[MS-SSDL]:
Store Schema Definition Language File Format Structure Specification

---

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.
- **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
09/03/2010	0.1	New	Released new document.
02/09/2011	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/07/2011	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/03/2011	0.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the structure and semantics of the store schema definition language (SSDL) for the **Entity Data Model (EDM)**. SSDL is a language based on XML that can be used for defining storage models by using the EDM.

SSDL is used to describe storage metadata and schema using concepts from the entity-relationship model.

Entities are instances of entity types (such as Customer or Employee) that are richly structured records with a key. The structure of an entity type is provided by its properties. An entity key is formed from a subset of the properties of the entity type. The key (such as CustomerId or EmployeeId) is a fundamental concept to uniquely identify and persist entity instances and to enable entity instances to participate in relationships or **associations**.

Entities are grouped into entity sets; for example, the Customers entity set is a set of Customer instances.

Associations (sometimes referred to as relationships) are instances of association types. Association types are used to specify a named relationship between two entity types. Thus, an association is a named relationship between two or more entities. Associations are grouped into association sets.

Unlike conceptual schema definition language (CSDL), SSDL does not support the following concepts:

- Complex types
- Inheritance
- Navigation properties
- Function imports

SSDL does include the concept of functions that allow store functions and stored procedures to be defined and represented as part of the store metadata.

Entity sets and association sets are grouped into one or more entity containers. Entity containers are conceptually similar to databases; however, because entity types and association types are declared outside an entity container, they can be re-used across entity containers.

Additionally, although CSDL is independent of any particular database, an instance of SSDL is tied to a particular database. In order to specify SSDL, it is necessary to reference the EDM-enabled ADO.NET **provider** in addition to a **provider manifest token**. The provider manifest token identifies a **provider manifest** that is built into the provider and specifies which data types and database functions are supported by the provider. The provider manifest also specifies which primitive EDM type each store data type maps to. For more information about provider manifests, see [MSDN-PMS].

1.1 Glossary

The following terms are defined in [MS-GLOS]:

XML namespace

The following terms are specific to this document:

alias qualified name: A QualifiedName attribute that is used to refer to a structural type, with the exception of the **namespace** that is being replaced by the namespace's alias. For example, if an **entity** type called "Person" is defined in the "Model.Store" namespace, and if that namespace has been given the alias "Self", the alias qualified name for the "Person" entity type is "Self.Person".

annotation: Any custom, application-specific extension to an instance of SSDL that is made by using custom attributes and elements that are not a part of this SSDL specification.

association: A named independent relationship between two EntityType definitions. Associations in the **Entity Data Model (EDM)** are first-class concepts and are always bi-directional.

cardinality: The measure of the number of elements in a set.

collection: An element that is used when a Function element is declared whose parameter or return type is not a single value but many. For example, a **Function** element may return a collection of varchar, that is, collection(varchar).

entity: An instance of an **EntityType** element that has a unique identity and an independent existence, and is an operational unit of consistency.

Entity Data Model (EDM): An entity-relationship data model.

facet: An element that provides information that specifies the usage of a type. For example, a user can use the Precision facet to define the precision of a DateTime property.

namespace: A name that is defined on the schema and subsequently used to prefix identifiers to form the **namespace qualified name** of an **EntityType** or Association type.

namespace qualified name: A **QualifiedName** that is used to refer to **EntityType** elements or **Association** type elements by using the name of the **namespace**, followed by a period, followed by the name of the **EntityType** or **Association** type.

provider: The **EDM**-enabled provider that is used to connect to the database.

provider manifest: A static XML-based declaration of all the store data types and the functions that an **EDM**-enabled **provider** supports against the store. This manifest is embedded into the provider assembly that supports the EDM.

provider manifest token: A simple name or token that identifies a specific **provider manifest**. If multiple provider manifests are specified by a provider, this token could be as simple as a version number that identifies a particular provider manifest.

store type: A data type that is specific to the database. The type must be supported by the **provider** that supports the **EDM**.

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

1.2 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MC-CSDL] Microsoft Corporation, "Conceptual Schema Definition File Format".

[MS-MSL] Microsoft Corporation, "Mapping Specification Format".

