[MS-DSDG-Diff]:

DataSet DiffGram Structure

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
3/5/2010	0.1	Major	First release.
4/21/2010	0.1.1	Editorial	Changed language and formatting in the technical content.
6/4/2010	0.1.2	Editorial	Changed language and formatting in the technical content.
9/3/2010	0.1.3	Editorial	Changed language and formatting in the technical content.
2/9/2011	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
7/7/2011	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
11/3/2011	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
1/19/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
2/23/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
3/27/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
5/24/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
6/29/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
10/23/2012	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
3/26/2013	0.1.3	None	No changes to the meaning, language, or formatting of the technical content.
6/11/2013	1.0	Major	Updated and revised the technical content.
8/8/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
12/5/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2014	1.0	None	No changes to the meaning, language, or formatting of the technical content.
5/20/2014	1.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	2.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
5/10/2016	2.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The DataSet DiffGram structure applies to a DiffGram that is an XML representation of a **DataSet** object. The **DiffGram** structure is useful for serializing schema and data for transmission over a network such as for use with a **web service**. Producers and consumers can use the **DiffGram** structure to encapsulate both the schema and the data of the **DataSet**.

Sections 1.7 and 2 of this specification are normative ~~and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119].~~ All other sections and examples in this specification are informative.

1.1 Glossary

~~The~~ This document uses the following terms ~~are specific to this document~~:

~~**.NET Framework**: An integral Windows component that supports building and running applications and XML web services. The Microsoft .NET Framework has two main components: the common language runtime and the .NET Framework class library. For more information about the .NET Framework, see [MSDN-.NET-FRAMEWORK]. The following versions of the .NET Framework are available in the following released Windows products or as supplemental software. Microsoft .NET Framework 1.0: Windows NT 4.0 operating system, Microsoft Windows 98 operating system, Windows 2000 operating system, Windows Millennium Edition operating system, Windows XP operating system, and Windows Server 2003 operating system. Microsoft .NET Framework 1.1: Windows 98, Windows 2000, Windows Millennium Edition, Windows XP, Windows Server 2003, Windows Server 2003 R2 operating system, Windows Vista operating system, and Windows Server 2008 operating system. Microsoft .NET Framework 2.0: Windows 98, Windows 2000, Windows Millennium Edition, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7 operating system, Windows Server 2008 R2 operating system, Windows 8 operating system, Windows Server 2012 operating system, Windows 8.1 operating system, Windows Server 2012 R2 operating system, Windows 10 operating system, and Windows Server 2016 Technical Preview operating system. Microsoft .NET Framework 3.0: Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, and Windows Server 2016 Technical Preview. Microsoft .NET Framework 3.5: Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, and Windows Server 2016 Technical Preview. Microsoft .NET Framework 4.0: Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, and Windows Server 2016 Technical Preview. Microsoft .NET Framework 4.5: Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, and Windows 10. Microsoft .NET Framework 4.6: Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, and Windows 10.~~

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

child element: In an **XML document**, an element that is subordinate to and is contained by another element, which is referred to as the parent element.

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

in-memory: A memory model in which multidimensional aggregates are precomputed and stored but not written out on disk. Instead, they are stored in computer memory.

locale: An identifier, as specified in [MS-LCID], that specifies preferences related to language. These preferences indicate how dates and times are to be formatted, how items are to be sorted alphabetically, how strings are to be compared, and so on.

Persistent Storage: Nonvolatile storage mediums, such as magnetic disks, tapes, and optical disks.

primary key: A field or set of fields that uniquely identifies each record in a table. A primary key cannot contain a null value.

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].

~~**schema**: A container that defines a namespace that describes the scope of EDM types. All EDM types are contained within some namespace.~~

serialize: The process of taking an in-memory data structure, flat or otherwise, and turning it into a flat stream of bytes. See also marshal.

SOAP: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. **SOAP** uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [SOAP1.2-1/2003].

SOAP envelope: A container for SOAP message information and the root element of a **SOAP** document. See [SOAP1.2-1/2007] section 5.1 for more information.

User Datagram Protocol (UDP): The connectionless protocol within TCP/IP that corresponds to the transport layer in the ISO/OSI reference model.

~~**web service**: A unit of application logic that provides data and services to other applications and can be called by using standard Internet transport protocols such as **HTTP**, Simple Mail Transfer Protocol (SMTP), or File Transfer Protocol (FTP). Web services can perform functions that range from simple requests to complicated business processes.~~

web service: A software entity that responds to SOAP messages ([SOAP1.1],.[WSDL]).

web service method: A procedure that is exposed to web service clients as an operation that can be called on the web service. Also referred to as web method.

XML attribute: A name/value pair, separated by an equal sign (=) and included in a tagged element, that modifies features of an element. All XML attribute values are stored as strings enclosed in quotation marks.

XML document: A document object that is well formed, as described in [~~XML~~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 element: An XML structure that typically consists of a start tag, an end tag, and the information between those tags. Elements can have attributes (1) and can contain other elements.

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 (XSD): A language that defines the elements, attributes, namespaces, and data types for XML documents as defined by [XMLSCHEMA1/2] and [W3C-XSD] standards. An XML schema uses XML syntax for its language.

~~**XML schema definition (XSD)**: The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring XML schemas.~~

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.

[ECMA-335:2006] ECMA, "Common Language Infrastructure (CLI) Partitions I to VI", 4th edition, Standard ECMA-335, June 2006, http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/Ecma-335%204th%20edition%20June%202006.pdf

[MC-ADONETDSSS] Microsoft Corporation, "ADO.NET DataSet Structure Schema", http://schemas.microsoft.com/2003/07/msdata.xsd

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-NRBF] Microsoft Corporation, ".NET Remoting: Binary Format Data Structure".

[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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC4646] Phillips, A., and Davis, M., Eds., "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006, http://www.rfc-editor.org/rfc/rfc4646.txt

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SQL92] Digital Equipment Corporation, "Database Language SQL", July 1992, http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[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] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

None.

1.3 Overview

DataSet is a class that is part of the Microsoft .NET Framework. The **DataSet** class provides an **in-memory** representation of relational data. A **DataSet** object contains a set of DataTable objects, Constraint objects, and DataRelation objects. A **DataTable** contains a collection of DataColumn objects that represents the schema and a collection of **DataRow** objects that represents the data.

In addition to storing information in the **DataSet**, applications can attach additional data to the **DataSet**, or to particular **DataTable** objects or **DataColumn** objects within a **DataTable**, by using extended properties. Extended properties are name/value pairs that are exposed to consumers of the **DataSet**, but do not affect the data or schema contained in the **DataSet** in any way.

In various scenarios, it is necessary to transfer a **DataSet** across application boundaries. This is usually accomplished by **serializing** the **DataSet** into a format suitable for transmission. This serialized form contains the **DataSet**, **DataTable** objects, **DataRow** objects, **DataColumn** objects, **Constraint** objects, **DataRelation** objects, and all of the extended properties. Common methods include a **web service method** that either takes or returns a **DataSet**.

The **DiffGram** is an **XML document** that contains a serialized form of a **DataSet**. Any **DataSet** instance can be serialized into a **DiffGram** that can be transmitted over a service interface or written to **persistent storage**. The **DiffGram** structure encapsulates all of the information required to re-create the in-memory **DataSet** in the exact state it was in at the time it was serialized. This includes the schema information that defines the structure of the data in the **DataSet** in addition to the data itself. The **DiffGram** also contains serialized representations of any extended properties that have been defined on the **DataSet** object, tables, columns, constraint objects, and relations.

1.4 Relationship to Protocols and Other Structures

Data types that the **DataSet** object uses for the type of the DataColumn object are specified in [MS-DTYP] and in [MS-NRBF]. Other types not included in these references are specified in this document. The schema and data in the **DataSet** are serialized as XML. There is a mapping between the **DataSet** in-memory representation and the **DiffGram** XML representation. This mapping is based on **XML Schema** specified in [XMLSCHEMA1] and [XMLSCHEMA2] .

Whenever a **DataSet** is returned from or received by a web service method, the **DiffGram** structure is used as the default serialization format. When used this way, the **DiffGram** can be wrapped in other data structures (for example, as specified in [SOAP1.1], section 4) that encapsulate other parts of the web service call.

Web services that exchange **DataSet** objects can use a variety of network protocols and encodings to transfer **DiffGram** XML documents. For example, one web service can choose to use a plain-text encoding of a **DiffGram** within a **Simple Object Access Protocol (SOAP)** envelope, transmitted by using **Hypertext Transfer Protocol (HTTP)** as specified in [RFC2616]. Another web service can use a binary encoding for the **SOAP envelope** that contains the DiffGram and can transmit it by using

User Datagram Protocol (UDP). The network protocols and encodings that can be used to transmit DiffGram XML documents are completely independent of the DiffGram XML format and are not covered in this document.

1.5 Applicability Statement

The **DiffGram** structure can be used whenever a serialized representation of a **DataSet** object is needed. More generally, the **DiffGram** can be used whenever it is necessary to serialize structure, data values, changes, and error information for tabular data. This document specifies the serialization of tabular data for the set of types that are supported by **DataSet**. Any other types are not specifically covered by this document.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

This section contains the following three subsections:

- **DataSet Concepts**: This section provides an overview of the concept of the **DataSet**, the .NET Framework class that contains an in-memory cache of data and schema information.
- **Data Model**: This section describes the data model used by **DataSet** to store information about data and schema. This section also covers the mapping between the .NET Framework data types and XML Schema types.
- **DiffGram XML Structure**: This section describes the **DiffGram** structure, which is a serialized XML representation of a **DataSet**.

2.1 DataSet Concepts

As previously discussed, a **DataSet** object contains DataTable objects and DataRelation objects. The schema of the **DataSet** is defined by the DataColumn objects that make up each of the **DataTable** objects together with the **DataRelation** objects and Constraint objects.

A **DataSet** is described by the following properties.

Property	Description
CaseSensitive	Indicates whether string comparisons within this **DataSet** are case-sensitive.
DataSetName	Specifies the name of the **DataSet**.
Locale	Specifies the **locale** of the data in the **DataSet**.
Namespace	Specifies the **XML namespace** of the serialized **DiffGram** that represents this **DataSet**.
Prefix	Specifies the XML prefix that aliases a namespace of the **DataSet**.
ExtendedProperties	Specifies the name/value pairs.

2.1.1 DataTable

DataTable objects contain one or more DataColumn objects, zero or more DataRow objects, and zero or more Constraint objects.

A **DataTable** object is described by the following properties.

Property	Description
CaseSensitive	Indicates whether string comparisons within this **DataTable** are case-sensitive.
Locale	Specifies the locale of the data in this **DataTable**.
Namespace	Specifies the XML namespace of the serialized DiffGram that represents this **DataTable**.
Prefix	Specifies the XML prefix that aliases a namespace of the **DataSet**.
TableName	Specifies the name of this **DataTable**.
ExtendedProperties	Specifies the name/value pairs of this **DataTable**.

2.1.1.1 DataColumn

As previously discussed, the **DataColumn** contains schema information for its corresponding data in the DataRow. The **DataColumn** object is described by the following properties.

Property	Description
AllowDbNull	Indicates whether all **DataRow** objects that contain this **DataColumn** MUST specify a non-null value.
AutoIncrement	Indicates whether this **DataColumn** automatically populates and increments its value for new **DataRow** objects that are added to the containing DataTable object.
AutoIncrementSeed	Indicates the number to increment from when the first **DataRow** is created.
AutoIncrementStep	Indicates how much to increment for each new **DataRow** that is added.
Caption	Specifies the caption of this **DataColumn**.
DataType	Specifies the data type of this **DataColumn**; an example would be **System.Int32** (an integer). Only specific types are understood by **DataSet**. For more information, see section 2.2.
DateTimeMode	Specifies one of the following values: Local or Unspecified. Applicable only when **DataType** is **DateTime**.
DefaultValue	Optional. If specified, indicates the default value that this **DataColumn** will be assigned when a new **DataRow** is created in which no other specific value is assigned.
Expression	Specifies a string that represents a calculated value.
MaxLength	Indicates the maximum length (in characters) of the values in this **DataColumn** Applicable only when **DataType** is a string.
ColumnName	Specifies the name of this **DataColumn**.
Namespace	Specifies the XML namespace of the serialized **DiffGram** that represents this **DataSet**.
Prefix	Specifies the XML prefix that aliases a namespace of the **DataSet**.
Readonly	Indicates whether the value of this **DataColumn** can be changed.
ColumnMapping	Indicates whether the instance values are represented in one of three different ways, Attribute, Element, or Hidden. This information influences how the **DataInstance** writes out values. For more information, see section 2.3.2.1.
Unique	Indicates whether the value of this **DataColumn** MUST be unique.
ExtendedProperties	Specifies the name/value pairs of this **DataColumn**.

