[MS-LETSF]:

LINQ Expression Tree Serialization Format

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
5/24/2012	1.0	New	Released new document.
6/29/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/23/2012	2.0	Major	Updated and revised the technical content.
3/26/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
6/11/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
12/5/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2014	3.0	Major	Updated and revised the technical content.
5/20/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
5/10/2016	4.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The LINQ Expression Tree Serialization Format provides a structure for serializing an arbitrary **LINQ expression tree**. The serialization enables scenarios in which the tree is communicated to a remote host for evaluation.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

common language runtime (CLR): The core runtime engine in the Microsoft .NET Framework for executing applications. The common language runtime supplies managed code with services such as cross-language integration, code access security, object lifetime management, and debugging and profiling support.

complex type: An element that can contain other elements or attributes and appears as <complexType> in an **XML document**. See also **simple type**.

expression: A combination of operators, symbols, constants, literal values, functions, names of fields or columns (2), controls, and properties that evaluates to a single value.

LINQ expression tree: A representation of an **expression** as a tree data structure, in which each node of the tree represents a subexpression.

LINQ expression tree node: A child element of an expression tree.

root element: The top-level element in an **XML document**. It contains all other elements and is not contained by any other element, as described in [XML].

simple type: An element that can contain only text and appears as <simpleType> in an **XML document** or any attribute of an element. Attributes are considered simple types because they contain only text. See also **complex type**.

XML document: A document object that is well formed, as described in [XML10/5], and might be valid. An XML document has a logical structure that is composed of declarations, elements, comments, character references, and processing instructions. It also has a physical structure that is composed of entities, starting with the root, or document, entity.

XML namespace: A collection of names that is used to identify elements, types, and attributes in XML documents identified in a URI reference [RFC3986]. A combination of XML namespace and local name allows XML documents to use elements, types, and attributes that have the same names but come from different sources. For more information, see [XMLNS-2ED].

XML schema: A description of a type of **XML document** that is typically expressed in terms of constraints on the structure and content of documents of that type, in addition to the basic syntax constraints that are imposed by XML itself. An XML schema provides a view of a document type at a relatively high level of abstraction.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents

in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2/2] Biron, P., and Malhotra, A., Eds., "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

1.2.2 Informative References

[MS-CEPM] Microsoft Corporation, "Microsoft Complex Event Processing Engine Manageability Protocol".

[MSDN-AsmblyNameClass] Microsoft Corporation, "AssemblyName Class", http://msdn.microsoft.com/en-us/library/system.reflection.assemblyname.aspx

[MSDN-DataContractSer] Microsoft Corporation, "DataContractSerializer Class", http://msdn.microsoft.com/en-us/library/system.runtime.serialization.datacontractserializer.aspx

[MSDN-ExprClass] Microsoft Corporation, "Expression Class", http://msdn.microsoft.com/en-us/library/system.linq.expressions.expression.aspx

[MSDN-ExprTrees] Microsoft Corporation, "Expression Trees (C# and Visual Basic)", http://msdn.microsoft.com/en-us/library/bb397951.aspx

[MSDN-IEnumerable] Microsoft Corporation, "IEnumerable Interface", http://msdn.microsoft.com/en-us/library/system.collections.ienumerable.aspx

1.3 Overview

LINQ expression trees represent code in a tree-like data structure in which each node is an **expression**. Such an expression might be a method call or a binary operation such as x < y.

The LINQ expression tree serialization format provides for serialization of arbitrary LINQ expression trees. A typical scenario that uses this format is one in which a client communicates a LINQ expression tree that a server evaluates.

For more information about LINQ expression trees, see [MSDN-ExprTrees] and [MSDN-ExprClass].

1.4 Relationship to Protocols and Other Structures

The LINQ expression tree serialization format is used by the Microsoft complex event processing (CEP) engine manageability protocol, as defined in [MS-CEPM].

1.5 Applicability Statement

This format is applicable to the serialization of LINQ expression trees for later storage or transport.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

As an **XML schema**, the LINQ expression tree serialization format comprises both **complex types** and **simple types**, as well as a single element that provides the root for a conforming **XML document**. This section defines these components. The full XML schema [XMLSCHEMA1] is included in Appendix A: Full XML Schema (section 5).

2.1 Namespace

The **XML namespace** URI for the LINQ expression tree serialization format (see Appendix A: Full XML Schema (section 5)) is:

http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression

2.2 Global Element

2.2.1 ExpressionRoot

An element of type ExpressionRoot specifies the **root element** for an XML document that describes a LINQ expression tree.

The following XML schema fragment specifies the contents of the **ExpressionRoot** element.

```
<xs:element name="ExpressionRoot" type="tns1:ExpressionRoot"/>
```

2.3 Complex Types

2.3.1 ExpressionRoot

An **ExpressionRoot** type contains a LINQ expression tree definition along with references to **common language runtime (CLR)** types and assemblies.

The following XML schema fragment specifies the contents of the **ExpressionRoot** complex type.

```
<xs:complexType name="ExpressionRoot">
  <xs:sequence>
    <xs:element name="Definition" type="tns1:Expression"/>
    <xs:element name="Types" type="tns1:ArrayOfType"/>
    <xs:element name="Assemblies" type="tns1:ArrayOfAssembly"/>
  </xs:sequence>
</xs:complexType>
```

Each **ExpressionRoot** element contains the following child elements:

- **Definition:** An Expression element that specifies the LINQ expression tree definition. Any TypeReference that appears in the definition MUST have an **ID** that matches the **ID** of a **Type** element under the **Types** element.
- **Types:** An ArrayOfType element that specifies an array of CLR type descriptions referenced by the **Definition** element. Any AssemblyReference appearing in a type description MUST have an **ID** that matches the **ID** of an Assembly element under the **Assemblies** element.
- **Assemblies:** An ArrayOfAssembly element that specifies the descriptions of the assemblies referenced in the **Types** element.

2.3.2 Expression

The **Expression** type provides the base type from which the types that represent **LINQ expression tree nodes** are derived.

This XML schema fragment specifies the contents of the **Expression** complex type.

```
<xs:complexType name="Expression">
  <xs:sequence/>
</xs:complexType>
```

2.3.2.1 Unary

The **Unary** type represents an expression that has a unary operator.

This XML schema fragment specifies the contents of the **Unary** complex type.

```
<xs:complexType name="Unary">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="NodeType" type="tns1:UnaryNodeType"/>
        <xs:element name="Type" type="tns1:TypeReference"/>
        <xs:element name="Operand" type="tns1:Expression"/>
        <xs:element minOccurs="0" name="Method" type="tns1:MethodInfo"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **Unary** type contains the following child elements:

- **NodeType:** A UnaryNodeType element that specifies the node type of this **Unary** expression.
- **Type:** A TypeReference element that specifies the static type of the expression.
- **Operand:** An Expression element that specifies the operand of the unary operation.
- **Method:** An optional MethodInfo element that specifies the implementing method for the unary operation.

2.3.2.2 Binary

The **Binary** type represents an expression that has a binary operator.

This XML schema fragment specifies the contents of the **Binary** complex type.

```
<xs:complexType name="Binary">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="NodeType" type="tns1:BinaryNodeType"/>
        <xs:element name="Left" type="tns1:Expression"/>
        <xs:element name="Right" type="tns1:Expression"/>
        <xs:element minOccurs="0" name="IsLiftedToNull" type="xs:boolean"/>
        <xs:element minOccurs="0" name="Method" type="tns1:MethodInfo"/>
        <xs:element minOccurs="0" name="Conversion" type="tns1:Lambda"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```

```
</xs:complexType>
```

The **Binary** type contains the following child elements:

- **NodeType:** A BinaryNodeType element that specifies the node type of this **Binary** expression.
- **Left:** An Expression element that specifies the left operand of the binary operation.
- **Right:** An **Expression** element that specifies the right operand of the binary operation.
- **IsLiftedToNull:** An optional **boolean** element ([XMLSCHEMA2/2] section 3.2.2) that indicates whether the operator call is lifted to null. An operator call is lifted if the operator expects non-nullable operands but nullable operands are passed to it. If the value of **IsLiftedToNull** is true, the operator returns a nullable type, and if a nullable operand evaluates to null, the operator returns null. If **IsLiftedToNull** is false, the operator returns a nullable type, and if a nullable operand evaluates to null, the operator is evaluated instead of automatically being lifted to return null.
- **Method:** An optional MethodInfo element that specifies the implementing method for the binary operation.
- **Conversion:** An optional Lambda element that specifies the type conversion function that is used by a coalescing or compound assignment operation.