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

[XML1.0] Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[XML Namespaces1.0] Bray, T., Hollander, D., and Layman, A., "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/REC-xml-names

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "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

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MSDN-PMS] Microsoft Corporation, "Provider Manifest Specification", http://msdn.microsoft.com/en-us/library/ee828423.aspx

1.3 Structure Overview (Synopsis)

Store schema definition language (SSDL) is an XML-based file format that describes the store schema in the Entity Data Model (EDM) and that is based on standards defined in [XML1.0] and [XMLSCHEMA1]. The root of SSDL is a Schema element. Below that root, the following subelements are supported: EntityType, Association, Function, and EntityContainer. The **EntityContainer** element conceptually represents a data source, and it can contain EntitySet, AssociationSet, and **Function** subelements.

Conceptually, an SSDL file has an overall structure that resembles the following.

```
<Schema>

   <EntityType/>
   <EntityType/>

   <Association/>
   <Association/>
   <Function/>
   <Function/>
   <EntityContainer>
```

```
        <EntitySet/>
        <EntitySet/>

        <AssociationSet/>
        <AssociationSet/>

    </EntityContainer>
    <EntityContainer/>
</Schema>
```

Note This is not a detailed specification. It is meant only to provide a visual overview.

1.4 Relationship to Protocols and Other Structures

Conceptual schema definition language (CSDL), mapping specification language (MSL), and store schema definition language (SSDL) are related schema definition languages that form the basis for the Entity Data Model (EDM). The EDM is a specification for defining conceptual data models. Applications can use the EDM to define a conceptual model that describes the entity, relationships, and sets required in the domain that is served by the application.

The following figure shows the relationship among the three structures.

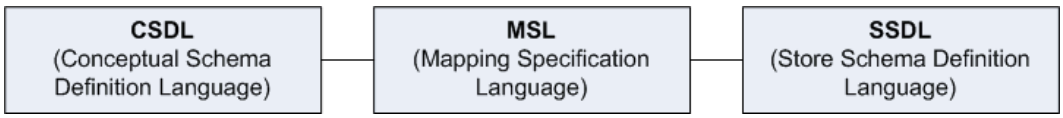

Figure 1: MSL defines the mapping between CSDL and SSDL

1.5 Applicability Statement

SSDL is an XML format that describes the structure and semantics of the Entity Data Model (EDM) schemas. Identifiers, such as names and **namespaces**, are all case-sensitive.

1.6 Versioning and Localization

This document describes the schema for SSDL version 2.0. There are no version-to-version differences between SSDL version 1.0 and SSDL version 2.0.

1.7 Vendor-Extensible Fields

SSDL supports application-specific customization and extension through the use of **annotations**. These annotations enable applications to embed application-specific or vendor-specific information into SSDL. The format does not specify how to process these custom-defined structures or how to distinguish structures from multiple vendors or layers. Parsers of the SSDL may ignore annotations that are not expected or not understood.

Annotations can be of two types: AnnotationAttribute and AnnotationElement.

The reserved **XML namespaces** for SSDL are as follows:

http://schemas.microsoft.com/ado/2006/04/edm/ssdl

http://schemas.microsoft.com/ado/2009/02/edm/ssdl

http://schemas.microsoft.com/ado/2009/11/edm/ssdl

http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator

2 Structures

2.1 Elements

2.1.1 Schema

The **Schema** element is the top-level SSDL construct that allows the creation of namespaces.

The contents of a namespace can be specified by one or more **Schema** instances. The identifiers that are used to name types MUST be unique within a namespace. For example, an EntityType element cannot have the same name as another **EntityType** element within the same namespace. The namespace forms a part of the type's fully qualified name and is a logical grouping of **EntityType** elements, Association elements, and Function elements. The **Alias** attribute is a simple identifier that is used as a short name for a namespace. This attribute is declared along with the namespace.

The following is the XML schema definition of the **Schema** element.

```
<xs:element name="Schema" type="edm:TSchema"/>
<xs:complexType name="TSchema">
  <xs:sequence>
    <xs:group ref="edm:GSchemaBodyElements" minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Namespace" type="edm:TQualifiedName" use="required" />
  <xs:attribute name="Alias" type="edm:TSimpleIdentifier" use="optional" />
  <xs:attribute name="Provider" type="edm:TSimpleIdentifier" use="required" />
  <xs:attribute name="ProviderManifestToken" type="edm:TSimpleIdentifier" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:group name="GSchemaBodyElements">
  <xs:choice>
    <xs:element name="Association" type="edm:TAssociation" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="EntityType" type="edm:TEntityType" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element ref="edm:EntityContainer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="Function" type="edm:TFunction" minOccurs="0" maxOccurs="unbounded" />
  </xs:choice>
</xs:group>
```

The following rules apply to the **Schema** element:

- The SSDL document MUST have the **Schema** element as its root element.
- A schema namespace MUST NOT be "System", "Transient", or "Edm".
- A schema definition can span more than one SSDL document.