2.1.1.2 DataRow

The **DataRow** object contains the actual data and errors, in addition to information on the changes for the data. For each **DataRow** object, two separate rows of information, the original values and the current values, are stored. These allow **DataSet** to track the changes.

The following is a visual representation of a single **DataRow**.

	Id	Name
Original	1	Chris
Current	1	Kris

The original row in the preceding representation contains the values that were originally loaded into the **DataSet**. The current row contains the current values, reflecting any changes to the original values that might have been made in memory. The values for original or current might be empty, but not both.

In addition to the values in current and original, there is information in the **DataRow** on error information. There is a **RowError** property, which is a string that indicates an error for the **DataRow**. There is also error information that is used for each DataColumn within the **DataRow**.

2.1.1.3 Constraint

There can be relationships and constraints between multiple DataTable objects or within a **DataTable** object. There are three supported constraints: **primary key**, foreign key, and unique, as specified in [SQL92].

A **UniqueConstraint** object describes which DataColumn object or set of **DataColumn** objects in a particular **DataTable** object require unique values across all DataRow objects. A **UniqueConstraint** that has a **PrimaryKey** property set to true describes the set of **DataColumn** objects in a particular **DataTable** used to identify a specific **DataRow** object. A **ForeignKeyConstraint** object connects one or more **DataColumn** objects in one **DataTable** to one or more **DataColumn** objects in another **DataTable**.

When a **DataRow** is added to a **DataTable** that has a foreign key constraint on one or more of its **DataColumn** objects, these **DataColumn** objects MUST contain values that exist in the linked **DataTable**. Otherwise, the constraint will be violated.

A **ForeignKeyConstraint** is described by the following properties.

Property	Description
AcceptRejectRule	Indicates the action that is to be taken when modifications become accepted as the current values.
Columns	Specifies the set of **DataColumn** objects to which this constraint applies.
ConstraintName	Specifies the name of this constraint.
DeleteRule	Indicates the action that occurs when a **DataRow** is deleted.
RelatedColumns	Specifies the set of **DataColumn** objects in the target **DataTable** to which this **DataTable** is related.
RelatedTable	Specifies the parent **DataTable** in this constraint.
Table	Specifies the name of the **DataTable** to which this constraint applies.
UpdateRule	Indicates the action that occurs when a **DataRow** is updated.
ExtendedProperties	Specifies the name/value pairs of this constraint.

A **UniqueConstraint** is described by the following properties.

Property	Description
Columns	Specifies the set of **DataColumn** objects to which this constraint applies.
ConstraintName	Specifies the name of this constraint.
IsPrimary	If true, specifies that the **DataColumn** objects to which this constraint applies are a primary key. Otherwise, false.
Table	Specifies the name of the **DataTable** to which this constraint applies.
ExtendedProperties	Specifies the name/value pairs of this constraint.

2.1.2 DataRelation

A **DataRelation** object represents a parent/child relationship between two DataTable objects that are connected by a **ForeignKeyConstraint** object. This **DataRelation** can be used to traverse the relationship graph between **DataTable** objects.

The **DataRelation** can also be used to specify an action to be taken when DataRow objects in a parent **DataTable** are deleted. The action can allow the change to cascade to the child **DataTable** or not. An example of a cascading action specifies that when a parent **DataTable** has a **DataRow** deleted, all **DataRow** objects in the child **DataTable** that were related to the deleted **DataRow** should be subsequently deleted as well.

A **DataRelation** is described by the following properties.

Property	Description
ChildColumns	Specifies the set of DataColumn objects in the child **DataTable** to which this **DataRelation** applies.
ChildKeyConstraint	Specifies the **ForeignKeyConstraint** for this **DataRelation**. For more information, see section 2.2.
ChildTable	Specifies the **DataTable** that is the child **DataTable** in this **DataRelation**.
Nested	Specifies whether this **DataRelation** is nested. In the **DiffGram**, a **Nested** child **DataTable** is a **child element** of the parent **DataTable** element. For more information, see section 2.3.
ParentColumns	Specifies the set of **DataColumn** objects in the parent **DataTable** to which this **DataRelation** applies.
ParentKeyConstraint	Specifies the set of **DataColumn** objects that acts as a unique or primary key for the parent **DataTable**. For more information, see section 2.2.
ParentTable	Specifies the **DataTable** that is the parent **DataTable** in this **DataRelation**.
RelationName	Specifies the name of this **DataRelation**.
ExtendedProperties	Specifies the name/value pairs of this **DataRelation**.

2.2 Data Model

This section explains the data types that can be used with the **DataSet** and their relationship to [XMLSCHEMA2] types in the context of the **DataSet DiffGram** structure. The conceptual representation of these types is described in the following sections or referenced from other protocols.

These types are mapped to the [XMLSCHEMA2] type that each corresponds to. The mapping to the [XMLSCHEMA2] type is important because in all cases except for two, these mappings dictate the serialization of an instance of data.

2.2.1 .NET Framework Types for DataColumn Objects

The following types are specified in [MS-DTYP] (introduced in the Microsoft .NET Framework 1.0).

Object Boolean

Byte

Int8

Int16

Int32

Int64

Uint16

Uint32

Double

String

The following types are specified in [MS-NRBF] (introduced in the .NET Framework 1.0).

Single

TimeSpan

DateTime

Decimal

The following table describes additional types that the **DataSet** object can use. Further information about how values are serialized is specified in section 2.2.3.

Type name	Description	.NET Framework introduction
System.Uri	A resource that is available to the application on the intranet or Internet.	.NET Framework 1.0
System.Guid	A GUID. A GUID is a 128-bit integer (16 bytes) that can be used across all computers and networks wherever a unique identifier is required.	.NET Framework 1.0
System.DateTimeOffset	A point in time, typically expressed as a date and time of day, relative to Coordinated Universal Time (UTC).	Microsoft .NET Framework 3.5
System.Numerics.BigInteger	An unbounded integer.	Microsoft .NET Framework 4.0
System.Data.SqlTypes.SqlBinary	A variable-length stream of binary data to be stored in or retrieved from a database.	Microsoft .NET Framework 2.0

Type name	Description	.NET Framework introduction
System.Data.SqlTypes.SqlBoolean	An integer value, either 1 or 0, to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlByte	A variable-length stream of bytes to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlChars	A variable-length stream of chars to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlDateTime	The date and time data - ranging in value from January 1, 1753 to December 31, 9999 to an accuracy of 3.33 milliseconds - to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlDecimal	A numeric value, between - 10^38 +1 and 10^38 - 1, with fixed precision and scale.	.NET Framework 2.0
System.Data.SqlTypes.SqlDouble	A floating-point number within the range of -1.79E +308 through 1.79E +308 to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlGuid	See **System.Guid** in this table.	.NET Framework 2.0
System.Data.SqlTypes.SqlInt16	A 16-bit signed integer to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlInt32	A 32-bit signed integer to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlInt64	A 64-bit signed integer to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlMoney	A currency value, ranging from -2^ 63 (or -922,337,203,685,477.5808) to 2^63 -1 (or +922,337,203,685,477.5807), with an accuracy to a ten-thousandth of currency unit, to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlSingle	A floating point number, within the range of -3.40E +38 through 3.40E +38, to be stored in or retrieved from a database.	.NET Framework 2.0
System.Data.SqlTypes.SqlString	A variable-length stream of characters to be stored in or retrieved from the database.	.NET Framework 2.0
System.Data.SqlTypes.SqlXml	XML data stored in or retrieved from a server.	.NET Framework 2.0

2.2.2 Mapping [XMLSchema2] to .NET Framework Types

The following table details the mapping from types that are defined in [XMLSCHEMA2] to the types in the .NET Framework. All types that are specified in the [XMLSCHEMA2] specification have a mapping.

XML Schema type	.NET Framework type	Comments
string	**String**	

XML Schema type	.NET Framework type	Comments
normalizedString	**String**	
Boolean	**Boolean**	
float	**Single**	
double	**Double**	
decimal	**Decimal**	
duration	**TimeSpan**	
Base64Binary	**Byte[]**	
hexBinary	**Byte[]**	
anyURI	**System.Uri**	
ID	**String**	
IDREF	**String**	
ENTITY	**String**	
NOTATION	**String**	Columns are not created for notation declaration elements; elements whose base type is NOTATION are created as string columns.
QName	**String**	
language	**String**	
IDREFS	**String**	
ENTITIES	**String**	
NMTOKEN	**String**	
NMTOKENS	**String**	
Name	**String**	
NCName	**String**	
integer	**Int64**	
nonPositiveInteger	**Int64**	
negativeInteger	**Int64**	
long	**Int64**	
int	**Int32**	
Short	**Int16**	
byte	**Sbyte**	
nonNegativeInteger	**UInt64**	
unsignedLong	**UInt64**	
unsignedInt	**UInt32**	

XML Schema type	.NET Framework type	Comments
unsignedShort	**UInt16**	
unsignedByte	**Byte**	
positiveInteger	**UInt64**	
dateTime	**DateTime**	
time	**DateTime**	
date	**DateTime**	
gYear	**DateTime**	The year is not validated; it is only read.
gYearMonth	**DateTime**	The year is not validated; it is only read.
gMonth	**DateTime**	The month is not validated; it is only read.
gMonthDay	**DateTime**	The month is not validated; it is only read.
gDay	**DateTime**	The day is not validated; it is only read.

2.2.3 Mapping .NET Framework Types to [XMLSCHEMA2] Types

The following table details the mapping from Microsoft .NET Framework types to types that are defined in [XMLSCHEMA2].

For some .NET Framework types, the mapping to the [XMLSCHEMA2] type is not sufficient to fully define the serialization format. In these cases, additional information is provided for what MUST be specified to add to the XML schema and/or what MUST be specified for the layout of the data.

Types that are not listed in the following table can be used in the **DiffGram**. However, their serialization formats are outside the scope of this document.

.NET Framework type	[XMLSCHEMA2] type

Char	The XML schema of a **Char** MUST be written as restriction of a string type where the length = 1, as in the following example.
	
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:length value="1" />`
	`</xs:restriction></xs:simpleType>`

String	String
Boolean	Boolean
Double	Double
Decimal	Decimal
TimeSpan	Duration

.NET Framework type	[XMLSCHEMA2] type
Byte[]	base64Binary
Single	Float
Int64	Long
Int32	Int
Int16	Short
Sbyte	Byte
UInt64	unsignedLong
UInt32	unsignedInt
UInt16	unsignedShort
Byte	unsignedByte
DateTime	dateTime
Guid	String Any **Guid** instance MUST be serialized into a string form as defined by the following **Augmented Backus-Naur Form (ABNF)**. guidLiteral = 8*HEXDIG "-" 4*HEXDIG "-" 4*HEXDIG "-" 12*HEXDIG
Uri	anyUri
BigInteger	Anytype Any **BigInteger** value MUST be serialized into the same form as the XML Schema (XSD) type integer as specified in [XMLSCHEMA2].
SqlBinary	hexBinary
SqlBoolean	Boolean
SqlByte	Byte
SqlBytes	base64Binary
SqlChars	String
SqlDateTime	dateTime
SqlDecimal	Decimal
SqlDouble	Double
SqlGuid	Guid
SqlInt16	Short
SqlInt32	Int
SqlInt64	Long
SqlMoney	Decimal
SqlSingle	Float

.NET Framework type	[XMLSCHEMA2] type
SqlString	String
SqlXml	Anytype

2.2.4 XSD Data Type Keywords

The following [XMLSCHEMA2] type keywords are supported for the [XMLSCHEMA2] types **float** and **double**. If the values of the .NET Framework types **Double** or **Single** are equal to the values that are described in the following table, the values MUST be set by using the correct schema type keyword as specified in [XMLSCHEMA2].