2.3.2.3 TypeBinary

The **TypeBinary** type represents an operation, such as a cast, between an expression and a type.

This XML schema fragment specifies the contents of the **TypeBinary** complex type.

```
<xs:complexType name="TypeBinary">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="NodeType" type="tns1:TypeBinaryNodeType"/>
        <xs:element name="TypeOperand" type="tns1:TypeReference"/>
        <xs:element name="Expression" type="tns1:Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **TypeBinary** type contains the following child elements:

- **NodeType:** A TypeBinaryNodeType element that specifies the node type of this **TypeBinary** expression.
- **TypeOperand:** A TypeReference element that specifies the type operand of a type test operation.
- **Expression:** An Expression element that specifies the expression operand of a type test operation.

2.3.2.4 Constant

The **Constant** type provides the base type for representations of expression that have a constant value.

This XML schema fragment specifies the contents of the **Constant** complex type.

```
<xs:complexType name="Constant" abstract="true">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element minOccurs="0" name="Type" type="tns1:TypeReference"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **Constant** type contains the following child element:

- **Type:** An optional TypeReference element that specifies the static type of the expression. The type can be omitted if it is implied by the serialized constant value.

2.3.2.4.1 ExpressionConstant

The **ExpressionConstant** type represents an expression that has a constant value in which the constant value is serialized as an expression that will evaluate to the constant value.

This XML schema fragment specifies the contents of the **ExpressionConstant** complex type.

```
<xs:complexType name="ExpressionConstant">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Constant">
      <xs:sequence>
        <xs:element name="Value" type="tns1:Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to the child element that is defined on its parent Constant type, the **ExpressionConstant** type contains the following child element:

- **Value:** An Expression element that specifies the value of the constant expression.

2.3.2.4.2 MethodInfoConstant

The **MethodInfoConstant** type represents an expression that has a constant value that represents method call information.

This XML schema fragment specifies the contents of the **MethodInfoConstant** complex type.

```
<xs:complexType name="MethodInfoConstant">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Constant">
      <xs:sequence>
        <xs:element name="Value" type="tns1:MethodInfo"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to the child element that is defined on its parent Constant type, the **MethodInfoConstant** type contains the following child element:

- **Value:** A MethodInfo element that specifies the value of the constant expression.

2.3.2.4.3 DelegateConstant

The **DelegateConstant** type represents an expression that has a constant value that represents a delegate value.

This XML schema fragment specifies the contents of the **DelegateConstant** complex type.

```
<xs:complexType name="DelegateConstant">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Constant">
      <xs:sequence>
        <xs:element minOccurs="0" name="Target" type="tns1:Constant"/>
        <xs:element name="Method" type="tns1:MethodInfo"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to the child element that is defined on its parent Constant type, the **DelegateConstant** type contains the following child elements:

- **Target:** An optional **Constant** element that specifies the class instance on which the current delegate invokes the instance method.
- **Method:** A MethodInfo element that specifies the method represented by the delegate.

2.3.2.4.4 PrimitiveConstant

The **PrimitiveConstant** type represents an expression that has a constant value that is natively serializable.

This XML schema fragment specifies the contents of the **PrimitiveConstant** complex type.

```
<xs:complexType name="PrimitiveConstant">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Constant">
      <xs:sequence>
        <xs:element minOccurs="0" nillable="true" name="Value" type="xs:anyType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to the child element that is defined on its parent Constant type, the **PrimitiveConstant** type contains the following child element:

- **Value:** An optional **anyType** element ([XMLSCHEMA1] section 2.2.1.1) that specifies the value of the constant expression.

2.3.2.4.5 TypeConstant

The **TypeConstant** type represents an expression that has a constant value that represents a type.

This XML schema fragment specifies the contents of the **TypeConstant** complex type.

```
<xs:complexType name="TypeConstant">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Constant">
      <xs:sequence>
        <xs:element name="Value" type="tns1:TypeReference"/>
```

```
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to the child element that is defined on its parent Constant type, the **TypeConstant** type contains the following child element:

- **Value:** A TypeReference element that specifies the value of the constant expression.

2.3.2.4.6 SerializedConstant

The **SerializedConstant** type represents an expression that has a constant value that can be serialized by using **DataContractSerializer** [MSDN-DataContractSer] in which the CLR type of the value is known by the serializer.

This XML schema fragment specifies the contents of the **SerializedConstant** complex type.

```
<xs:complexType name="SerializedConstant">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Constant">
      <xs:sequence>
        <xs:element minOccurs="0" name="Value" type="xs:string"/>
        <xs:element name="ValueType" type="tns1:TypeReference"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to the child element that is defined on its parent Constant type, the **SerializedConstant** type contains the following child elements:

- **Value:** An optional **string** element ([XMLSCHEMA2/2] section 3.2.1) that specifies the value of the constant expression in the form of XML document text.
- **ValueType:** A TypeReference element that specifies the CLR type of the serialized value. This **ValueType** can differ from the type of the expression.

2.3.2.5 Lambda

The **Lambda** type describes a lambda expression. This expression captures a block of code that is similar to a method body.

This XML schema fragment specifies the contents of the **Lambda** complex type.

```
<xs:complexType name="Lambda">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="Parameters" type="tns1:ArrayOfParameterDeclaration"/>
        <xs:element name="Body" type="tns1:Expression"/>
        <xs:element minOccurs="0" name="Type" type="tns1:TypeReference"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **Lambda** type contains the following child elements:

- **Parameters:** An ArrayOfParameterDeclaration element that specifies the parameters of the lambda expression.
- **Body:** An Expression element that specifies the body of the lambda expression. A ParameterReference appearing in the definition MUST have an **ID** that matches the **ID** of a ParameterDeclaration element under the **Parameters** element.
- **Type:** An optional TypeReference element that specifies the static type of the expression that this **Lambda** expression represents.

2.3.2.6 Conditional

The **Conditional** type represents an expression that has a conditional operator.

This XML schema fragment specifies the contents of the **Conditional** complex type.

```
<xs:complexType name="Conditional">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="Test" type="tns1:Expression"/>
        <xs:element name="IfTrue" type="tns1:Expression"/>
        <xs:element name="IfFalse" type="tns1:Expression"/>
        <xs:element minOccurs="0" name="Type" type="tns1:TypeReference"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **Conditional** type contains the following child elements:

- **Test:** An Expression element that specifies the test of the conditional operation. This expression MUST return a Boolean result.
- **IfTrue:** An **Expression** element that specifies the expression to execute if the test evaluates to true.
- **IfFalse:** An **Expression** element that specifies the expression to execute if the test evaluates to false.
- **Type:** An optional TypeReference element that specifies the static type of the expression that this **Conditional** represents.

2.3.2.7 New

The **New** type represents a constructor call.

This XML schema fragment specifies the contents of the **New** complex type.

```
<xs:complexType name="New">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:choice minOccurs="1" maxOccurs="1">
          <xs:element minOccurs="1" name="Type" type="tns1:TypeReference"/>
          <xs:sequence>
            <xs:element minOccurs="1" name="Constructor" type="tns1:ConstructorInfo"/>
            <xs:element minOccurs="0" name="Arguments" type="tns1:ArrayOfExpression"/>
            <xs:element minOccurs="0" name="Members" type="tns1:ArrayOfMemberInfo"/>
            </xs:sequence>
          </xs:choice>
```

```
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **New** type contains the following child elements:

- **Type:** A TypeReference element that specifies the static type of the expression that this **New** expression represents.
- **Constructor:** A ConstructorInfo element that specifies the called constructor.
- **Arguments:** An optional ArrayOfExpression element that specifies the arguments to the constructor.
- **Members:** An optional ArrayOfMemberInfo element that specifies the members that can retrieve the values of the fields that were initialized with constructor arguments.

2.3.2.8 NewArray

The **NewArray** type represents creating a new array and possibly initializing the elements of the new array.

This XML schema fragment specifies the contents of the **NewArray** complex type.

