime value. |
| <i>Equals(DateTime, DateTime)</i> | Boolean | Returns a value indicating whether two DateTime instances have the same date and time value or not. |
| <i>IsLeapYear(int32)</i> | Boolean | Returns an indication whether the specified year is a leap year or not. |
| <i>IsDaylightSavingTime()</i> | Boolean | Indicates whether this instance of DateTime is within the daylight-saving time range for the current time zone or not. |

| | | |
|---------------------------------------|----------|--|
| <i>FromBinary(int64)</i> | DateTime | Deserializes a 64-bit binary value and recreates an original serialized DateTime object. |
| <i>FromFileTime(int64)</i> | DateTime | Converts the specified Windows file time to an equivalent local time. |
| <i>FromFileTimeUtc(int64)</i> | DateTime | Converts the specified Windows file time to an equivalent UTC time. |
| <i>FromOADate(double)</i> | DateTime | Returns a DateTime equivalent to the specified OLE Automation Date. |
| <i>AddTicks(int64)</i> | DateTime | Returns a new DateTime that adds the specified number of ticks to the value of this instance. |
| <i>AddMilliseconds(double)</i> | DateTime | Returns a new DateTime that adds the specified number of milliseconds to the value of this instance. |
| <i>AddSeconds(double)</i> | DateTime | Returns a new DateTime that adds the specified number of seconds to the value of this instance. |
| <i>AddMinutes(double)</i> | DateTime | Returns a new DateTime that adds the specified number of minutes to the value of this instance. |

| | | |
|---------------------------------|----------|---|
| <i>AddHours(double)</i> | DateTime | Returns a new DateTime that adds the specified number of hours to the value of this instance. |
| <i>AddDays(double)</i> | DateTime | Returns a new DateTime that adds the specified number of days to the value of this instance. |
| <i>AddMonths(int32)</i> | DateTime | Returns a new DateTime that adds the specified number of months to the value of this instance. |
| <i>AddYears(int32)</i> | DateTime | Returns a new DateTime that adds the specified number of years to the value of this instance. |
| <i>Parse(string)</i> | DateTime | Converts the string representation of a date and time to its DateTime equivalent by using the conventions of the current culture. |
| <i>ToLocalTime()</i> | DateTime | Converts the value of the current DateTime object to local time. |
| <i>ToUniversalTime()</i> | DateTime | Converts the value of the current DateTime object to Coordinated Universal Time (UTC). |

| | | |
|-----------------------------------|--------|---|
| <i>ToLongDateString()</i> | String | Converts the value of the current DateTime object to its equivalent long date string representation. |
| <i>ToLongTimeString()</i> | String | Converts the value of the current DateTime object to its equivalent long time string representation. |
| <i>ToShortDateString()</i> | String | Converts the value of the current DateTime object to its equivalent short date string representation. |
| <i>ToString()</i> | String | Converts the value of the current DateTime object to its equivalent string representation using the formatting conventions of the current culture. |
| <i>ToString(string)</i> | String | Converts the value of the current DateTime object to its equivalent string representation using the specified format and the formatting conventions of the current culture. |
| <i>ToBinary()</i> | Int64 | Serializes the current DateTime object to a 64-bit binary value that subsequently can be used to recreate the DateTime object. |

| | | |
|-------------------------------|--------|--|
| <i>ToFileTime()</i> | Int64 | Converts the value of the current DateTime object to a Windows file time. |
| <i>ToFileTimeUtc()</i> | Int64 | Converts the value of the current DateTime object to a Windows file time. |
| <i>ToOADate()</i> | Double | Converts the value of this instance to the equivalent OLE Automation date. |

Table 83: DateTime Methods

18.3. DateTime Operators

| Operator | Return Type | Description |
|--|-------------|--|
| <i>Equality (==)</i> | Boolean | Determines whether two specified instances of DateTime are equal or not. |
| <i>Inequality (!=)</i> | Boolean | Determines whether two specified instances of DateTime are not equal or not. |
| <i>GreaterThan (>)</i> | Boolean | Determines whether one specified DateTime is later than another specified DateTime or not. |
| <i>GreaterThanOrEqual (>=)</i> | Boolean | Determines whether one specified DateTime represents a date and time that is the same as or later than another specified DateTime. |

| | | |
|---------------------------------------|---------|--|
| <i>LessThan (<)</i> | Boolean | Determines whether one specified DateTime is earlier than another specified DateTime or not. |
| <i>LessThanOrEqual (<=)</i> | Boolean | Determines whether one specified DateTime represents a date and time that is the same as or earlier than another specified DateTime. |

Table 84: DateTime Operators

18.4. DateTime Examples





|     | | | | | |
|---|--------------------|---------------|---|--------------------------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Regular Expression | | DateTime.MinValue | 0001-01-01 00:13:35.000 | 2025-11-11 12:18:55.101 |
| MyAttribute02 | Regular Expression | | DateTime.MaxValue | 10000-01-01 00:59:59.000 | 2025-11-11 12:18:55.102 |
| MyAttribute03 | Regular Expression | | DateTime.Now | 2025-11-11 13:18:55.000 | 2025-11-11 12:18:55.102 |
| MyAttribute04 | Regular Expression | | DateTime.DaysInMonth(2025, 02) | 28 | 2025-11-11 12:18:55.103 |
| MyAttribute05 | Regular Expression | | (\$MyAttribute03).AddDays(10) | 2025-11-21 13:18:55.000 | 2025-11-11 12:18:55.102 |
| MyAttribute06 | Regular Expression | | (\$MyAttribute03).ToString("yyyy-MM-dd HH:mm:ss") | 2025-11-11 12:18:55 | 2025-11-11 12:18:55.102 |

Figure 26: DateTime Operations Example

19. Custom Methods

| Method | Return Type | Description |
|--|-------------|---|
| <i>Conditional(bool, object valueTrue, object valueFalse)</i> | Object | Returns valueTrue or valueFalse based on the result of the predicate. |

Table 85: Custom Methods

19.1. Custom Methods Examples





|     | | | | | |
|---|--------------------|---------------|---|-------|-------------------------|
| General | | Attributes | | | |
| Property | Address | Default Value | Regular Expression | Value | Timestamp |
| MyAttribute01 | Static | 35 | | 35 | 2025-11-11 12:26:46.703 |
| MyAttribute02 | Regular Expression | | Custom.Conditional(\$MyAttribute01 > 25, "Hot", "Cold") | Hot | 2025-11-11 12:26:46.703 |

Figure 27: Custom Methods Example

(!) Note

Some general expressions can also be defined using conditional logic.

Example:

Temp > 25.0 ? "Hot" : "Cold"

Evaluates the condition (Temp > 25.0) and returns either the **trueValue** or **falseValue**, depending on whether the condition evaluates to **true** or **false**.

For additional information on this guide, questions, or problems to report, please contact:

Offices

- Americas: +1 713 609 9208
- Europe-Africa-Middle East: +216 71 195 360

Email

- Support Services: customerservice@integrationobjects.com
- Sales: sales@integrationobjects.com

To find out how you can benefit from other Integration Objects' products and services, please visit our website:

Online

- www.integrationobjects.com