XSD keyword	Description	.NET Framework type
0, -0	Positive and negative zero.	0, -0
INF, -INF	Positive and negative infinity.	Double.PositiveInfinity Double.NegativeInfinity Single.PositiveInfinity Single.NegativeInfinity
NaN	Not a number.	Double.Nan Single.Nan

2.3 DiffGram XML Structure

A valid **DataSet DiffGram** structure MUST conform to the following rules:

- The **DiffGram** MUST have a **root element**, hereafter referred to as the <root> element.
- If the <root> element contains at least one <schema> element as specified in [XMLSCHEMA2], the following rules MUST apply:
  - If the <root> element does not contain one or more <schema> elements, the schema is assumed to be predefined between the producer and consumer.
  - If the <root> element contains a <diffgr:diffgram> element, hereafter referred to as the **DiffGram Data** element, the <root> element MUST be a <diffgram> element as defined in the namespace urn:schemas-microsoft-com:xml-diffgram-v1. If the **DiffGram Data** element is not specified, the consumer MUST assume that the data is empty.
- The XML that comprises a **DiffGram Data** element MUST include required **XML elements** and **XML attributes** as specified in the following sections of this document. These XML elements and XML attributes are defined in various XML namespaces. The following table lists these XML namespaces and specifies the XML namespace prefixes that are commonly associated with them. Producers and consumers of **DataSet DiffGram** structures MUST ensure that the XML references these namespaces by using the mechanisms that are specified in [XMLNS], but they SHOULD<1> use the prefixes that are shown in the following table. For clarity, when XML elements and attributes from these namespaces are referenced in this document, their fully-qualified names are used.

Description	Namespace URI	Commonly used prefix	Reference
XML Schema elements and attributes	http://www.w3.org/2001/XMLSchema	xs	[XMLSCHEMA1] [XMLSCHEMA2]
DiffGram elements and attributes	urn:schemas-microsoft-com:xml-diffgram-v1diffgr	diffgr	This namespace is internal to the **DiffGram** structure and is described in section 2.3.2.
DataSet specific annotations	urn:schemas-microsoft-com:xml-msdata	msdata	[MC-ADONETDSSS]
DataSet extended properties	urn:schemas-microsoft-com:xml-msprop	msprop	User and application-specific information can be annotated on the **DataSet** schema with extended properties. The extended properties are defined in this namespace.

The sections that follow define the <schema> elements and the **DiffGram Data** element in more detail. At a basic level, the purpose of these elements can be explained as follows:

- The **DiffGram** <schema> elements define the XML Schema that is specified in [XMLSCHEMA1] and [XMLSCHEMA2], which is a representation of the structure of the data that is contained in the **DiffGram Data** element. The **DiffGram** <schema> elements are then mapped to **DataSet** concepts.
- The **DiffGram Data** element encapsulates the values of the data in the **DataSet**.

2.3.1 DiffGram <schema> Elements

2.3.1.1 DataSet DiffGram Schema Mapping

Any XML document that is specified in [XMLSCHEMA1] and [XMLSCHEMA2] can be mapped to a relational structure in the **DataSet** object. Following is a formal set of rules that outlines this mapping to a **DataSet** object. Cases in which the schema specified in [XMLSCHEMA1] and [XMLSCHEMA2] are not understood are specifically mentioned.

2.3.1.1.1 <schema> Element

Following are the rules that MUST be followed for the <schema> element:

- If the **schema** element contains an element named **element** whose **msdata:IsDataSet** attribute is set to true, this element MUST be mapped to the **DataSet** object and hereafter is referred to as the **DataSet Schema** element. Otherwise, the <schema> element itself is mapped to the **DataSet**, and the name of the **DataSet** is mapped from the **id** attribute, if it is specified.
  - If a child element of the <schema> element has a **msdata:IsDataSet** attribute set to false, this child element MUST be mapped to a DataTable object and not a **DataSet**.
- If the <schema> element does not contain an element whose **msdata:IsDataSet** attribute is set to true, the following rules apply:
  - If there is an **id** attribute of the <schema> element, it MUST be mapped to the name of the **DataSet**.
- If the attribute **targetNamespace** is specified on the <schema> element, it must be set as the **targetNamespace** of the **DataSet** and MUST be a valid namespace string as specified in

[XMLSCHEMA1]. If the attribute **targetNamespace** is not specified, the **namespace** property of the **DataSet** MUST be set to an empty string.

- <Attribute> elements that are children of the <schema> element and are not referenced elsewhere MUST be ignored. If the <attribute> elements are referenced, they MUST follow the mapping rules that are specified in section 2.3.1.1.15.
- Any element that is a <simpleType> element as specified in [XMLSCHEMA2] and that is a direct child of the <schema> element and not referenced as specified in [XMLSCHEMA2], MUST be ignored in the mapping. If the <simpleType> element is referenced, it MUST follow the mapping rules that are specified in section 2.3.1.1.14.
- Any element that is a <complexType> element as specified in [XMLSCHEMA2] and that is a direct child of the <schema> element and not referenced as specified in [XMLSCHEMA2], MUST be ignored in the mapping. If the <complexType> element is referenced, it MUST map to a **DataTable** and MUST follow the rules that are specified in section 2.3.1.1.8.
- Any element that is a <group> element that is a direct child of the <schema> element and not referenced as specified in [XMLSCHEMA2], this element MUST be ignored in the mapping. If the named <group> element is referenced, it MUST follow the mapping rules that are specified in section 2.3.1.1.5.
- Any element that has the **abstract** attribute equal to true and that is not referenced as specified in [XMLSCHEMA2] by any other elements MUST be ignored. If the element that is referenced has the **abstract** attribute equal to true, the referencing type follows the rules that are described in section 2.3.1.1.10.

2.3.1.1.2 DataSet Schema Element

If the **DataSet Schema** element is present, the name of the **DataSet** object is determined by the following rules:

- If the **name** attribute is specified, the name of the **DataSet** MUST be set to the value of the **name** attribute.
- If the **name** attribute is not specified in the **DataSet Schema** element, the value of the **id** attribute of the <schema> element MUST be mapped to the name of the **DataSet**.
- If the **DataSet Schema** element contains one or more child <complexType> elements with a <choice> element compositor, the child <complexType> elements will map to one or more DataTable objects.
- If the **msdata:CaseSensitive** attribute is specified within the **DataSet Schema** element, its value MUST be true or false.
- If the **msdata:CaseSensitive** attribute is not specified, the value MUST be false. The value MUST map to the **CaseSensitive** property on **DataSet**.
- If the **msdata:Locale** attribute is specified within the **DataSet Schema** element, the value of the **msdata:Locale** attribute MUST follow [RFC4646]. If the attribute is not specified and **msdata:UseCurrentLocale** is specified and is equal to true, the **Locale** property on the **DataSet** MUST be set to the current local of the computer based on [RFC4646].
- If **msdata:UseCurrentLocale** is specified and is equal to false or if **msdata:UseCurrentLocale** is not specified and **msdata:Locale** is not specified, the **Locale** property on the **DataSet** MUST be "en-us".
- If the **msdata:Prefix** attribute is specified, the value will be the **Prefix** property of the **DataSet**. If the attribute is not specified, the **Prefix** property of the **DataSet** MUST be an empty string.

- If the **DataSet Schema** element contains attributes that specify the urn:schemas-microsoft-com:xml-msprop namespace, each attribute name/value pair MUST specify a name/value pair in the **ExtendedProperties** that exists on the **DataSet** object.
- Any additional attribute that is not specified earlier in this section MUST be ignored.

The following schema excerpt illustrates this.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="MyDataSet" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="dataset" msdata:IsDataSet ="true">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element ref="customer"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CustomerName" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

2.3.1.1.3 <include> Element

As defined in [XMLSCHEMA2], the <include> element is optionally used within a <schema> element to include other definitions and declarations in the schema. The following schema excerpt is a sample usage.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="MyDataSet" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:include schemaLocation="http://www.example.org/XMLSchema1.xsd"/>
  <xs:element name="dataset" msdata:IsDataSet="true">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element ref="customer"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Here the **DataSet** object, DataTable objects, and DataColumn objects all belong to the **targetNamespace** <schema> element unless explicitly specified in the child elements of the <include> element.

2.3.1.1.4 <import> Element

As defined in [XMLSCHEMA2] the <import> element is optionally used within the <schema> element. If the <import> element is specified, it MUST follow the rules specified in [XMLSCHEMA1]. Following is an example that uses <import>.

```
<xs:schema id="SampleDataSet"
  targetNamespace="http://www.microsoft.com"
  xmlns="http://www.microsoft.com"
  xmlns:xs="http://www.w3.org/2000/10/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
  xmlns:myns="http://www.example.com/myns"
```

```
  elementFormDefault = "qualified" >
  <!-- import for types from myns -->
  <xs:import namespace="http://www.example.com/myns" schemaLocation="
http://www.example.com/myns/person.xsd"/>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="name" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="string">
              <xs:attribute name="age" type="myns:age" />
              <xs:attribute ref="myns:height" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

In the example, the **age** attribute belongs to the "http://www.microsoft.com" **targetNamespace** because it is named, although it has the same type information as the **age** attribute from the "http://www.example.com/myns" namespace. However, the **myns:height** attribute belongs to the "http://www.example.com/myns" imported namespace, because it is referenced.

Whenever an element is referenced to a type that is defined in another namespace, the **Namespace** property on the respective component MUST be set accordingly. In the preceding example, the **Column.Namespace** property for the **DataColumn age** object is "http://www.microsoft.com" and for the **DataColumn height** object is "http://www.example.com/myns". All XML instance data for these **DataColumn** objects must be in the correct namespace. Otherwise, it MUST be ignored.

2.3.1.1.5 <annotation> Elements within XSD Schemas

The following rules MUST be used for processing <annotation> elements within XSD Schemas. As specified in [XMLSCHEMA2], an <annotation> element is optional, but it is ignored when mapping to **DataSet** concepts except in the following cases:

- The <annotation> element MUST NOT be ignored in the mapping process when it contains an <msdata:Relationship> subelement. An <msdata:Relationship> element MUST be mapped to a DataRelation object. If an <msdata:Relationship> element is specified, the following rules MUST be followed:
  - The <msdata:Relationship> element MUST be used only when there is no corresponding **ForeignKeyConstraint** by a <keyref> element. For more information, see section 2.1.2.
  - If the <msdata:Relationship> is nested within one element that is mapped to a DataTable object (known as a parent **DataTable**), which is in turn nested within another element mapped to a **DataTable** (known as a child **DataTable**), the **Nested** property of the **DataRelation** property MUST be true.

    Otherwise, if the <msdata:Relationship> is not nested within a **DataTable**, the **Nested** property of the **DataRelation** property MUST be false. Note that the term **Nested** here refers only to the XML structure of the **DiffGram**, not to any relational data concept.
  - The **msdata:parent** attribute MUST exist and the value MUST map to a **DataTable**. If the <msdata:Relationship> is nested within one element that is mapped to a **DataTable** (known as a parent **DataTable**), which is in turn nested within another element mapped to a **DataTable** (known as a child **DataTable**), the parent attribute value MUST be the same name as the parent **DataTable**.

- The **msdata:child** attribute MUST exist and the value MUST map to a **DataTable**. If the <msdata:Relationship> is nested within one element that is mapped to a **DataTable** (known as a parent **DataTable**) which is in turn nested within another element mapped to a **DataTable** (known as a child **DataTable**), the child attribute value MUST be the same name as the child **DataTable**.
- The **msdata:parentKey** attribute MUST exist. The values MUST be a comma-separated list of column names. These column names MUST be DataColumn objects that exist in the already specified parent **DataTable**. The **DataColumn** objects specified are the ones to which this relationship applies.
- The **msdata:childKey** attribute MUST exist. The values MUST be a comma-separated list of **DataColumn** names. These **DataColumn** names MUST be **DataColumn** objects that exist in the already specified child **DataTable**.
- The **name** attribute MUST be specified and mapped to the **RelationName** property of the **DataRelation**.
- The <msdata:Relationship> elements MUST be written subsequent to the **Nested** parent/child or MUST be written after the associated parent and child **DataTable** elements in the **DiffGram** structure.
- If the <msdata:Relationship> element is **Nested**, there MUST be only one <msdata:Relationship> that is nested within the child **DataRelation** object.
- If the <msdata:Relationship> element has attributes that specify the urn:schemas-microsoft-com:xml-msprop namespace, each attribute name/value pair MUST specify a name/value pair in the **ExtendedProperties** property that exists on the **DataRelation** object.

In the following example, a relationship is defined for col1 and col2 between table1 and table2.