```
<xs:complexType name="NewArray">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="NodeType" type="tns1:NewArrayNodeType"/>
        <xs:element name="ElementType" type="tns1:TypeReference"/>
        <xs:element name="Expressions" type="tns1:ArrayOfExpression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **NewArray** type contains the following child elements:

- **NodeType:** A NewArrayNodeType element that specifies the node type of this **NewArray** expression.
- **ElementType:** A TypeReference element that specifies the element type of the array.
- **Expressions:** An ArrayOfExpression element that specifies the bounds of the array if the value of the **NodeType** element is NewArrayBounds (see section 2.4.4), or the values to initialize the elements of the new array if the value of the **NodeType** element is NewArrayInit (see section 2.4.4).

2.3.2.9 MethodCall

The **MethodCall** type represents a call to either a static or an instance method.

This XML schema fragment specifies the contents of the **MethodCall** complex type.

```
<xs:complexType name="MethodCall">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
```

```
        <xs:element minOccurs="0" name="Object" type="tns1:Expression"/>
        <xs:element name="Method" type="tns1:MethodInfo"/>
        <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **MethodCall** type contains the following child elements:

- **Object:** An optional Expression element that specifies the **Expression** that represents the instance for instance method calls. The **Expression** element MUST be omitted for static method calls.
- **Method:** A MethodInfo element that specifies the **MethodInfo** for the method to be called.
- **Arguments:** An ArrayOfExpression element that specifies a collection of expressions that represent arguments of the called method.

2.3.2.10 Invocation

The **Invocation** type represents an expression that applies a delegate or lambda expression to a list of argument expressions.

This XML schema fragment specifies the contents of the **Invocation** complex type.

```
<xs:complexType name="Invocation">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="Expression" type="tns1:Expression"/>
        <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **Invocation** type contains the following child elements:

- **Expression:** An Expression element that specifies the delegate or lambda expression to be applied.
- **Arguments:** An ArrayOfExpression element that specifies the arguments that the delegate or lambda expression is applied to.

2.3.2.11 Member

The **Member** type represents the field or property to be accessed.

This XML schema fragment specifies the contents of the **Member** complex type.

```
<xs:complexType name="Member">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element minOccurs="0" name="Expression" type="tns1:Expression"/>
        <xs:element name="Member" type="tns1:MemberInfo"/>
      </xs:sequence>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
```

The **Member** type contains the following child elements:

- **Expression:** An optional Expression element that specifies the object that contains the field or property.
- **Member:** A MemberInfo element that specifies the field or property to be accessed.

2.3.2.12 MemberInit

The **MemberInit** type represents calling a constructor and initializing one or more members of the new object.

This XML schema fragment specifies the contents of the **MemberInit** complex type.

```
<xs:complexType name="MemberInit">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="NewExpression" type="tns1:New"/>
        <xs:element name="Bindings" type="tns1:ArrayOfMemberBinding"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **MemberInit** type contains the following child elements:

- **NewExpression:** A New element that specifies the expression that represents the constructor call.
- **Bindings:** An ArrayOfMemberBinding element that specifies the bindings that describe how to initialize the members of the newly created object.

2.3.2.13 ListInit

The **ListInit** type represents a constructor call that has a collection initializer.

This XML schema fragment specifies the contents of the **ListInit** complex type.

```
<xs:complexType name="ListInit">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="NewExpression" type="tns1:New"/>
        <xs:element name="Initializers" type="tns1:ArrayOfElementInit"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **ListInit** type contains the following child elements:

- **NewExpression:** A New element that specifies the expression that contains a call to the constructor of a collection type.

- **Initializers:** An ArrayOfElementInit element that specifies the element initializers that are used to initialize a collection.

2.3.2.14 ParameterReference

The **ParameterReference** type represents a parameter reference expression.

This XML schema fragment specifies the contents of the **ParameterReference** complex type.

```
<xs:complexType name="ParameterReference">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element minOccurs="0" name="ID" type="xs:int"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **ParameterReference** type contains the following child element:

- **ID:** An optional **int** element ([XMLSCHEMA2/2] section 3.3.17) that specifies the identifier for the referenced parameter declaration. The **Parameters** array of the enclosing Lambda MUST define this value.

2.3.2.15 Index

The **Index** type represents indexing a property or array.

This XML schema fragment specifies the contents of the **Index** complex type.

```
<xs:complexType name="Index">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Expression">
      <xs:sequence>
        <xs:element name="Object" type="tns1:Expression"/>
        <xs:element minOccurs="0" name="Indexer" type="tns1:PropertyInfo"/>
        <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **Index** type contains the following child elements:

- **Object:** An Expression element that specifies an object to index.
- **Indexer:** An optional PropertyInfo element that specifies the **PropertyInfo** for the property if the expression represents an indexed property. Otherwise, the **PropertyInfo** element MUST be omitted.
- **Arguments:** An ArrayOfExpression element that specifies the arguments that will be used to index the property or array.

2.3.3 Type

The **Type** type provides a description of a CLR type.

This XML schema fragment specifies the contents of the **Type** complex type.

```
<xs:complexType name="Type">
  <xs:sequence>
    <xs:element name="ID" type="xs:int"/>
    <xs:element minOccurs="0" name="GenericArguments" type="tns1:ArrayOfTypeReference"/>
    <xs:element name="AssemblyRef" type="tns1:AssemblyReference"/>
    <xs:element name="FullName" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

The **Type** type contains the following child elements:

- **ID:** An **int** element ([XMLSCHEMA2/2] section 3.3.17) that specifies an identifier that is used to reference this type description.
- **GenericArguments:** An optional ArrayOfTypeReference element that specifies an array of **TypeReference** objects that represents the type arguments of a generic type. The array is empty when the type is not generic.
- **AssemblyRef:** An AssemblyReference element that specifies the assembly containing the type.
- **FullName:** A **string** element ([XMLSCHEMA2/2] section 3.2.1) that specifies the full name of the type.

2.3.3.1 AnonymousType

The **AnonymousType** type provides a description of an anonymous CLR type. Because there is a one-to-one correspondence between **GenericArguments** and **Properties** in the **AnonymousType** type, the number of items in each sequence MUST match.

This XML schema fragment specifies the contents of the **AnonymousType** complex type.

```
<xs:complexType name="AnonymousType">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:Type">
      <xs:sequence>
        <xs:element minOccurs="0" name="Properties"
type="tns1:ArrayOfAnonymousTypeProperty"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to child elements that are defined on its parent Type type, the **AnonymousType** type contains the following child element:

- **Properties:** An optional ArrayOfAnonymousTypeProperty element that specifies the public instance properties of this anonymous type.

2.3.4 ArrayOfType

The **ArrayOfType** type provides a sequence of Type objects.

This XML schema fragment specifies the contents of the **ArrayOfType** complex type.

```
<xs:complexType name="ArrayOfType">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Type" nillable="true"
type="tns1:Type"/>
  </xs:sequence>
```

```
</xs:complexType>
```

The **ArrayOfType** type contains the following child element:

- **Type:** An optional **Type** element that specifies an element of the sequence.

2.3.5 AnonymousTypeProperty

The **AnonymousTypeProperty** type provides a description of a property of an anonymous type. Note that only the name is required, not the type, because the type is encoded in the **GenericArguments** element of Type.

This XML schema fragment specifies the contents of the **AnonymousTypeProperty** complex type.

```
<xs:complexType name="AnonymousTypeProperty">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

The **AnonymousTypeProperty** type contains the following child element:

- **Name**: A **string** element ([XMLSCHEMA2/2] section 3.2.1) that specifies the name of the property.

2.3.6 ArrayOfAnonymousTypeProperty

The **ArrayOfAnonymousTypeProperty** type provides a sequence of AnonymousTypeProperty objects.

This XML schema fragment specifies the contents of the **ArrayOfAnonymousTypeProperty** complex type.

```
<xs:complexType name="ArrayOfAnonymousTypeProperty">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="AnonymousTypeProperty"
type="tns1:AnonymousTypeProperty"/>
  </xs:sequence>
</xs:complexType>
```

The **AnonymousTypeProperty** type contains the following child element:

- **AnonymousTypeProperty:** An optional **AnonymousTypeProperty** element that specifies the element of the sequence.

2.3.7 TypeReference

The **TypeReference** type provides a reference to a CLR type description.

This XML schema fragment specifies the contents of the **TypeReference** complex type.

```
<xs:complexType name="TypeReference">
  <xs:sequence>
    <xs:element minOccurs="0" name="ID" type="xs:int"/>
  </xs:sequence>
```

```
</xs:complexType>
```

The **TypeReference** type contains the following child element:

- **ID:** An optional **int** element ([XMLSCHEMA2/2] section 3.3.17) that specifies the identifier for the referenced type description. This **ID** MUST be defined in the **Types** element of the ExpressionRoot.

2.3.8 ArrayOfTypeReference

The **ArrayOfTypeReference** type represents a sequence of TypeReference objects.

This XML schema fragment specifies the contents of the **ArrayOfTypeReference** complex type.

```
<xs:complexType name="ArrayOfTypeReference">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="TypeReference"
type="tns1:TypeReference"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfTypeReference** type contains the following child element:

- **TypeReference**: An optional **TypeReference** element that specifies a member of the array.

2.3.9 MemberInfo

The **MemberInfo** type represents the base class for CLR type member references.

This XML schema fragment specifies the contents of the **MemberInfo** complex type.

```
<xs:complexType abstract="true" name="MemberInfo">
  <xs:sequence>
    <xs:element minOccurs="0" name="DeclaringType" type="tns1:TypeReference"/>
    <xs:element minOccurs="0" name="Name" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

The **MemberInfo** type contains the following child elements:

- **DeclaringType:** An optional TypeReference element that specifies the type that declares the current member.
- **Name:** An optional **string** element ([XMLSCHEMA2/2] section 3.2.1) that specifies the name of the current member.

2.3.9.1 MethodInfo

The **MethodInfo** type provides a reference to a method declared on a CLR type.

This XML schema fragment specifies the contents of the **MethodInfo** complex type.

```
<xs:complexType name="MethodInfo">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:MemberInfo">
      <xs:sequence>
```

```
        <xs:element minOccurs="0" name="GenericArguments" type="tns1:ArrayOfTypeReference"/>
        <xs:element name="ParameterTypes" type="tns1:ArrayOfTypeReference"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to child elements defined on its parent MemberInfo type, the **MethodInfo** type contains the following child elements:

- **GenericArguments:** An optional ArrayOfTypeReference element that specifies an array of TypeReference objects that represent the type arguments of a generic method or the type parameters of a generic method definition. For a non-generic method, the array is empty.
- **ParameterTypes:** An **ArrayOfTypeReference** element that specifies an array of **TypeReference** objects that represent the types of parameters of the method.

2.3.9.2 PropertyInfo

The **PropertyInfo** type provides a reference to a CLR type property.

This XML schema fragment specifies the contents of the **PropertyInfo** complex type.

```
<xs:complexType name="PropertyInfo">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:MemberInfo">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **PropertyInfo** type has no elements beyond those that are defined on its parent MemberInfo type.

2.3.9.3 FieldInfo

The **FieldInfo** type provides a reference to a CLR type field.

This XML schema fragment specifies the contents of the **FieldInfo** complex type.

```
<xs:complexType name="FieldInfo">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:MemberInfo">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The **FieldInfo** type has no elements beyond those that are defined on its parent MemberInfo type.

2.3.10 ArrayOfMemberInfo

The **ArrayOfMemberInfo** type provides a sequence of MemberInfo objects.

This XML schema fragment specifies the contents of the **ArrayOfMemberInfo** complex type.

```
<xs:complexType name="ArrayOfMemberInfo">
```

```
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="MemberInfo"
type="tns1:MemberInfo"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfMemberInfo** type contains the following child element:

- **MemberInfo**: An optional **MemberInfo** element that specifies an element of the sequence.

2.3.11 ElementInit

The **ElementInit** type represents an initializer for a single element of an IEnumerable collection [MSDN-IEnumerable].

This XML schema fragment specifies the contents of the **ElementInit** complex type.

```
<xs:complexType name="ElementInit">
  <xs:sequence>
    <xs:element name="AddMethod" type="tns1:MethodInfo"/>
    <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
  </xs:sequence>
</xs:complexType>
```

The **ElementInit** type contains the following child elements:

- **AddMethod:** A MethodInfo element that specifies the instance method that is used to add an element to a collection.
- **Arguments:** An ArrayOfExpression element that specifies the collection of arguments that are passed to a method that adds an element to a collection.

2.3.12 ArrayOfElementInit

The **ArrayOfElementInit** type provides a sequence of ElementInit objects.

This XML schema fragment specifies the contents of the **ArrayOfElementInit** complex type.

```
<xs:complexType name="ArrayOfElementInit">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ElementInit"
type="tns1:ElementInit"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfElementInit** type contains the following child element:

- **ElementInit**: An optional **ElementInit** element that specifies an element of the sequence.

2.3.13 ParameterDeclaration

The **ParameterDeclaration** type provides a declaration of a lambda expression parameter.

This XML schema fragment specifies the contents of the **ParameterDeclaration** complex type.

```
<xs:complexType name="ParameterDeclaration">
  <xs:sequence>
```

```
    <xs:element name="ID" type="xs:int"/>
    <xs:element minOccurs="0" name="Name" type="xs:string"/>
    <xs:element name="Type" type="tns1:TypeReference"/>
  </xs:sequence>
</xs:complexType>
```

The **ParemeterDeclaration** type contains the following child elements:

- **ID:** An **int** element ([XMLSCHEMA2/2] section 3.3.17) that specifies the identifier that is used to reference the current parameter.
- **Name:** An optional **string** element ([XMLSCHEMA2/2] section 3.2.1) that specifies the name of the parameter.
- **Type:** A TypeReference element that specifies the static type of the parameter that this **ParameterDeclaration** represents.

2.3.14 ArrayOfParameterDeclaration

The **ArrayOfParameterDeclaration** type provides a sequence of ParameterDeclaration objects.

This XML schema fragment specifies the contents of the **ArrayOfParameterDeclaration** complex type.

```
<xs:complexType name="ArrayOfParameterDeclaration">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ParameterDeclaration"
type="tns1:ParameterDeclaration"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfParameterDeclaration** type contains the following child element:

- **ParameterDeclaration:** An optional ParameterDeclaration element that specifies an element of the sequence.

2.3.15 MemberBinding

The **MemberBinding** type provides the base type from which the types that represent bindings that are used to initialize members of a newly created object derive.

This XML schema fragment specifies the contents of the **MemberBinding** complex type.

```
<xs:complexType name="MemberBinding" abstract="true">
  <xs:sequence>
    <xs:element name="Member" type="tns1:MemberInfo"/>
  </xs:sequence>
</xs:complexType>
```

The **MemberBinding** type contains the following child element:

- **Member:** A MemberInfo element that specifies the field or property to be initialized.

2.3.15.1 MemberAssignment

The **MemberAssignment** type represents assignment operation for a field or property of an object.

This XML schema fragment specifies the contents of the **MemberAssignment** complex type.

```
<xs:complexType name="MemberAssignment">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:MemberBinding">
      <xs:sequence>
        <xs:element name="Expression" type="tns1:Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to child element that is defined on its parent MemberBinding type, the **MemberAssignment** type contains the following child element:

- **Expression**: An Expression element that specifies the expression to assign to the field or property.

2.3.15.2 MemberListBinding

The **MemberListBinding** type represents initializing the elements of a collection member of a newly created object.

This XML schema fragment specifies the contents of the **MemberListBinding** complex type.

```
<xs:complexType name="MemberListBinding">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:MemberBinding">
      <xs:sequence>
        <xs:element name="Initializers" type="tns1:ArrayOfElementInit"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to child element that is defined on its parent MemberBinding type, the **MemberListBinding** type contains the following child element:

- **Initializers**: An ArrayOfElementInit element that specifies the element initializers that initialize a collection member of a newly created object.

2.3.15.3 MemberMemberBinding

The **MemberMemberBinding** type represents initializing members of a member of a newly created object.

This XML schema fragment specifies the contents of the **MemberMemberBinding** complex type.