```
<xs:schema id="MyDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

  <xs:element name="table1">
    <xs:complexType>
      <xs:all>
        <xs:element name="col1" minOccurs="0" type="xs:string" />
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:element name="table2">
    <xs:complexType>
      <xs:all>
        <xs:element name="col1" minOccurs="0" type="xs:string" />
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:annotation>
    <xs:appinfo>
      <msdata:Relationship name="Relation1" msdata:parent="table1" msdata:child="table2"
msdata:parentkey="col1" msdata:childkey="col1" />
    </xs:appinfo>
  </xs:annotation>

</xs:schema>
```

In the following example, a relationship is defined for col1 and col2 between Table1 and Table2. The relationship is written on the child **DataTable**.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="DataSet" targetNamespace="xyz" xmlns=""
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="Table1">
    <xs:complexType>
      <xs:all>
        <xs:element name="col1" minOccurs="0" type="xs:string" />
        <xs:element name="Table2">
          <xs:annotation>
            <xs:appinfo>
              <msdata:Relation name="Relation11" msdata:parent="Table1" msdata:child="Table2"
msdata:parentkey="col1" msdata:childkey="col1" msdata:CreateConstraints="False" />
            </xs:appinfo>
          </xs:annotation>
          <xs:complexType >
            <xs:all>
              <xs:element name="col1" minOccurs="0" type="xs:string" />
              <xs:element name="col2" minOccurs="0" type="xs:string" />
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.3.1.1.6 <group> Element

As specified in [XMLSCHEMA2], the <group> element is optional and MUST follow the following rules:

- The <group> element MUST follow the rules in [XMLSCHEMA1].
- The **minOccurs** and **maxOccurs** attributes on the <group> element MUST be ignored and have no effect in the mapping process.
- The child elements of the <group> element MUST be deserialized in accordance with the rules in section 2.3.1.1.13.1.

2.3.1.1.7 Usage of the ref Attribute

As defined in [XMLSCHEMA2], named <schema> elements are optionally used with **ref** attributes within a **DiffGram** structure. If these <schema> elements are specified, they MUST follow [XMLSCHEMA2]. For the purposes of the **DiffGram**, when the schema is being read, any named elements that are not referenced within the declared schema instances will not be processed.

2.3.1.1.8 Element Containing <complexType> Elements

The following rules apply when a particular element is mapped to a DataTable object.

- For an element that contains one or more <complexType> elements:
  - If the element is defined by a <complexType> directly or through a **ref** attribute, the element MUST be mapped to a **DataTable**.
  - There MUST be a **name** attribute specified in the <complexType> element. The value of the **name** attribute MUST map to the name of the **DataTable**. The combination of the **name** and the XML namespace of the **DataTable** object MUST be unique. The name value MUST map to the **TableName** property on the associated **DataTable**.
- The following rules MUST be followed to determine the **namespace** property of the **DataTable** from the **DiffGram** element that maps to a **DataTable**.

- If there is a **targetNamespace** attribute on the **DataSet** element, the element MUST map to the **DataTable namespace** property.
- If the **targetNamespace** attribute is not specified or its value is empty, the **namespace** property on the mapped **DataTable** MUST be the **targetNamespace** attribute of the current <schema> element.
- If the **DataTable** object is referenced from an <import> element schema, the **DataTable namespace** property MUST be mapped to the **targetNamespace** of the imported schema.

- Any child element non-repeatable <simpleType> elements (<simpleType> elements with **maxOccurs** equals 1, or attributes), within the <complexType> element as specified in an <all>, <sequence>, or <choice> element, MUST be mapped to a DataColumn within the **DataTable** that the element is mapped to. Each element that maps to a **DataColumn** MUST follow the rules in section 2.3.1.1.14.
- If the **msdata:CaseSensitive** attribute is specified, the value MUST be true or false. If the attribute is not specified, the default value is false. The value MUST map to the **CaseSensitive** property on the mapped **DataTable**.
- If the **msdata:Locale** attribute is specified, the value MUST follow [RFC4646]. If the attribute is not specified, the value MUST be what the **Locale** value of the **DataSet** object is, as specified in section 2.3.1.1.2. The value MUST map to the **Locale** property on the mapped **DataTable**.
- If the element is mapped to a **DataTable**, each attribute that specifies the urn:schemas-microsoft-com:xml-msprop namespace MUST specify a name/value pair in the **ExtendedProperties** property that exists on the **DataTable**.

  If the element is mapped to a **DataColumn**, each attribute that specifies the urn:schemas-microsoft-com:xml-msprop namespace MUST specify a name/value pair in the **ExtendedProperties** that exists on the **DataColumn**.
- The following attributes within the element MUST be ignored for elements that contain <complexType> elements:
  - **block**
  - **default**
  - **equivClass**
  - **final**
  - **fixed**
  - **id**
  - **minOccurs** (This attribute is invalid according to [XMLSCHEMA2] if the parent element is <schema>.)
  - **maxOccurs**>1 for <complexType> elements. (This attribute is invalid according to [XMLSCHEMA2] if the parent element is <schema>.)

Following is an example of how a <complexType> element is mapped to a **DataTable**.

```
<xs:element name="orderdetail">
  <xs:complexType>
  <xs:all>
     <xs:element name="orderID" minOccurs="0" type="xs:string"></xs:element>
    <xs:element name="description" minOccurs="0" type="xs:string"></xs:element>
  </xs:all>
```

```
 </xs:complexType>
 …
</xs:element>
```

The "OrderID" and "description" become **DataColumn** objects of the "orderdetail" **DataTable**.

2.3.1.1.9 <complexType> and <simpleType> Element Inheritance

XSD schemas support two styles of <complexType> element inheritance by using the extension or restriction elements. The following rules MUST be followed when mapping elements to DataSet concepts:

- The <restriction> element MUST be ignored in the mapping process.
- The <extension> element MUST be read to determine the base class.
- When mapping <complexType> elements with inheritance, if two or more <complexType> elements inherit from the same base class, the elements MUST map to separate DataTable objects.

2.3.1.1.10 <complexType> Inheritance

A derived <complexType> element MUST have as a base type either another <complexType> or another <simpleType> element. Any derived <complexType> that is not referenced as specified in [XMLSCHEMA2] MUST be ignored and not mapped.

In both cases, the following attributes are supported:

- **base**: This attribute MUST be used to determine any elements or attributes from the base class that are used in the new derived class and mapped to DataColumn objects.
- **abstract**: This attribute MUST be ignored in the mapping process.

2.3.1.1.11 <complexType> <complexContent>

The following rules MUST be followed to map a <complexType> that contains a <complexContent>:

- **base**: This attribute MUST be used to determine any elements or attributes from the base class that are used in the new derived class and are mapped to DataColumn objects.
- Any other elements in the sequence of the extension of the <complexContent> are mapped in accordance with the rules that are specified in section 2.3.1.1.13.1.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
         targetNamespace="http://cars.example.com/schema"
         xmlns:target="http://cars.example.com/schema">

 <complexType name="Vehicle" abstract="true">
    <sequence>
      <element name="type" type="string"/>
    </sequence>
  </complexType>

 <complexType name="Car">
  <complexContent>
   <extension base="target:Vehicle">
     <sequence>
      <element name="EngineSize" type="string"/>
     </sequence>
   </extension>
  </complexContent>
 </complexType>
```

```
 <complexType name="Plane">
  <complexContent>
   <extension base="target:Vehicle">
     <sequence>
      <element name="WingSpan" type="string"/>
     </sequence>
   </extension>
  </complexContent>
 </complexType>

</schema>
```

In the preceding example, the "Car" and "Plane" tables both have a **DataColumn** named "type". The "Car" DataTable also has a **DataColumn** that is called "EngineSize", and the "Plane" **DataTable** has a **DataColumn** that is called "WingSpan". Note that a **DataTable** is not created for the "Vehicle" <complexType> element.

The following provides an example of an extension from a base type.

```
<xs:complexType name="Address">
  <xs:sequence>
   <xs:element name="name"   type="xs:string"/>
   <xs:element name="street" type="xs:string"/>
   <xs:element name="city"   type="xs:string"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="USAddress">
  <xs:complexContent>
   <xs:extension base="Address">
    <xs:sequence>
     <xs:element name="state" type="xs:string"/>
     <xs:element name="zip"   type="xs:positiveInteger"/>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
```

In the preceding example, a **DataTable** is created for "USAddress" with columns called "name", "street", "city", "state", and "zip".

When deriving from a <complexType> element, either the sequence or the <group> element MUST be used. If there is a base class, it MUST NOT specify any other <group>, <choice>, or <all> elements.

If the <complexType> with <complexContent> element has a base attribute that is a <simpleType>, the mapping MUST be processed the same way as described earlier in this section.

2.3.1.1.12 <complexType> <simpleContent>

As defined in [XMLSCHEMA2], <complexType> elements optionally derive from <simpleType> elements, with new attributes added. In this case, the following rules MUST be followed:

- The element MUST be mapped to a DataTable.
- Each attribute MUST map to a DataColumn and MUST follow all the rules that are specified in section 2.3.1.1.15.
- Each element MUST map to a **DataColumn**, where the name of the **DataColumn** is specified by the **msdata:DataColumn** attribute or, if this attribute is not specified, by the name of the element together with "_text" appended to the end.

- The following code example illustrates a <complexType> with <simpleContent>.

```
<xs:element name="internationalPrice">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:decimal">
                <xs:attribute name="currency" type="xs:string" />
                <xs:attribute name="diff" type="xs:decimal" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
</xs:element>
```

The data instance might specify a value, as shown in the following example.

```
<internationalPrice currency="EUR" diff="1.53">423.46</internationalPrice>
```

In this case, a column that has the name of the table, followed by "_text", is created in the **DataTable** and populated with the values from the data instances. In the preceding example, the element <internationalPrice> is mapped to a **DataTable**, and the <internationalPrice> value, currency, and diff are mapped to **DataColumn** objects.

2.3.1.1.12.1 <simpleType> Inheritance via <restriction>

As defined in [XMLSCHEMA2], <simpleType> elements can be derived from other <simpleType> elements via a <restriction> element. When a <simpleType> derives from a <restriction> element where the type is a string, the **length** or **maxLength** attributes MUST be mapped to the **MaxLength** property of the DataColumn property. All the <restriction> attributes MUST be ignored except for **maxLength** when the type is a string.

2.3.1.1.12.2 <simpleType> Columns Marked as Abstract

As defined in [XMLSCHEMA2], a <simpleType> can optionally derive from abstract types. These elements MUST not be mapped to DataColumn objects.

2.3.1.1.13 Content of <complexType> Element

This section details the mapping of <complexType> elements that are contained within an element named <element>.

The following content within a <complexType> element MUST be ignored:

- minBound
- minExclusive
- minInclusive
- maxBound
- maxExclusive
- maxInclusive
- precision
- scale
- length

- minlength
- maxlength
- encoding
- period
- enumeration
- pattern
- any
- anyAttribute
- choice
- fixed
- all

2.3.1.1.13.1 <element> Element

If the element contains a <complexType> element or has a **maxOccurs** value that is greater than 1, it MUST be mapped to a DataTable object as defined in section 2.3.1.1.8.

The following criteria MUST be used to determine whether there is a nested DataRelation object or a **Constraint** object:

- If the <msdata:Relationship> element is specified in the child element that maps to the **DataTable**, this <relationship> element MUST map to the **DataRelation** whose **Nested** property MUST be set to true.
- If there is no <msdata:Relationship> element contained within the <element> element, a constraint can optionally be specified by the <selector> element of the <keyref> element, which appears anywhere within the schema section of the **DiffGram** document subsequent to the <element> element, containing the same **DataTable Name** as the child element that is mapped to a **DataTable**.
- If a parent/child relationship is specified by the <complexType> and no **DataRelation** is found, the following rules MUST be followed for mapping to the parent and child **DataTable** objects:
  - A foreign key **DataColumn** object named "<parenttable>_ID" MUST be added to the **DataTable** that maps to the inner child element, and the **ColumnMapping** property of the **DataColumn** object is set to **Hidden**.
  - If there is no primary key for the parent **DataTable**, a **DataColumn** named "<tablename>_ID" MUST be added to the **DataTable** that maps to the parent element.
  - A **DataRelation** MUST be created for the parent/child and be named "<parenttable>_<childtable>", whose **Nested** property is set to true.
  - A **Constraint** named "Constraint[n]" (where *n* is a unique number) MUST be created with a default value of "cascade" for the **UpdateRule** and **DeleteRule** properties and a default value of "none" for the **AcceptRejectRule** property.
- If the element contains a <simpleType> element and if the **maxOccurs** attribute specifies a value greater than 1, the element MUST be mapped to a **DataTable** as defined earlier in this section.