```
<xs:complexType name="MemberMemberBinding">
  <xs:complexContent mixed="false">
    <xs:extension base="tns1:MemberBinding">
      <xs:sequence>
        <xs:element name="Bindings" type="tns1:ArrayOfMemberBinding"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

In addition to child element that is defined on its parent MemberBinding type, the **MemberMemberBinding** type contains the following child element:

- **Bindings:** An ArrayOfMemberBinding element that specifies the bindings that describe how to initialize the members of a member.

2.3.16 ArrayOfMemberBinding

The **ArrayOfMemberBinding** type provides a sequence of **MemberBinding** objects.

This XML schema fragment specifies the contents of the **ArrayOfMemberBinding** complex type.

```
<xs:complexType name="ArrayOfMemberBinding">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="MemberBinding"
type="tns1:MemberBinding"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfMemberBinding** type contains the following child element:

- **MemberBinding**: An optional MemberBinding element that specifies an element of the sequence.

2.3.17 ConstructorInfo

The **ConstructorInfo** type provides a reference to a constructor that is declared on a CLR type.

This XML schema fragment specifies the contents of the **ConstructorInfo** complex type.

```
<xs:complexType name="ConstructorInfo">
  <xs:sequence>
    <xs:element name="DeclaringType" type="tns1:TypeReference"/>
    <xs:element name="ParameterTypes" type="tns1:ArrayOfTypeReference"/>
  </xs:sequence>
</xs:complexType>
```

The **ConstructorInfo** type contains the following child elements:

- **DeclaringType:** A TypeReference element that specifies the type that declares the current constructor.
- **ParameterTypes:** An ArrayOfTypeReference element that specifies an array of **TypeReference** objects that represent the types of parameters of the constructor.

2.3.18 AssemblyReference

The **AssemblyReference** type provides a reference to a CLR assembly description.

This XML schema fragment specifies the contents of the **AssemblyReference** complex type.

```
<xs:complexType name="AssemblyReference">
  <xs:sequence>
    <xs:element name="ID" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

The **AssemblyReference** type contains the following child element:

- **ID:** An **int** element ([XMLSCHEMA2/2] section 3.3.17) that specifies an identifier for the referenced assembly description. This **ID** MUST be defined in the **Assemblies** element of the ExpressionRoot.

2.3.19 Assembly

The **Assembly** type provides a description of a CLR assembly.

This XML schema fragment specifies the contents of the **Assembly** complex type.

```
<xs:complexType name="Assembly">
  <xs:sequence>
    <xs:element name="FullName" type="xs:string"/>
    <xs:element name="ID" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

The **Assembly** type contains the following child elements:

- **FullName:** A **string** element ([XMLSCHEMA2/2] section 3.2.1) that specifies the display name of the assembly. See [MSDN-AsmblyNameClass] for a description of the format of the display name of an assembly.
- **ID:** An **int** element ([XMLSCHEMA2/2] section 3.3.17) that specifies the identifier that is used to reference this assembly description.

2.3.20 ArrayOfAssembly

The **ArrayOfAssembly** type provides a sequence of **Assembly** objects.

This XML schema fragment specifies the contents of the **ArrayOfAssembly** complex type.

```
<xs:complexType name="ArrayOfAssembly">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Assembly" type="tns1:Assembly"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfAssembly** type contains the following child element:

- **Assembly**: An optional Assembly element that specifies an element of the sequence.

2.3.21 ArrayOfExpression

The **ArrayOfExpression** type provides a sequence of **Expression** objects.

This XML schema fragment specifies the contents of the **ArrayOfExpression** complex type.

```
<xs:complexType name="ArrayOfExpression">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Expression"
type="tns1:Expression"/>
  </xs:sequence>
</xs:complexType>
```

The **ArrayOfExpression** type contains the following child element:

- **Expression:** An optional Expression element that specifies an element of the sequence.

2.4 Simple Types

2.4.1 TypeBinaryNodeType

The **TypeBinaryNodeType** type describes the node types for a TypeBinary expression.

This XML schema fragment specifies the contents of the **TypeBinaryNodeType** simple type.

```
<xs:simpleType name="TypeBinaryNodeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="TypeEqual"/>
    <xs:enumeration value="TypeIs"/>
  </xs:restriction>
</xs:simpleType>
```

The following table defines the values for the **TypeBinaryNodeType** type.

Value	Meaning
TypeEqual	An exact type test.
TypeIs	A type test, such as 'obj is SampleType' in C#.

2.4.2 UnaryNodeType

The **UnaryNodeType** type describes the node types for a Unary expression.

This XML schema fragment specifies the contents of the **UnaryNodeType** simple type.

```
<xs:simpleType name="UnaryNodeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ArrayLength"/>
    <xs:enumeration value="Convert"/>
    <xs:enumeration value="ConvertChecked"/>
    <xs:enumeration value="Negate"/>
    <xs:enumeration value="NegateChecked"/>
    <xs:enumeration value="Not"/>
    <xs:enumeration value="Quote"/>
    <xs:enumeration value="TypeAs"/>
    <xs:enumeration value="UnaryPlus"/>
    <xs:enumeration value="IsTrue"/>
    <xs:enumeration value="IsFalse"/>
    <xs:enumeration value="OnesComplement"/>
  </xs:restriction>
</xs:simpleType>
```

The following table defines the values for the **UnaryNodeType** type.

Value	Meaning
ArrayLength	An operation that obtains the length of a one-dimensional array.
Convert	A cast or conversion operation, such as '(SampleType)obj' in C#. For a numeric conversion, if the converted value is too large for the destination type, no exception is thrown.

Value	Meaning
ConvertChecked	A cast or conversion operation, such as '(SampleType)obj' in C#. For a numeric conversion, if the converted value does not fit the destination type, an exception is thrown.
Negate	An arithmetic negation operation, such as '(-a)', that does not have overflow checking.
NegateChecked	An arithmetic negation operation, such as' (-a)', that has overflow checking.
Not	A bitwise complement or logical negation operation. In C#, it is equivalent to '(~a)' for integral types and to '(!a)' for Boolean values.
Quote	An expression that has a constant value of type Expression. A Quote node can contain references to parameters that are defined in the context of the expression it represents.
TypeAs	An explicit reference or boxing conversion operation, such as '(obj as SampleType)' in C#, in which null is supplied if the conversion fails.
UnaryPlus	A unary plus operation, such as '(+a)'. The result of a predefined unary plus operation is the value of the operand. However, user-defined implementations might have unusual results.
IsTrue	A true condition value.
IsFalse	A false condition value.
OnesComplement	A ones complement operation, such as '(~a)' in C#.

2.4.3 BinaryNodeType

The **BinaryNodeType** type describes the node types for a Binary expression.

This XML schema fragment specifies the contents of the **BinaryNodeType** simple type.

```
<xs:simpleType name="BinaryNodeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Add"/>
    <xs:enumeration value="AddChecked"/>
    <xs:enumeration value="And"/>
    <xs:enumeration value="AndAlso"/>
    <xs:enumeration value="ArrayIndex"/>
    <xs:enumeration value="Coalesce"/>
    <xs:enumeration value="Divide"/>
    <xs:enumeration value="Equal"/>
    <xs:enumeration value="ExclusiveOr"/>
    <xs:enumeration value="GreaterThan"/>
    <xs:enumeration value="GreaterThanOrEqual"/>
    <xs:enumeration value="LeftShift"/>
    <xs:enumeration value="LessThan"/>
    <xs:enumeration value="LessThanOrEqual"/>
    <xs:enumeration value="Modulo"/>
    <xs:enumeration value="Multiply"/>
    <xs:enumeration value="MultiplyChecked"/>
    <xs:enumeration value="NotEqual"/>
    <xs:enumeration value="Or"/>
    <xs:enumeration value="OrElse"/>
    <xs:enumeration value="Power"/>
    <xs:enumeration value="RightShift"/>
    <xs:enumeration value="Subtract"/>
    <xs:enumeration value="SubtractChecked"/>
  </xs:restriction>
</xs:simpleType>
```

The following table defines the values for the **BinaryNodeType** type.

Value	Meaning
Add	An addition operation, such as 'a + b', that does not have overflow checking for numeric operands.
AddChecked	An addition operation, such as '(a + b)', that has overflow checking for numeric operands.
And	A bitwise or logical AND operation, such as '(a & b)' in C#.
AndAlso	A conditional AND operation that evaluates the second operand only if the first operand evaluates to true. It corresponds to '(a && b)' in C#.
ArrayIndex	An indexing operation in a one-dimensional array, such as 'array[index]' in C#.
Coalesce	A node that represents a null coalescing operation, such as '(a ?? b)' in C#.
Divide	A division operation, such as '(a / b)', for numeric operands.
Equal	A node that represents an equality comparison, such as '(a == b)' in C#.
ExclusiveOr	A bitwise or logical XOR operation, such as '(a ^ b)' in C#.
GreaterThan	A "greater than" comparison, such as '(a > b)'.
GreaterThanOrEqual	A "greater than or equal to" comparison, such as '(a >= b)'.
LeftShift	A bitwise left-shift operation, such as '(a << b)'.
LessThan	A "less than" comparison, such as '(a < b)'.
LessThanOrEqual	A "less than or equal to" comparison, such as '(a <= b)'.
Modulo	An arithmetic remainder operation, such as '(a % b)' in C#.
Multiply	A multiplication operation, such as '(a * b)', that does not have overflow checking for numeric operands.
MultiplyChecked	A multiplication operation, such as '(a * b)', that has overflow checking for numeric operands.
NotEqual	An inequality comparison, such as '(a != b)' in C#.
Or	A bitwise or logical OR operation, such as '(a | b)' in C#.
OrElse	A short-circuiting conditional OR operation, such as '(a || b)' in C#.
Power	A mathematical operation that raises a number to a power, such as '(a ^ b)' in Visual Basic.
RightShift	A bitwise right-shift operation, such as '(a >> b)'.
Subtract	A subtraction operation, such as '(a - b)', that does not have overflow checking for numeric operands.
SubtractChecked	An arithmetic subtraction operation, such as '(a - b)', that has overflow checking for numeric operands.

2.4.4 NewArrayNodeType

The **NewArrayNodeType** type describes the node types for a NewArray expression.

This XML schema fragment specifies the contents of the **NewArrayNodeType** simple type.

```
<xs:simpleType name="NewArrayNodeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="NewArrayInit"/>
    <xs:enumeration value="NewArrayBounds"/>
  </xs:restriction>
</xs:simpleType>
```

The following table defines the values for the **NewArrayNodeType** type.

Value	Meaning
NewArrayInit	An operation that creates a new one-dimensional array and initializes it from a list of elements, such as 'new SampleType[]{a, b, c}' in C#.
NewArrayBounds	An operation that creates a new array in which the bounds for each dimension are specified, such as 'new SampleType[dim1, dim2]' in C#.

3 Structure Examples

3.1 Array Initialization, Application of Extension Method, Lambda, and Primitive Constants

The following C# expression example creates a new integer array, populates that array, and then produces an array with the square of its contents.

```
(new int[] { 1, 2, 3 }).Select(x => x * x)
```

The following is a valid serialization of this expression.

```
<?xml version="1.0" encoding="utf-16"?>
<ExpressionRoot xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression">
  <Definition i:type="MethodCall">
    <Method>
      <DeclaringType>
        <ID>0</ID>
      </DeclaringType>
      <Name>Select</Name>
      <GenericArguments>
        <TypeReference>
          <ID>2</ID>
        </TypeReference>
        <TypeReference>
          <ID>2</ID>
        </TypeReference>
      </GenericArguments>
      <ParameterTypes>
        <TypeReference>
          <ID>1</ID>
        </TypeReference>
        <TypeReference>
          <ID>3</ID>
        </TypeReference>
      </ParameterTypes>
    </Method>
    <Arguments>
      <Expression i:type="NewArray">
        <NodeType>NewArrayInit</NodeType>
        <ElementType>
          <ID>2</ID>
        </ElementType>
        <Expressions>
          <Expression i:type="PrimitiveConstant">
            <Value xmlns:d7p1="http://www.w3.org/2001/XMLSchema" i:type="d7p1:int">1</Value>
          </Expression>
          <Expression i:type="PrimitiveConstant">
            <Value xmlns:d7p1="http://www.w3.org/2001/XMLSchema" i:type="d7p1:int">2</Value>
          </Expression>
          <Expression i:type="PrimitiveConstant">
            <Value xmlns:d7p1="http://www.w3.org/2001/XMLSchema" i:type="d7p1:int">3</Value>
          </Expression>
        </Expressions>
      </Expression>
      <Expression i:type="Lambda">
        <Parameters>
          <ParameterDeclaration>
            <ID>0</ID>
            <Name>x</Name>
            <Type>
              <ID>2</ID>
            </Type>
```

```
        </ParameterDeclaration>
      </Parameters>
      <Body i:type="Binary">
        <NodeType>Multiply</NodeType>
        <Left i:type="ParameterReference">
          <ID>0</ID>
        </Left>
        <Right i:type="ParameterReference">
          <ID>0</ID>
        </Right>
      </Body>
      <Type>
        <ID>3</ID>
      </Type>
    </Expression>
  </Arguments>
</Definition>
<Types>
  <Type>
    <ID>0</ID>
    <AssemblyRef>
      <ID>0</ID>
    </AssemblyRef>
    <FullName>System.Linq.Enumerable</FullName>
  </Type>
  <Type>
    <ID>2</ID>
    <AssemblyRef>
      <ID>1</ID>
    </AssemblyRef>
    <FullName>System.Int32</FullName>
  </Type>
  <Type>
    <ID>1</ID>
    <GenericArguments>
      <TypeReference>
        <ID>2</ID>
      </TypeReference>
    </GenericArguments>
    <AssemblyRef>
      <ID>1</ID>
    </AssemblyRef>
    <FullName>System.Collections.Generic.IEnumerable`1</FullName>
  </Type>
  <Type>
    <ID>3</ID>
    <GenericArguments>
      <TypeReference>
        <ID>2</ID>
      </TypeReference>
      <TypeReference>
        <ID>2</ID>
      </TypeReference>
    </GenericArguments>
    <AssemblyRef>
      <ID>1</ID>
    </AssemblyRef>
    <FullName>System.Func`2</FullName>
  </Type>
</Types>
<Assemblies>
  <Assembly>
    <FullName>System.Core, Version=4.0.0.0, Culture=neutral,
              PublicKeyToken=b77a5c561934e089</FullName>
    <ID>0</ID>
  </Assembly>
  <Assembly>
    <FullName>mscorlib, Version=4.0.0.0, Culture=neutral,
              PublicKeyToken=b77a5c561934e089</FullName>
```

```
      <ID>1</ID>
    </Assembly>
  </Assemblies>
</ExpressionRoot>
```

3.2 Method Invocation and Binary Operators

The following C# expression example makes multiple calls to the method "Method" and applies the binary "+" operator.

```
Method(1, "string argument", Guid.NewGuid()) + Method(Int32.MinValue, string.Empty,
Guid.Empty)
```