- Any <complexType> element in a <relationship> element MUST follow the rules specified in this section.
- If the element is a <simpleType> element or references a <simpleType> element and has a **maxOccurs** attribute value of 1, or if the **maxOccurs** attribute is not specified, in which case [XMLSCHEMA2] defines the default value to be 1, the element MUST map to a **DataColumn** and MUST follow the rules in section 2.3.1.1.14.

Following is an example of the preceding rules.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="MyDataSet" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="customer">
    <xs:complexType>
      <xs:all>
        <!-- "orders" becomes a related DataTable -->
        <xs:element name="order">
          <xs:complexType>
            <xs:all>
              <xs:element name="orderID" minOccurs="1" type="xs:string"></xs:element>
              <xs:element name="orderAmount" minOccurs="0"  type="xs:int"
default="100"></xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
        <xs:element name="Name" type="xs:string" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

In the preceding XSD example, the following items apply:

- The **DataTable** "customer" is defined.
- "Orders" becomes a separate **DataTable**, and the following occurs:
  - A "customer_id" **DataColumn** is added to the "customer" **DataTable**.
  - A "customer_id" **DataColumn** is added to the "orders" **DataTable**.
  - A "customer_order" relation is created between the "customer" and "order" tables.
  - A "Constraint1" constraint is defined between the "customer" and "order" tables.
- A **DataColumn** that has a name equal to "Name" is added to the customer **DataTable**.

2.3.1.1.13.2 <all>, <sequence>, and <choice> Elements

Elements appearing with <all>, <sequence>, or <choice> compositors as specified in [XMLSCHEMA2] MUST be evaluated according to the rules that are specified in section 2.3.1.1.8. The fact that these elements appear within an <all>, <sequence>, or <choice> compositor MUST be ignored for the purpose of mapping.

For example, in the following schema snippet, the <complexType> will map to a DataTable object with a **Name** property set to "Order". "Orderdetails", "ordertype2", and "ordertype1" will map to DataColumn objects.

```
          <xs:element name="Order2">
             <xs:complexType>
               <xs:choice minOccurs="1">
```

```
        <xs:element name = "ordertype1" type="xs:string"/>
        <xs:sequence>
          <xs:element name = "ordertype2" type="xs:string"/>
          <xs:element name = "orderdetails" type="xs:string"/>
        </xs:sequence>
      </xs:choice>
    </xs:complexType>
  </xs:element>
```

2.3.1.1.13.3 <any> Element

The <any> element within a <complexType> element MUST be ignored.

2.3.1.1.13.4 <attribute> Groups

<Attribute> groups are deserialized and individually mapped to DataColumn objects as described in section 2.3.1.1.15. <Attribute> groups MUST be defined at the <schema> element level.

2.3.1.1.13.5 <anyAttribute> Element

The <anyAttribute> element within a <complexType> element MUST be ignored.

2.3.1.1.13.6 <attribute> Element

The <Attribute> elements within a <complexType> element MUST be mapped to columns, as described in section 2.3.1.1.15.

2.3.1.1.14 <simpleType> Element Within <complexType> Elements

If the element is a <complexType> with <simpleContent> or if it is a <simpleType> element, the following mapping process MUST occur:

- The element MUST be mapped to a DataColumn object within the DataTable object that the parent element maps to.
- The **ColumnMapping** property of the **DataColumn** MUST be **Element**.
- If the type is a <complexType> with <simpleContent>, apply the rules described in section 2.3.1.1.12.
- If the element contains a **ref** attribute, the referenced type MUST be used for the remainder of the rules of this section.
- The name of the **DataColumn** MUST be the name of the element. This name MUST be unique within the **targetNamespace** attribute that the **DataColumn** element is in.
- If the type references a type from an imported element schema, the **namespace** property of the **DataColumn** MUST be the **targetNamespace** of this imported schema. Otherwise, the **namespace** of the **DataTable** MUST be the **targetNamespace** attribute from the current <schema> element.
- The following rules that MUST be followed to determine what data type the **DataColumn** will map to depending on how the type and the **msdata:DataType** attribute are used:
  - If the **msdata:DataType** attribute is present, the value MUST map to a data type as specified in section 2.2.3. The **msdata:DataType** value will either be the namespace and the class name or a fully qualified type name as specified by [ECMA-335:2006]. Only types that are defined in section 2.2.3 SHOULD<2> be specified in the non-fully-qualified form.

- If the **type** attribute is present and the **msdata:DataType** attribute is present, the value of the **type** attribute MUST be the correct [XMLSCHEMA2] type for the value of the **msdata:DataType** attribute that is specified according to section 2.2.3.
- If the **type** attribute is present and the **msdata:DataType** attribute is not present, the **DataColumn** data type MUST map to the data type according to section 2.2.2.
- If the **type** attribute is not present, the **DataColumn** data type MUST map to "string".

- The default attribute MUST not be specified if the **DataColumn** data type is **SqlXml** or any other data type that is not explicitly listed in section 2.2.3.
- If the element has a **default** attribute, the **DefaultValue** property of the **DataColumn** is set to this value. The value of the **default** attribute MUST be serialized according to the data type-to-[XMLSCHEMA2]-type serialization rules that are described in section 2.2.3.
- If the element has a **minOccurs** value of 0 and no **nillable** attribute is specified, the **AllowDbNull** property of the **DataColumn** MUST be mapped to true. If the **minOccurs** value is 1, the value of the **AllowDbNull** property of the **DataColumn** MUST be false. As specified in [XMLSCHEMA2], if **minOccurs** is not specified, the default value is 1, so the **AllowDbNull** property MUST be false.
- If the value of the **minOccurs** attribute is greater than 0, the **nillable** attribute optionally can be specified as specified in [XMLSCHEMA2]. If the **nillable** attribute is specified, its value MUST be true or false, and the **AllowDbNull** property of the **DataColumn** MUST be the value of the **nillable** attribute. If the attribute is not specified and **minOccurs** is greater than 0, the **nillable** value MUST be false and the **AllowDbNull** property of the **DataColumn** MUST be false.
- If the element has a **minOccurs** value of 0, the **nillable** attribute MUST be ignored.
- If the **msdata:AutoIncrement** attribute is specified, the **DataColumn AutoIncrement** property value MUST be true or false. If the attribute is not specified, the **DataColumn AutoIncrement** property value MUST be false.
- If the **msdata:AutoIncrementSeed** attribute is specified, the **DataColumn AutoIncrementSeed** property value MUST be a valid long value as specified in [XMLSCHEMA2]. If the attribute is not specified, the **DataColumn AutoIncrementSeed** property value MUST be zero.
- If the **msdata:AutoIncrementStep** attribute is specified, the **DataColumn AutoIncrementStep** property value MUST be a valid long value as specified in [XMLSCHEMA2]. If the attribute is not specified, the **DataColumn AutoIncrementStep** property value MUST be 1.
- If the **msdata:Caption** attribute is specified, the **DataColumn Caption** property value MUST be a valid string as specified in [XMLSCHEMA2]. If the attribute is not specified, the value MUST be the name of the **DataColumn** name.
- If the **msdata:Expression** attribute is specified, the **DataColumn Expression** property value MUST be a valid string as specified in [XMLSCHEMA2]. If the attribute is not specified, the **DataColumn Expression** property value is an empty string.
- If the **msdata:ReadOnly** attribute is specified, the **DataColumn ReadOnly** property value MUST be true or false. If the attribute is not specified, the **DataColumn ReadOnly** property value MUST be false.
- The following attributes MUST be ignored for <simpleType> elements within <complexType> elements:
  - **block**
  - **id**

In the following schema excerpt, the following items apply:

- "OrderID" becomes a **DataColumn** of the "order" **DataTable**.
  - Its ordinal is 0.
  - The data type of the **DataColumn** is "string".
  - **AllowDbNull** is false.
- "OrderAmount" becomes a **DataColumn** of the "order" **DataTable**.
  - Its ordinal is 1.
  - The data type of the **DataColumn** is "int".
  - **AllowDbNull** is true.
  - The **DefaultValue** is set to 100.
- "OrderDate" becomes a **DataColumn** of the "order" **DataTable**.
  - Its ordinal is 2.
  - The data type of the **DataColumn** is "string".
  - **AllowDbNull** is true.
- "OrderItem" becomes a **DataColumn** of the "order" **DataTable**.
  - Its ordinal is 3.
  - The data type of the **DataColumn** is "string".
  - **AllowDbNull** is true. (The nillable attribute is ignored.)
- "OrderItem2" becomes a **DataColumn** of the "order" **DataTable**.
  - Its ordinal is 4.
  - The data type of the **DataColumn** is "string".
  - **AllowDbNull** is false. (The nillable attribute is read.)

```
<!-- "order" DataTable -->
        <xs:element name="order">
          <xs:complexType>
            <xs:all>
              <xs:element name="orderID" minOccurs="1" type="xs:string"></xs:element>
              <xs:element name="orderAmount" minOccurs="0"  type="xs:int"
default="100"></xs:element>
              <xs:element name="orderDate" nillable="true" type="xs:string"></xs:element>
              <xs:element name="orderItem" minOccurs="0" nillable="false"
type="xs:string"></xs:element>
              <xs:element name="orderItem2" minOccurs="1" nillable="false"
type="xs:string"></xs:element>
            </xs:all>
          </xs:complexType>
        <xs:element>
```

2.3.1.1.15 <attribute> Element within <complexType>

An <attribute> element for which a **use** attribute is specified and the value is "prohibited" MUST be mapped to a DataColumn object with a **ColumnMapping** property set to **Hidden**. An <attribute> element that has a **use** attribute specified with a value that is other than "prohibited" MUST be mapped to a **DataColumn** with a **ColumnMapping** property set to **Attribute**.

If the <attribute> element contains a **ref** attribute, the referenced type as specified in [XMLSCHEMA2] MUST be used for the remainder of the rules of this section:

- The name of the **DataColumn** MUST be the name of the <attribute> element. This name and its **targetNamespace** attribute MUST be unique.
- If the <attribute> element references a type from an imported schema, the **namespace** property of the **DataColumn** MUST be the **targetNamespace** of this imported schema. Otherwise, the **namespace** of the DataTable object MUST be the **targetNamespace** attribute from the current <schema> element.
- Following are the rules that MUST be followed to determine what data type the **DataColumn** maps to depending on how the **type** attribute and **msdata:DataType** attribute are used:
  - If only the **msdata:DataType** attribute is specified, its value MUST map to a data type. The **msdata:DataType** value will either be the **namespace** and the class name or a fully qualified type name as specified in [ECMA-335:2006]. Types defined in section 2.2.3 SHOULD<3> be specified in the non-fully qualified form.
  - If the **type** attribute is present and the **msdata:DataType** attribute is present, the value of the **type** attribute MUST be the correct [XMLSCHEMA2] type for the value of the **msdata:DataType** attribute that is specified according to section 2.2.3.
  - If the **type** attribute is present and the **msdata:DataType** attribute is not present, the **DataColumn** data type MUST map to the data type according to section 2.2.2.
  - If the **type** attribute is not present, the **DataColumn** data type MUST map to "string".
- If the <attribute> element specifies a **use** attribute that is set to "default", the **DefaultValue** property of the **DataColumn** MUST be set to the value of the **value** attribute. The value MUST be serialized according to the data type to [XMLSCHEMA2] data type serialization rules that are described in section 2.2.3.
- If the <attribute> element specifies a **use** attribute that is set to "required", the **AllowDbNull** property of the **DataColumn** MUST be set to false. Otherwise, the **AllowDbNull** property of the **DataColumn** MUST be set to true.
- If the <attribute> element specifies a **use** attribute that is set to optional, the **AllowDbNull** property of the **DataColumn** MUST be set to true.
- If the <attribute> element specifies a **use** attribute that is set to fixed, the **DefaultValue** property of the **DataColumn** property MUST be set to the value of the **value** attribute. The value MUST be serialized according to the data type to [XMLSCHEMA2] data type serialization rules that are described in section 2.2.3.
- If the <attribute> element has a **use** attribute that is set to fixed, the **ReadOnly** property of the **DataColumn** MUST be set to true. If the attribute is not set to fixed, the **ReadOnly** property of the **DataColumn** MUST be set to false.
- If the <attribute> has no **use** attribute, the **AllowDbNull** property of the **DataColumn** MUST be set to true and the **DefaultValue** property of the **DataColumn** MUST be set to an empty string.
- If the **minOccurs** attribute is specified, the value MUST NOT be greater than 1:
  - If the **minOccurs** attribute is 0, the **AllowDbNull** property of the **DataColumn** is true.