The following is a valid serialization of this expression.

```
<?xml version="1.0" encoding="utf-16"?>
<ExpressionRoot xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression">
  <Definition i:type="Binary">
    <NodeType>Add</NodeType>
    <Left i:type="MethodCall">
      <Method>
        <DeclaringType>
          <ID>0</ID>
        </DeclaringType>
        <Name>Method</Name>
        <ParameterTypes>
          <TypeReference>
            <ID>1</ID>
          </TypeReference>
          <TypeReference>
            <ID>2</ID>
          </TypeReference>
          <TypeReference>
            <ID>3</ID>
          </TypeReference>
        </ParameterTypes>
      </Method>
      <Arguments>
        <Expression i:type="PrimitiveConstant">
          <Value xmlns:d6p1="http://www.w3.org/2001/XMLSchema" i:type="d6p1:int">1</Value>
        </Expression>
        <Expression i:type="PrimitiveConstant">
          <Value xmlns:d6p1="http://www.w3.org/2001/XMLSchema" i:type="d6p1:string">string
argument</Value>
        </Expression>
        <Expression i:type="MethodCall">
          <Method>
            <DeclaringType>
              <ID>3</ID>
            </DeclaringType>
            <Name>NewGuid</Name>
            <ParameterTypes />
          </Method>
          <Arguments />
        </Expression>
      </Arguments>
    </Left>
    <Right i:type="MethodCall">
      <Method>
        <DeclaringType>
          <ID>0</ID>
        </DeclaringType>
        <Name>Method</Name>
```

```
          <ParameterTypes>
            <TypeReference>
              <ID>1</ID>
            </TypeReference>
            <TypeReference>
              <ID>2</ID>
            </TypeReference>
            <TypeReference>
              <ID>3</ID>
            </TypeReference>
          </ParameterTypes>
        </Method>
        <Arguments>
          <Expression i:type="PrimitiveConstant">
            <Value xmlns:d6p1="http://www.w3.org/2001/XMLSchema" i:type="d6p1:int">-
2147483648</Value>
          </Expression>
          <Expression i:type="Member">
            <Member i:type="FieldInfo">
              <DeclaringType>
                <ID>2</ID>
              </DeclaringType>
              <Name>Empty</Name>
            </Member>
          </Expression>
          <Expression i:type="Member">
            <Member i:type="FieldInfo">
              <DeclaringType>
                <ID>3</ID>
              </DeclaringType>
              <Name>Empty</Name>
            </Member>
          </Expression>
        </Arguments>
      </Right>
    </Definition>
    <Types>
      <Type>
        <ID>0</ID>
        <AssemblyRef>
          <ID>0</ID>
        </AssemblyRef>
        <FullName>SerializerSamples.Program</FullName>
      </Type>
      <Type>
        <ID>1</ID>
        <AssemblyRef>
          <ID>1</ID>
        </AssemblyRef>
        <FullName>System.Int32</FullName>
      </Type>
      <Type>
        <ID>2</ID>
        <AssemblyRef>
          <ID>1</ID>
        </AssemblyRef>
        <FullName>System.String</FullName>
      </Type>
      <Type>
        <ID>3</ID>
        <AssemblyRef>
          <ID>1</ID>
        </AssemblyRef>
        <FullName>System.Guid</FullName>
      </Type>
    </Types>
    <Assemblies>
      <Assembly>
        <FullName>SerializerSamples, Version=1.0.0.0, Culture=neutral,
```

```
                PublicKeyToken=null</FullName>
      <ID>0</ID>
    </Assembly>
    <Assembly>
      <FullName>mscorlib, Version=4.0.0.0, Culture=neutral,
                PublicKeyToken=b77a5c561934e089</FullName>
      <ID>1</ID>
    </Assembly>
  </Assemblies>
</ExpressionRoot>
```

3.3 Generic Types and Method Invocation

This C# expression example creates a new instance of a tuple that is parameterized to contain an integer and a string, fills the expression with an integer and a string, and then retrieves the first item.

```
new Tuple<int, string>(0, "demo").Item1
```

The following is a valid serialization of this expression.

```
<?xml version="1.0" encoding="utf-16"?>
<ExpressionRoot xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression">
  <Definition i:type="Member">
    <Expression i:type="New">
      <Constructor>
        <DeclaringType>
          <ID>0</ID>
        </DeclaringType>
        <ParameterTypes>
          <TypeReference>
            <ID>1</ID>
          </TypeReference>
          <TypeReference>
            <ID>2</ID>
          </TypeReference>
        </ParameterTypes>
      </Constructor>
      <Arguments>
        <Expression i:type="PrimitiveConstant">
          <Value xmlns:d6p1="http://www.w3.org/2001/XMLSchema" i:type="d6p1:int">0</Value>
        </Expression>
        <Expression i:type="PrimitiveConstant">
          <Value xmlns:d6p1="http://www.w3.org/2001/XMLSchema"
i:type="d6p1:string">demo</Value>
        </Expression>
      </Arguments>
    </Expression>
    <Member i:type="PropertyInfo">
      <DeclaringType>
        <ID>0</ID>
      </DeclaringType>
      <Name>Item1</Name>
    </Member>
  </Definition>
  <Types>
    <Type>
      <ID>1</ID>
      <AssemblyRef>
        <ID>0</ID>
      </AssemblyRef>
      <FullName>System.Int32</FullName>
    </Type>
    <Type>
```

```
      <ID>2</ID>
      <AssemblyRef>
        <ID>0</ID>
      </AssemblyRef>
      <FullName>System.String</FullName>
    </Type>
    <Type>
      <ID>0</ID>
      <GenericArguments>
        <TypeReference>
          <ID>1</ID>
        </TypeReference>
        <TypeReference>
          <ID>2</ID>
        </TypeReference>
      </GenericArguments>
      <AssemblyRef>
        <ID>0</ID>
      </AssemblyRef>
      <FullName>System.Tuple`2</FullName>
    </Type>
  </Types>
  <Assemblies>
    <Assembly>
      <FullName>mscorlib, Version=4.0.0.0, Culture=neutral,
                PublicKeyToken=b77a5c561934e089</FullName>
      <ID>0</ID>
    </Assembly>
  </Assemblies>
</ExpressionRoot>
```

4 Security

4.1 Security Considerations for Implementers

None.

4.2 Index of Security Fields

None.

5 Appendix A: Full XML Schema

For ease of implementation, the following is the full XML schema for this protocol.

```
<xs:schema
xmlns:tns1="http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression"
elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/ComplexEventProcessing/2011/10/LinqExpression">
  <xs:element name="ExpressionRoot" type="tns1:ExpressionRoot"/>
  <xs:complexType name="ExpressionRoot">
    <xs:sequence>
      <xs:element name="Definition" type="tns1:Expression"/>
      <xs:element name="Types" type="tns1:ArrayOfType"/>
      <xs:element name="Assemblies" type="tns1:ArrayOfAssembly"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Expression">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="ExpressionConstant">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Constant">
        <xs:sequence>
          <xs:element name="Value" type="tns1:Expression"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="TypeBinary">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element name="NodeType" type="tns1:TypeBinaryNodeType"/>
          <xs:element name="TypeOperand" type="tns1:TypeReference"/>
          <xs:element name="Expression" type="tns1:Expression"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType name="TypeBinaryNodeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="TypeEqual"/>
      <xs:enumeration value="TypeIs"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="TypeReference">
    <xs:sequence>
      <xs:element minOccurs="0" name="ID" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Unary">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element name="NodeType" type="tns1:UnaryNodeType"/>
          <xs:element name="Type" type="tns1:TypeReference"/>
          <xs:element name="Operand" type="tns1:Expression"/>
          <xs:element minOccurs="0" name="Method" type="tns1:MethodInfo"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType name="UnaryNodeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ArrayLength"/>
```

```
      <xs:enumeration value="Convert"/>
      <xs:enumeration value="ConvertChecked"/>
      <xs:enumeration value="Negate"/>
      <xs:enumeration value="NegateChecked"/>
      <xs:enumeration value="Not"/>
      <xs:enumeration value="Quote"/>
      <xs:enumeration value="TypeAs"/>
      <xs:enumeration value="UnaryPlus"/>
      <xs:enumeration value="IsTrue"/>
      <xs:enumeration value="IsFalse"/>
      <xs:enumeration value="OnesComplement"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="MethodInfo">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:MemberInfo">
        <xs:sequence>
          <xs:element minOccurs="0" name="GenericArguments"
type="tns1:ArrayOfTypeReference"/>
          <xs:element name="ParameterTypes" type="tns1:ArrayOfTypeReference"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType abstract="true" name="MemberInfo">
    <xs:sequence>
      <xs:element minOccurs="0" name="DeclaringType" type="tns1:TypeReference"/>
      <xs:element minOccurs="0" name="Name" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PropertyInfo">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:MemberInfo">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="FieldInfo">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:MemberInfo">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ArrayOfTypeReference">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="TypeReference"
type="tns1:TypeReference"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Constant" abstract="true">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element minOccurs="0" name="Type" type="tns1:TypeReference"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="MethodInfoConstant">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Constant">
        <xs:sequence>
          <xs:element name="Value" type="tns1:MethodInfo"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DelegateConstant">
```

```
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Constant">
          <xs:sequence>
            <xs:element minOccurs="0" name="Target" type="tns1:Constant"/>
            <xs:element name="Method" type="tns1:MethodInfo"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="PrimitiveConstant">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Constant">
          <xs:sequence>
            <xs:element minOccurs="0" nillable="true" name="Value" type="xs:anyType"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TypeConstant">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Constant">
          <xs:sequence>
            <xs:element name="Value" type="tns1:TypeReference"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SerializedConstant">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Constant">
          <xs:sequence>
            <xs:element minOccurs="0" name="Value" type="xs:string"/>
            <xs:element name="ValueType" type="tns1:TypeReference"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="Invocation">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Expression">
          <xs:sequence>
            <xs:element name="Expression" type="tns1:Expression"/>
            <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ArrayOfExpression">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="Expression"
  type="tns1:Expression"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Binary">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Expression">
          <xs:sequence>
            <xs:element name="NodeType" type="tns1:BinaryNodeType"/>
            <xs:element name="Left" type="tns1:Expression"/>
            <xs:element name="Right" type="tns1:Expression"/>
            <xs:element minOccurs="0" name="IsLiftedToNull" type="xs:boolean"/>
            <xs:element minOccurs="0" name="Method" type="tns1:MethodInfo"/>
            <xs:element minOccurs="0" name="Conversion" type="tns1:Lambda"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:simpleType name="BinaryNodeType">
      <xs:restriction base="xs:string">
```

```
        <xs:enumeration value="Add"/>
        <xs:enumeration value="AddChecked"/>
        <xs:enumeration value="And"/>
        <xs:enumeration value="AndAlso"/>
        <xs:enumeration value="ArrayIndex"/>
        <xs:enumeration value="Coalesce"/>
        <xs:enumeration value="Divide"/>
        <xs:enumeration value="Equal"/>
        <xs:enumeration value="ExclusiveOr"/>
        <xs:enumeration value="GreaterThan"/>
        <xs:enumeration value="GreaterThanOrEqual"/>
        <xs:enumeration value="LeftShift"/>
        <xs:enumeration value="LessThan"/>
        <xs:enumeration value="LessThanOrEqual"/>
        <xs:enumeration value="Modulo"/>
        <xs:enumeration value="Multiply"/>
        <xs:enumeration value="MultiplyChecked"/>
        <xs:enumeration value="NotEqual"/>
        <xs:enumeration value="Or"/>
        <xs:enumeration value="OrElse"/>
        <xs:enumeration value="Power"/>
        <xs:enumeration value="RightShift"/>
        <xs:enumeration value="Subtract"/>
        <xs:enumeration value="SubtractChecked"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="Lambda">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Expression">
          <xs:sequence>
            <xs:element name="Parameters" type="tns1:ArrayOfParameterDeclaration"/>
            <xs:element name="Body" type="tns1:Expression"/>
            <xs:element minOccurs="0" name="Type" type="tns1:TypeReference"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ArrayOfParameterDeclaration">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="ParameterDeclaration"
  type="tns1:ParameterDeclaration"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ParameterDeclaration">
      <xs:sequence>
        <xs:element name="ID" type="xs:int"/>
        <xs:element minOccurs="0" name="Name" type="xs:string"/>
        <xs:element name="Type" type="tns1:TypeReference"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Conditional">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Expression">
          <xs:sequence>
            <xs:element name="Test" type="tns1:Expression"/>
            <xs:element name="IfTrue" type="tns1:Expression"/>
            <xs:element name="IfFalse" type="tns1:Expression"/>
            <xs:element minOccurs="0" name="Type" type="tns1:TypeReference"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ListInit">
      <xs:complexContent mixed="false">
        <xs:extension base="tns1:Expression">
          <xs:sequence>
            <xs:element name="NewExpression" type="tns1:New"/>
            <xs:element name="Initializers" type="tns1:ArrayOfElementInit"/>
          </xs:sequence>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="New">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:choice minOccurs="1" maxOccurs="1">
            <xs:element minOccurs="1" name="Type" type="tns1:TypeReference"/>
            <xs:sequence>
              <xs:element minOccurs="1" name="Constructor" type="tns1:ConstructorInfo"/>
              <xs:element minOccurs="0" name="Arguments" type="tns1:ArrayOfExpression"/>
              <xs:element minOccurs="0" name="Members" type="tns1:ArrayOfMemberInfo"/>
            </xs:sequence>
          </xs:choice>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ConstructorInfo">
    <xs:sequence>
      <xs:element name="DeclaringType" type="tns1:TypeReference"/>
      <xs:element name="ParameterTypes" type="tns1:ArrayOfTypeReference"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ArrayOfMemberInfo">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="MemberInfo"
type="tns1:MemberInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ParameterReference">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element minOccurs="0" name="ID" type="xs:int"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Index">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element name="Object" type="tns1:Expression"/>
          <xs:element minOccurs="0" name="Indexer" type="tns1:PropertyInfo"/>
          <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Member">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element minOccurs="0" name="Expression" type="tns1:Expression"/>
          <xs:element name="Member" type="tns1:MemberInfo"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="MemberInit">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element name="NewExpression" type="tns1:New"/>
          <xs:element name="Bindings" type="tns1:ArrayOfMemberBinding"/>
        </xs:sequence>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ArrayOfMemberBinding">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="MemberBinding"
type="tns1:MemberBinding"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MemberBinding" abstract="true">
    <xs:sequence>
      <xs:element name="Member" type="tns1:MemberInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MemberAssignment">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:MemberBinding">
        <xs:sequence>
          <xs:element name="Expression" type="tns1:Expression"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="MemberListBinding">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:MemberBinding">
        <xs:sequence>
          <xs:element name="Initializers" type="tns1:ArrayOfElementInit"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ArrayOfElementInit">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="ElementInit"
type="tns1:ElementInit"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ElementInit">
    <xs:sequence>
      <xs:element name="AddMethod" type="tns1:MethodInfo"/>
      <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MemberMemberBinding">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:MemberBinding">
        <xs:sequence>
          <xs:element name="Bindings" type="tns1:ArrayOfMemberBinding"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="MethodCall">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element minOccurs="0" name="Object" type="tns1:Expression"/>
          <xs:element name="Method" type="tns1:MethodInfo"/>
          <xs:element name="Arguments" type="tns1:ArrayOfExpression"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="NewArray">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Expression">
        <xs:sequence>
          <xs:element name="NodeType" type="tns1:NewArrayNodeType"/>
          <xs:element name="ElementType" type="tns1:TypeReference"/>
```

```
          <xs:element name="Expressions" type="tns1:ArrayOfExpression"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType name="NewArrayNodeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="NewArrayInit"/>
      <xs:enumeration value="NewArrayBounds"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ArrayOfType">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Type" nillable="true"
type="tns1:Type"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Type">
    <xs:sequence>
      <xs:element name="ID" type="xs:int"/>
      <xs:element minOccurs="0" name="GenericArguments" type="tns1:ArrayOfTypeReference"/>
      <xs:element name="AssemblyRef" type="tns1:AssemblyReference"/>
      <xs:element name="FullName" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AnonymousType">
    <xs:complexContent mixed="false">
      <xs:extension base="tns1:Type">
        <xs:sequence>
          <xs:element minOccurs="0" name="Properties"
type="tns1:ArrayOfAnonymousTypeProperty"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ArrayOfAnonymousTypeProperty">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="AnonymousTypeProperty"
type="tns1:AnonymousTypeProperty"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AnonymousTypeProperty">
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AssemblyReference">
    <xs:sequence>
      <xs:element name="ID" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ArrayOfAssembly">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Assembly" type="tns1:Assembly"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Assembly">
    <xs:sequence>
      <xs:element name="FullName" type="xs:string"/>
      <xs:element name="ID" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

6 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

- Microsoft SQL Server 2012
- Microsoft SQL Server 2014
- Microsoft SQL Server 2016

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
6 Appendix B: Product Behavior	Added SQL Server 2016 to the list of applicable products.	Y	Content update.

8 Index