    - If the **minOccurs** attribute is 1, the **AllowDbNull** property of the **DataColumn** is false.

- All <attribute> elements within the <complexType> element MUST be children of the <complexType> element and not part of any compositor element (<all>, <sequence>, or <choice>).
- The **id** attribute of the <attribute> element MUST be ignored if present.
- If the **msdata:AutoIncrement** attribute is specified, the **DataColumn AutoIncrement** property value MUST be true or false. If the attribute is not specified, the **DataColumn AutoIncrement** property value MUST be false.
- If the **AutoIncrementSeed** attribute is specified, the **DataColumn AutoIncrementSeed** property value MUST be a valid long value as specified in [XMLSCHEMA2]. If the attribute is not specified, the **DataColumn AutoIncrementSeed** property value MUST be zero.
- If the **msdata:AutoIncrementStep** attribute is specified, the **DataColumn AutoIncrementStep** property value MUST be a valid long value as specified in [XMLSCHEMA2]. If the attribute is not specified, the **DataColumn AutoIncrementStep** property value MUST be 1.
- If the **msdata:Caption** attribute is specified, the **DataColumn Caption** property value MUST be a valid string as specified in [XMLSCHEMA2]. If the attribute is not specified, the value MUST be the name of the **DataColumn** name.
- If the **msdata:Expression** attribute is specified, the **DataColumn Expression** property value MUST be a valid string as specified in [XMLSCHEMA2]. If the attribute is not specified, the **DataColumn Expression** property value is an empty string.
- If the **msdata:ReadOnly** attribute is specified, the **DataColumn ReadOnly** property value MUST be true or false. If the attribute is not specified, the **DataColumn ReadOnly** property value MUST be false.

In the following schema excerpt, the following items apply:

- "Name" becomes a **DataColumn** of the "customer" **DataTable**.
- The data type of the **DataColumn** is "string".
- **AllowDbNull** is set to false.

```
<xs:element name="customer">
  <xs:complexType>
  <xs:all>
  ….
  </xs:all>
  <xs:attribute name="name" type="xs:string"></xs:attribute>
 </xs:complexType>
</xs:element>
```

2.3.1.1.16 Elements Containing <IdentityConstraintDefinition> Elements

As defined in [XMLSCHEMA2], elements optionally contain the following <IdentityConstraintDefinition> elements to define the following types of constraints:

- **unique**
- **key**
- **keyref**

If there are no constraints, there MUST NOT be constraints mapped to the **DataSet** object.

2.3.1.1.16.1 <unique> Element

A <unique> element MUST map to a **UniqueConstraint** concept, which is part of the **DataSet** concept. When a <unique> element is found within the schema as specified in [XMLSCHEMA1] and [XMLSCHEMA2], the following rules MUST be followed:

- If a <unique> element appears within a <complexType> element that has been mapped to a DataTable, the <unique> element MUST map to a unique constraint for that **DataTable**.
- The **id** attribute MUST be ignored.
- If the **msdata:ConstraintName** attribute is not specified, it MUST be mapped to the **ConstraintName** property of the **UniqueConstraint**. If this attribute is not specified, the **name** attribute value MUST be mapped to **ConstraintName** property of the **UniqueConstraint**.
- The <selector> element MUST exist and MUST have an **xpath** attribute that maps to a **DataTable** that the unique constraint is part of.
- One or more <field> elements MUST exist. In each <field> element, as specified in [XMLSCHEMA2], there MUST contain an **xpath** attribute that maps to a DataColumn that MUST be part of the **DataTable** that is specified in the <selector> element.
- If the element contains the **msdata:PrimaryKey** attribute and the value equals true, the primary key of the **DataTable** MUST be mapped to the **DataColumn** objects that are specified by the <field> elements. If the **msdata:PrimaryKey** attribute is not specified or its value is equal to false, a **UniqueConstraint** MUST be mapped to the **DataColumn** objects that are specified by the <field> elements.
- Any **DataColumn** objects that are part of a **UniqueConstraint** object optionally can be used as part of a **ForeignKeyConstraint** object to another **DataTable**. Any **DataColumn** objects that are not part of a **UniqueConstraint** MUST NOT be a member of a **ForeignKeyConstraint**.
- Each attribute on the <unique> element that specifies the urn:schemas-microsoft-com:xml-msprop namespace MUST specify a name/value pair in the **ExtendedProperties** that exists on the **UniqueConstraint** object.

2.3.1.1.16.2 <key> Element

A <key> element in the XSD MUST map to a **UniqueConstraint** object, which is part of the **DataSet** object. When a <key> element is found within the schema as specified in [XMLSCHEMA1] and [XMLSCHEMA2], the following rules MUST be followed:

- If a <key> element appears within a <complexType> element that has been mapped to a DataTable object, the <key> element MUST map to a unique constraint that is a primary key for that **DataTable**.
- The **id** attribute MUST be ignored.
- If the **msdata:ConstraintName** attribute is not specified, it MUST be mapped to the **ConstraintName** property of the **UniqueConstraint**. If this attribute is not specified, the **name** attribute value MUST be mapped to the **ConstraintName** property of the **UniqueConstraint**.
- The <selector> element MUST exist and MUST have an **xpath** attribute that maps to a **DataTable** that the unique constraint is part of.
- One or more <field> elements MUST exist. In each <field> element, there MUST exist an **xpath** attribute that maps to a DataColumn object that MUST be part of the **DataTable** that is specified in the <selector> element.

- If the element contains the **msdata:PrimaryKey** attribute and the value equals true, the primary key of the **DataTable** MUST be mapped to the **DataColumn** objects that are specified by the <field> elements. If the **msdata:PrimaryKey** attribute is not specified or is equal to false, a **UniqueConstraint** MUST be mapped to the **DataColumn** objects that are specified by the <field> elements.
- Any **DataColumn** objects that are part of a **UniqueConstraint** object optionally can be used as part of a **ForeignKeyConstraint** object to another **DataTable**. Any **DataColumn** objects that are not part of a **UniqueConstraint** MUST NOT be a member of **ForeignKeyConstraint**.
- Each attribute on the <key> element that specifies the urn:schemas-microsoft-com:xml-msprop namespace MUST specify a name/value pair in the **ExtendedProperties** that exists on the **UniqueConstraint** object.

2.3.1.1.16.3 <keyref> Element

If a <keyref> element occurs in the XSD and the value of its **msdata:ConstraintOnly** attribute is true, or this attribute is not specified, the <keyref> element MUST be mapped to a **ForeignKeyConstraint** object and a DataRelation object in a **DataSet** object.

If the value of the **msdata:ConstraintOnly** attribute is false, the <keyref> element MUST be mapped only to the **ForeignKeyConstraint** object. The following rules MUST be followed to map the <keyref> element to a **ForeignKeyConstraint**:

- If a <keyref> element appears within a <complexType> element that has been mapped to a DataTable, the <keyref> ~~element's~~element's <selector> element MUST map to the same **DataTable**.
- The **id** attribute MUST be ignored.
- If the **msdata:ConstraintName** attribute is specified, it MUST be mapped to the **ConstraintName** property of the **UniqueConstraint**. If this attribute is not specified, the **name** attribute value MUST be mapped to the **ConstraintName** property of the **ForeignKeyConstraint**.
- The <selector> element MUST exist and MUST have an **xpath** attribute that maps to a **DataTable** that the **ForeignKeyConstraint** is part of. The **Table** property of the **ForeignKeyConstraint** MUST map to the **DataTable** that is identified by the <selector> element.
- One or more <field> elements MUST exist. In each <field> element, there MUST exist an **xpath** attribute that maps to a DataColumn that MUST be part of the **DataTable** that is specified in the <selector> element. The **Columns** property of the **ForeignKeyConstraint** MUST map to the **DataColumn** objects that are identified by the <field> elements.
- The **refer** attribute MUST be a name that links this to another existing <keyref> element or <unique> element to which this foreign key relates. The **RelatedTable** and **RelatedColumns** properties of the **ForeignKeyConstraint** MUST map to the **DataTable** and **DataColumn** objects that are specified by the other <unique> element.
- If the **msdata:UpdateRule** attribute is specified, the value MUST be None, Cascade, SetNull, or SetDefault. If the attribute is not specified, the value is set to Cascade. The **UpdateRule** property of the **ForeignKeyConstraint** MUST be set to the specified value.
- If the **msdata:DeleteRule** attribute is specified, the value MUST be None, Cascade, SetNull, or SetDefault. If the attribute is not specified, the value is set to Cascade. The **DeleteRule** property of the **ForeignKeyConstraint** MUST be set to the specified value.
- If the **msdata:AcceptRejectRule** attribute is specified, the value MUST be None, Cascade, SetNull, or SetDefault. If the attribute is not specified, the value is set to Cascade. The **AcceptRejectRule** property of the **ForeignKeyConstraint** MUST be set to the specified value.

- If the **msdata:IsNested** attribute is specified and the value equals true, the child **DataTable** MUST be nested within an <all>, <sequence>, or <choice> element of the parent **DataTable**. If **msdata:IsNested** equals false, the child **DataTable** MUST NOT be nested within the parent **DataTable**. There MUST NOT be more than one nested **Constraint** for any one child **DataTable**.
- Each attribute on the <keyref> element that specifies the namespace urn:schemas-microsoft-com:xml-msprop MUST have an attribute name that starts with fk_

Each attribute on the <keyref> element that specifies the urn:schemas-microsoft-com:xml-msprop namespace and that has an attribute name that starts with "fk_" MUST specify a name/value pair in the **ExtendedProperties** property that is specified on the **ForeignKeyConstraint** object where the name that is used is the attribute name with the starting "fk_" string removed.

If the **msdata:ConstraintOnly** attribute is equal to true, the following rules MUST be followed to map the already mapped **ForeignKeyConstraint** to the **DataRelation**:

- If the **msdata:RelationName** attribute is specified, its value MUST be mapped to the **RelationName** property of the **DataRelation**. If **msdata:RelationName** is not specified and the **msdata:ConstraintName** attribute is not specified, its value MUST be mapped to the **RelationName** property of the **DataRelation**.

  If the **msdata:ConstraintName** attribute is not specified, the **name** attribute value MUST be mapped to the **RelationName** property of the **DataRelation**.
- The **ChildColumns** property of the **DataRelation** MUST be set to the same **DataColumn** objects as the related **Columns** property of the **ForeignKeyConstraint**.
- The **ChildTable** property of the **DataRelation** MUST be set to the same **DataTable** as the **RelatedTable** property of the **ForeignKeyConstraint**.
- The **ChildKeyConstraint** property MUST be mapped to the **ForeignKeyConstraint**.
- The **ParentColumns** property of the **DataRelation** MUST be set to the same **DataColumn** objects as the related **RelatedColumns** property of the **ForeignKeyConstraint**.
- The **ParentTable** property of the **DataRelation** MUST be set to the same **DataTable** as the related **RelatedTable** property of the **ForeignKeyConstraint**.
- The **ParentKeyConstraint** property MUST be mapped to the **UniqueConstraint** that the **ForeignKeyConstraint** maps to.
- Each attribute on the <keyref> element that specifies the urn:schemas-microsoft-com:xml-msprop namespace MUST have an attribute name that starts with "fk_" or "rel".
- Each attribute on the <keyref> element that specifies the urn:schemas-microsoft-com:xml-msprop namespace and that has an attribute name that starts with "fk_" MUST specify a name/value pair in the **ExtendedProperties** that exists on the **ForeignKeyConstraint** object where the name that is used is the attribute name with the starting "fk_" string removed.
- Each attribute on the <keyref> element that specifies the urn:schemas-microsoft-com:xml-msprop namespace and that has an attribute name that starts with "rel_" MUST specify a name/value pair in the **ExtendedProperties** that exists on the **DataRelation** object where the name that is used is the attribute name with the starting "rel_" string removed.
- There MUST NOT be more than one nested **DataRelation** for any one child **DataTable**.

The following is an example of <keyref> and <unique> elements.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
```

```
  <xs:element name="NewDataSet" msdata:IsDataSet="true">
    <xs:complexType>
      <xs:choice>
        <!-- "order" DataTable -->
        <xs:element name="order">
          <xs:complexType>
            <xs:all>
              <xs:element name="orderID" minOccurs="0"  type="xs:string"></xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element >
        <!-- "orderdetail" DataTable -->
        <xs:element name="orderdetail">
          <xs:complexType>
            <xs:all>
              <xs:element name="orderID" minOccurs="0"  type="xs:string"></xs:element>
              <xs:element name="description" minOccurs="0" type="xs:string"
default="mms"></xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
    <!-- orderdetail has a FK constraint to orders.OrderID -->
    <xs:unique name = "OrderKey">
      <xs:selector xpath=".//order"/>
      <xs:field xpath="orderID"/>
    </xs:unique>
    <!-- orderdetail has a FK constraint to orders.OrderID -->
    <xs:keyref name = "OrderDetailForiegnKey" refer="OrderKey">
      <xs:selector xpath=".//orderdetail"/>
      <xs:field xpath="orderID"/>
    </xs:keyref>
  </xs:element>
</xs:schema>
```

- A "UniqueOrderID" unique constraint is created for the "orderID" **DataColumn** in the "order" **DataTable**.
- The "orderID" **DataColumn** of the "order" **DataTable** has a unique property of true.
- An "OrderDetailKey" foreign key constraint is created between the "orderID" foreign key **DataColumn** in the "orderDetail" **DataTable** and the "UniqueOrderID" key that is created in the "order" **DataTable**.

2.3.2 DiffGram Data Element

Conceptually, the **DiffGram Data** element encapsulates the XML representation of the data in the **DataSet** object. At a high-level, the **DiffGram Data** element contains the following child elements:

- A DataInstance element's rows, and the data that they contain, MUST conform to the rules that are specified in sections 2.1 and 2.2.
- A <urn:schemas-microsoft-com:xml-diffgram-v1:before> element (hereafter referred to as the "<before> element").
- A <urn:schemas-microsoft-com:xml-diffgram-v1:errors> element (hereafter referred to as the "<errors> element").

This high-level structure is defined by the following XSD.

```
<?xml version="1.0" standalone="yes"?>
<xs:schema targetNamespace="urn:schemas-microsoft-com:xml-diffgram-v1" xmlns="urn:schemas-
microsoft-com:xml-diffgram-v1" xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1" xmlns:msdata="urn:schemas-microsoft-
com:xml-msdata" xmlns:msprop="urn:schemas-microsoft-com:xml-msprop"
attributeFormDefault="qualified" elementFormDefault="qualified">
<xs:import namespace="urn:schemas-microsoft-com:xml-msdata"/>
  <xs:attribute name="id" type="xs:string"/>
<xs:attribute name="hasChanges">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Inserted"/>
      <xs:enumeration value="Modified"/>
    </xs:restriction>
  </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="hasErrors" type="xs:boolean"/>
  <xs:attribute name="Error" type="xs:string"/>
  <xs:element name="diffgram">
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="1">
        <xs:any namespace="##other" processContents="lax" minOccurs="0" />
        <xs:element name="before" minOccurs="0">
          <xs:complexType>
            <xs:choice maxOccurs="unbounded">
              <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <xs:element name="errors" minOccurs="0">
          <xs:complexType>
            <xs:choice maxOccurs="unbounded">
              <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

In the preceding XSD, the following are the definitions of the **DiffGram Data** element, the **DataInstance** element, the <before> element, and the <errors> element.

```
  <xs:element name="diffgram">
        <xs:any namespace="##other" processContents="lax" minOccurs="0" />
        <xs:element name="before" minOccurs="0">
        <xs:element name="errors" minOccurs="0">
```

Note that because the schema of the **DataInstance** element is defined by the <schema> element of the **DiffGram** (and varies per **DataSet**), it is not possible to write a single schema for the **DiffGram Data** element that properly defines the **DataInstance** element. Therefore, the preceding schema allows any content in the place of the **DataInstance** element. Producers and consumers of **DiffGrams** MUST make sure that the **DataInstance** element's rows, and the data they contain, MUST conform to the rules that are specified in sections 2.1 and 2.2 and 2.3.

The sub-sections that follow define the parts of the **DiffGram Data** element in more detail.

2.3.2.1 DataInstance Element

The **DataInstance** element contains one first-level child element per DataRow object of data in the **DataSet** ~~object's~~object's tables (hereafter, first-level child elements of the **DataInstance** element are referred to as "**DataRow** elements"). Each **DataRow** in the **DataSet** MUST be serialized by using the XML element (defined in the <schema> element of the **DiffGram**) that corresponds to its DataTable object.

The **DataRow** values in the **DataInstance** element MUST be the current **DataRow** values in the **DataSet** at the time the **DataSet** is serialized into the **DiffGram** structure. That is, the **DataInstance** element MUST reflect all changes made to the **DataRow** values within the **DataSet** and the new rows that were added after the **DataSet** was last synchronized with its data source. Rows that were deleted from the **DataSet** MUST NOT appear in the **DataInstance** element.

The following is an example of a **DataInstance** element that can appear in a **DiffGram**.

```
<SalesDS>
<Customers diffgr:id="Customers1" msdata:rowOrder="0" diffgr:hasChanges="inserted">
  <CustId>A</CustId>
  <CustName>C1</CustName>
</Customers>
<Customers diffgr:id="Customers2" msdata:rowOrder="1">
  <CustId>B</CustId>
  <CustName>C2</CustName>
</Customers>
<Customers diffgr:id="Customers3" msdata:rowOrder="2" diffgr:hasChanges="modified">
  <CustId>C</CustId>
  <CustName>C3</CustName>
</Customers>
<Customers diffgr:id="Customers5" msdata:rowOrder="4" diffgr:hasErrors="true">
  <CustId>E</CustId>
  <CustName>C5</CustName>
</Customers>
</SalesDS>
```

The following rules apply to the **DataRow** elements within the **DataInstance** element:

- Each **DataRow** element MUST have a urn:schemas-microsoft-com:xml-diffgram-v1:id attribute. The string value of this attribute acts as the **DataRow** identifier within the scope of the DiffGram Data element. The **DataInstance** element MUST NOT have two **DataRow** elements with the same value for the urn:schemas-microsoft-com:xml-diffgram-v1:id attribute.
- Each **DataRow** element MUST have a urn:schemas-microsoft-com:xml-msdata:rowOrder attribute whose value MUST be an integer that specifies the 0-based ordinal position of the **DataRow** that the element represents within its **DataTable**.
- Row elements that represent rows that were added or modified in the **DataSet** after it was last synchronized with its data source MUST have the urn:schemas-microsoft-com:xml-diffgram-v1:hasChanges attribute. This ~~attribute's~~attribute's value MUST be "inserted" for rows that were added to the **DataSet** after it was last synchronized with its data source, and the value MUST be "modified" for rows that were changed after the **DataSet** was last synchronized with its data source.
- Only **DataRow** elements that represent rows that are associated with error information that is included in the <errors> element of the **DiffGram** MUST have the urn:schemas-microsoft-com:xml-diffgram-v1:hasErrors attribute set to "true".
- Each **DataRow** element that is in a DataRelation as a child MUST have a urn:schemas-microsoft-com:xml-diffgram-v1:parentId attribute where the value is equal to the **id** attribute of the parent **DataRow**.
- All DataColumn objects for which the **ColumnMapping** property equals Attribute MUST write out the name of the **DataColumn** as the attribute name and the value of that **DataColumn** within the **DataRow** element.
- All **DataColumn** objects for which the **ColumnMapping** property equals Element MUST write out the name of the **DataColumn** as the child element of the **DataRow** element along with the corresponding value of that **DataColumn** within the **DataRow** element.

- For all **DataColumn** objects for which the **ColumnMapping** property equals Hidden, the **msdata:hidden[ColumnName]** attribute MUST be used. For example, <Customers diffgr:id="Customers1" msdata:hiddenContactTitle="Owner"> would specify a hidden column named ContactTitle.
- Only if the schema for this data instance is nested MUST the data instance be nested as well. The following is an example of this.

```
<Orders diffgr:id="Orders3" msdata:rowOrder="2" diffgr:hasChanges="inserted">
        <Id>1</Id>
           <OrderDetails diffgr:id="OrderDetails4" msdata:rowOrder="3"
diffgr:hasChanges="inserted">
        <Id>10</Id>
          <OrdersId>1</OrdersId>
        </OrderDetails>
 </Orders>
```

2.3.2.2 <before> Element

The <before> element contains the original values of rows that were modified or deleted in the **DataSet** object after it was originally loaded. For each modified or deleted DataRow object, the <before> element MUST contain a first-level child element that serializes the original values for the **DataRow** as it was first loaded into the **DataSet** (before any modifications were made).

As is true for the DataInstance element, each **DataRow** MUST be serialized by using the XML element (defined in the <schema> element of the **DiffGram**) that corresponds to its DataTable. Hereafter, first-level child elements of the <before> element are referred to as "before **DataRow** elements".

Rows that were not changed or deleted in the **DataSet** after it was last synchronized with its data source MUST NOT appear in the <before> element. Rows that were added to the **DataSet** after it was last synchronized with its data source MUST NOT appear in the <before> element.

The following is an example of a <before> element from a **DiffGram** structure.

```
<diffgr:before>
<Customers diffgr:id="Customers3" msdata:rowOrder="2">
  <CustId>C</CustId>
  <CustName>C33</CustName>
</Customers>
<Customers diffgr:id="Customers4" msdata:rowOrder="3">
  <CustId>D</CustId>
  <CustName>C4</CustName>
</Customers>
</diffgr:before>
```

The following rules apply to the before **DataRow** elements within the <before> element:

- Each <before> **DataRow** element MUST have a urn:schemas-microsoft-com:xml-diffgram-v1:id attribute. The string value of this attribute acts as the **DataRow** identifier within the scope of the DiffGram Data element. The <before> element MUST NOT have two before **DataRow** elements that have the same value for the urn:schemas-microsoft-com:xml-diffgram-v1:id attribute.
- <Before> **DataRow** elements MUST NOT have the urn:schemas-microsoft-com:xml-diffgram-v1:hasChanges attribute set.
- Each <before> **DataRow** element MUST have a urn:schemas-microsoft-com:xml-msdata:rowOrder attribute whose integer value specifies the 0-based ordinal position of the **DataRow** that the element represents within its **DataTable** object.

- Every **DataRow** element in a **DataInstance** element of a **DiffGram** that has a urn:schemas-microsoft-com:xml-diffgram-v1:hasChanges attribute that is set to "modified" MUST have a corresponding before **DataRow** element in the <before> element of the **DiffGram** that has the same value for the urn:schemas-microsoft-com:xml-diffgram-v1:id attribute. In these cases, the value of the urn:schemas-microsoft-com:xml-msdata:rowOrder attribute on the <before> **DataRow** element MUST be ignored.
- Only <before> **DataRow** elements that represent rows that are associated with error information that is included in the <errors> element of the **DiffGram** MUST have the urn:schemas-microsoft-com:xml-diffgram-v1:hasErrors attribute set to true.

2.3.2.3 <errors> Element

The <errors> element is used to serialize application-specified error information that is associated with rows in a **DataSet** object. Similar to the DataInstance element and <before> element, error information for a given DataRow object MUST be serialized by using the XML element (defined in the <schema> element of the **DiffGram** structure) that corresponds to the DataTable of the **DataRow**. Each first-level child element within the <errors> element (hereafter referred to as an "error **DataRow** element") represents the error information associated with a single **DataRow**.

The following rules apply to the error **DataRow** elements within the <errors> element:

- Each error **DataRow** element MUST have a urn:schemas-microsoft-com:xml-diffgram-v1:id attribute. The string value of this attribute acts as the **DataRow** identifier within the scope of the DiffGram Data element. The <errors> element MUST NOT have two error **DataRow** elements that have the same value for the urn:schemas-microsoft-com:xml-diffgram-v1:id attribute.
- For every error **DataRow** element, the **DiffGram** ~~structure's~~structure's **DataInstance** element or <before> element MUST contain a corresponding **DataRow** element or before **DataRow** element that has the same urn:schemas-microsoft-com:xml-diffgram-v1:id attribute value and whose urn:schemas-microsoft-com:xml-diffgram-v1:hasErrors attribute is set to true.

The following is an example of an <errors> element that can appear in a **DiffGram**.

```
<diffgr:errors>
  <Customers diffgr:id="Customers5" diffgr:Error="This customer data is not correct">
  <CustName diffgr:Error="This customer DataRow DataColumn value is not correct" />
  </Customers>
</diffgr:errors>
```

Within the <errors> element, the urn:schemas-microsoft-com:xml-diffgram-v1:Error attribute is used to specify error information. If there is an error, this attribute MUST appear on a **DataRow** element or on a child element within a **DataRow** element and it MUST have a non-empty value. When the attribute appears on a child element within a **DataRow** element, the error information that is specified applies to the particular DataColumn that the child element represents.

3 Structure Examples

Following is a comprehensive example of a **DataSet DiffGram** XML structure that contains the following:

- Tables, rows, columns (Schema and data)
- Relationships
- Constraints
- Row error information
- Nested schema and instances
- Changes to the data in the **Before** section (showing the data in different states, including modified, inserted, and deleted).

```
<?xml version="1.0" encoding="utf-8" ?>
<DataSet xmlns="http://tempuri.org/">
  <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
                             xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
      <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="ProductCategories">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Id" type="xs:int" minOccurs="0" />
                <xs:element name="Products" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <msdata:Relationship name="ProductCategories_Products"
msdata:parent="ProductCategories"
                                           msdata:child="Products" msdata:parentkey="Id"
msdata:childkey="ProductCategoriesId" />
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="Id" type="xs:int" />
                      <xs:element name="ProductCategoriesId" type="xs:int" minOccurs="0" />
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Orders">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Id" type="xs:int" minOccurs="0" />
                <xs:element name="OrderDetails" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="Id" type="xs:int" />
                      <xs:element name="OrdersId" type="xs:int" minOccurs="0" />
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Customer">
            <xs:complexType>
              <xs:sequence>
```

```
              <xs:element name="Id" type="xs:int" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="CustomerDetails">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Id" type="xs:int" />
              <xs:element name="CustomerId" type="xs:int" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Region">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Id" type="xs:int" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="RegionDetails">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Id" type="xs:int" />
              <xs:element name="RegionId" type="xs:int" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="OtherTable">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Id" type="xs:int" minOccurs="0" msdata:Ordinal="0" />
              <xs:element name="SqlXmlColumn" msdata:DataType="System.Data.SqlTypes.SqlXml"
                          type="xs:anyType" minOccurs="0" msdata:Ordinal="1" />
            </xs:sequence>
            <xs:attribute name="DateTimeOffSetColumn"
msdata:DataType="System.DateTimeOffset"
                          type="xs:anyType" use="prohibited" />
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
    <xs:unique name="Constraint1" msdata:PrimaryKey="true">
      <xs:selector xpath=".//Products" />
      <xs:field xpath="Id" />
    </xs:unique>
    <xs:unique name="OrderDetails_Constraint1" msdata:ConstraintName="Constraint1"
msdata:PrimaryKey="true">
      <xs:selector xpath=".//OrderDetails" />
      <xs:field xpath="Id" />
    </xs:unique>
    <xs:unique name="Orders_Constraint1" msdata:ConstraintName="Constraint1">
      <xs:selector xpath=".//Orders" />
      <xs:field xpath="Id" />
    </xs:unique>
    <xs:unique name="Customer_Constraint1" msdata:ConstraintName="Constraint1">
      <xs:selector xpath=".//Customer" />
      <xs:field xpath="Id" />
    </xs:unique>
    <xs:unique name="CustomerDetails_Constraint1" msdata:ConstraintName="Constraint1"
msdata:PrimaryKey="true">
      <xs:selector xpath=".//CustomerDetails" />
      <xs:field xpath="Id" />
    </xs:unique>
    <xs:unique name="RegionDetails_Constraint1" msdata:ConstraintName="Constraint1"
msdata:PrimaryKey="true">
      <xs:selector xpath=".//RegionDetails" />
      <xs:field xpath="Id" />
    </xs:unique>
    <xs:keyref name="Customer_CustomerDetails" refer="Customer_Constraint1">
      <xs:selector xpath=".//CustomerDetails" />
```

```
        <xs:field xpath="CustomerId" />
      </xs:keyref>
      <xs:keyref name="Order_OrderDetail" refer="Orders_Constraint1" msdata:IsNested="true">
        <xs:selector xpath=".//OrderDetails" />
        <xs:field xpath="OrdersId" />
      </xs:keyref>
    </xs:element>
    <xs:annotation>
      <xs:appinfo>
        <msdata:Relationship name="Region_RegionDetail" msdata:parent="Region"
msdata:child="RegionDetails"
                                                          msdata:parentkey="Id"
msdata:childkey="RegionId" />
      </xs:appinfo>
    </xs:annotation>
  </xs:schema>
  <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
                   xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    <NewDataSet>
      <ProductCategories diffgr:id="ProductCategories1" msdata:rowOrder="0">
        <Id>3</Id>
        <Products diffgr:id="Products2" msdata:rowOrder="1">
          <Id>33</Id>
          <ProductCategoriesId>3</ProductCategoriesId>
        </Products>
        <Products diffgr:id="Products3" msdata:rowOrder="2" diffgr:hasChanges="inserted">
          <Id>16</Id>
          <ProductCategoriesId>3</ProductCategoriesId>
        </Products>
      </ProductCategories>
      <ProductCategories diffgr:id="ProductCategories2" msdata:rowOrder="1">
        <Id>4</Id>
      </ProductCategories>
      <ProductCategories diffgr:id="ProductCategories3" msdata:rowOrder="2"
diffgr:hasChanges="inserted">
        <Id>50</Id>
        <Products diffgr:id="Products4" msdata:rowOrder="3" diffgr:hasChanges="inserted">
          <Id>100</Id>
          <ProductCategoriesId>50</ProductCategoriesId>
        </Products>
      </ProductCategories>
      <Orders diffgr:id="Orders1" msdata:rowOrder="0">
        <Id>2</Id>
        <OrderDetails diffgr:id="OrderDetails2" msdata:rowOrder="1">
          <Id>31</Id>
          <OrdersId>2</OrdersId>
        </OrderDetails>
        <OrderDetails diffgr:id="OrderDetails3" msdata:rowOrder="2"
diffgr:hasChanges="inserted">
          <Id>12</Id>
          <OrdersId>2</OrdersId>
        </OrderDetails>
      </Orders>
      <Orders diffgr:id="Orders2" msdata:rowOrder="1">
        <Id>3</Id>
      </Orders>
      <Orders diffgr:id="Orders3" msdata:rowOrder="2" diffgr:hasChanges="inserted">
        <Id>1</Id>
        <OrderDetails diffgr:id="OrderDetails4" msdata:rowOrder="3"
diffgr:hasChanges="inserted">
          <Id>10</Id>
          <OrdersId>1</OrdersId>
        </OrderDetails>
      </Orders>
      <Customer diffgr:id="Customer1" msdata:rowOrder="0">
        <Id>5</Id>
      </Customer>
      <Customer diffgr:id="Customer2" msdata:rowOrder="1">
        <Id>6</Id>
      </Customer>
```

```
      <Customer diffgr:id="Customer3" msdata:rowOrder="2" diffgr:hasChanges="inserted">
        <Id>25</Id>
      </Customer>
      <CustomerDetails diffgr:id="CustomerDetails2" msdata:rowOrder="1">
        <Id>35</Id>
        <CustomerId>5</CustomerId>
      </CustomerDetails>
      <CustomerDetails diffgr:id="CustomerDetails3" msdata:rowOrder="2"
diffgr:hasChanges="inserted">
        <Id>18</Id>
        <CustomerId>5</CustomerId>
      </CustomerDetails>
      <CustomerDetails diffgr:id="CustomerDetails4" msdata:rowOrder="3"
diffgr:hasChanges="inserted">
        <Id>50</Id>
        <CustomerId>25</CustomerId>
      </CustomerDetails>
      <Region diffgr:id="Region1" msdata:rowOrder="0">
        <Id>10</Id>
      </Region>
      <Region diffgr:id="Region2" msdata:rowOrder="1">
        <Id>11</Id>
      </Region>
      <Region diffgr:id="Region3" msdata:rowOrder="2" diffgr:hasChanges="inserted">
        <Id>324</Id>
      </Region>
      <RegionDetails diffgr:id="RegionDetails2" msdata:rowOrder="1">
        <Id>40</Id>
        <RegionId>10</RegionId>
      </RegionDetails>
      <RegionDetails diffgr:id="RegionDetails3" msdata:rowOrder="2"
diffgr:hasChanges="inserted">
        <Id>22</Id>
        <RegionId>10</RegionId>
      </RegionDetails>
      <RegionDetails diffgr:id="RegionDetails4" msdata:rowOrder="3"
diffgr:hasChanges="inserted">
        <Id>110</Id>
        <RegionId>324</RegionId>
      </RegionDetails>
      <OtherTable diffgr:id="OtherTable1" msdata:rowOrder="0" diffgr:hasChanges="modified"
                  diffgr:hasErrors="true" msdata:hiddenDateTimeOffSetColumn="2009-09-
27T11:39:11.0671954-07:00">
        <Id>1</Id>
        <SqlXmlColumn>
          <foo>
            <MyValue>Christro</MyValue>
          </foo>
        </SqlXmlColumn>
      </OtherTable>
      <OtherTable diffgr:id="OtherTable3" msdata:rowOrder="2"
                  msdata:hiddenDateTimeOffSetColumn="2009-05-13T11:39:11.0641954-07:00">
        <Id>1</Id>
        <SqlXmlColumn>
          <foo>
            <MyValue>Steveob</MyValue>
          </foo>
        </SqlXmlColumn>
      </OtherTable>
    </NewDataSet>
    <diffgr:before>
      <Products diffgr:id="Products1" diffgr:parentId="ProductCategories1"
msdata:rowOrder="0">
        <Id>14</Id>
        <ProductCategoriesId>3</ProductCategoriesId>
      </Products>
      <OrderDetails diffgr:id="OrderDetails1" diffgr:parentId="Orders1" msdata:rowOrder="0">
        <Id>11</Id>
        <OrdersId>2</OrdersId>
      </OrderDetails>
```

```
      <CustomerDetails diffgr:id="CustomerDetails1" msdata:rowOrder="0">
        <Id>15</Id>
        <CustomerId>5</CustomerId>
      </CustomerDetails>
      <RegionDetails diffgr:id="RegionDetails1" msdata:rowOrder="0">
        <Id>20</Id>
        <RegionId>10</RegionId>
      </RegionDetails>
      <OtherTable diffgr:id="OtherTable1" msdata:rowOrder="0"
                  msdata:hiddenDateTimeOffSetColumn="2009-08-13T11:39:11.0611954-07:00">
        <Id>1</Id>
        <SqlXmlColumn>
          <foo>
            <MyValue>Christro</MyValue>
          </foo>
        </SqlXmlColumn>
      </OtherTable>
      <OtherTable diffgr:id="OtherTable2" msdata:rowOrder="1"
                  msdata:hiddenDateTimeOffSetColumn="2009-09-13T11:39:11.0631954-07:00">
        <Id>1</Id>
        <SqlXmlColumn>
          <foo>
            <MyValue>aconrad</MyValue>
          </foo>
        </SqlXmlColumn>
      </OtherTable>
    </diffgr:before>
    <diffgr:errors>
      <OtherTable diffgr:id="OtherTable1" diffgr:Error="RowError">
        <DateTimeOffSetColumn diffgr:Error="ColumnError" />
      </OtherTable>
    </diffgr:errors>
  </diffgr:diffgram>
</DataSet >
```

4 Security Considerations

None.

5 Appendix A: 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.

This document specifies version-specific details in the Microsoft .NET Framework. For information about which versions of .NET Framework are available in each released Windows product or as supplemental software, see ~~**.NET Framework**~~. [MS-NETOD] section 4.

- Microsoft .NET Framework 1.0
- Microsoft .NET Framework 1.1
- Microsoft .NET Framework 2.0
- Microsoft .NET Framework 3.5
- Microsoft .NET Framework 4.0
- Microsoft .NET Framework 4.5
- Microsoft .NET Framework 4.6

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.

<1> Section 2.3: **DataSet** uses only those prefixes that are specified in the table.

<2> Section 2.3.1.1.14: The <simpleType> element within complexTypes: **DataSet** always writes out the non-fully qualified form for types that are described in section 2.2.3.

<3> Section 2.3.1.1.15: **DataSet** always writes out the non-fully qualified form for types that are described in section 2.2.3.

6 Change Tracking

This section identifies No table of changes that were made to thisis available. The document is either new or has had no changes since theits 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
5 Appendix A: Product Behavior	Added .NET Framework 4.6 to the list of applicable products.	Y	Content update.

7 Index

.

<

A

C

D

E

F

G

H

I

L