2.1.2 EntityType

The **EntityType** element specifies the type and structure of entities that use it as an underlying type. **EntityType** elements specify the conceptual concerns within a data model, such as Customer, Order and Supply (to use the example of a typical line-of-business system).

An entity represents one particular instance of the **EntityType** element, such as a specific customer or a specific order. In the context of SSDL, an **EntityType** corresponds to an extent on the database (either a table or a view).

An **EntityType** has a **Name** attribute, a payload that consists of one or more declared properties, and an EntityKey element that specifies the set of properties whose values uniquely identify an entity within an entity set. The type of a Property element MUST be a valid **store type**.

The following is the XML schema definition of the **EntityType** element.

```
<xs:complexType name="TEntityType">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:element name="Key" type="edm:TEntityKeyElement" minOccurs="0" maxOccurs="1"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="Property" type="edm:TEntityProperty" minOccurs="0"
maxOccurs="unbounded" />
    </xs:choice>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

The following rule applies to the **EntityType** element:

- The **Name** attribute of an **EntityType** MUST be unique across all **EntityType** types, association types, and functions that are specified in the same namespace.

2.1.3 Property

The **Property** element specifies the properties of an EntityType element. An **EntityType** element can have one or more **Property** elements. A property description consists of the name and type of the property and a set of **facets**, such as Nullable or Default. Facets are optional, and they specify the further behavior of a given type.

The following is the XML schema definition of the **Property** element.

```
<xs:complexType name="TEntityProperty">
  <xs:sequence>
    <xs:group ref="edm:GEmptyElementExtensibility" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attributeGroup ref="edm:TCommonPropertyAttributes"/>
  <xs:attribute name="StoreGeneratedPattern" type="edm:TStoreGeneratedPattern" use="optional"
/>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:attributeGroup name="TCommonPropertyAttributes">
  <xs:attribute name="Name" type="edm:TSimpleIdentifier" use="required" />
  <xs:attribute name="Type" type="edm:TPropertyType" use="required" />
  <xs:attribute name="Nullable" type="xs:boolean" default="true" use="optional" />
  <xs:attribute name="DefaultValue" type="xs:string" use="optional" />
  <!-- Start Facets -->
  <xs:attribute name="MaxLength" type="edm:TMaxLengthFacet" use="optional" />
  <xs:attribute name="FixedLength" type="edm:TIsFixedLengthFacet" use="optional" />
  <xs:attribute name="Precision" type="edm:TPrecisionFacet" use="optional" />
  <xs:attribute name="Scale" type="edm:TScaleFacet" use="optional" />
```

```
    <xs:attribute name="Unicode" type="edm:TIsUnicodeFacet" use="optional" />
    <xs:attribute name="Collation" type="edm:TCollationFacet" use="optional" />
    <!--End Facets -->
  </xs:attributeGroup>
  <xs:simpleType name="TStoreGeneratedPattern">
    <xs:restriction base="xs:token">
      <xs:enumeration value="None" />
      <xs:enumeration value="Identity" />
      <xs:enumeration value="Computed" />
    </xs:restriction>
  </xs:simpleType>
```

The following rules apply to the **Property** element:

- The **Type** attribute MUST be a valid store type.
- The **Name** attribute of each **Property** element MUST be unique within a given **EntityType** element.
- The following facets are optional to specify on the **Property** element:

  - MaxLength
  - FixedLength
  - Precision
  - Scale
  - Unicode
  - Collation

  The provider that provides EDM support MAY choose to implement support for these facets, their default values, and so on. The set of facets that a provider can support depends on the base EDM type of the store type, as specified in the provider manifest. For more details about the set of facets that are valid for each EDM primitive type, see section 2.2 of [MC-CSDL].
- The **Property** element MUST contain no more than one Documentation element.

2.1.4 EntityKey

The **EntityKey** element specifies which Property elements form a key that can uniquely identify instances of an EntityType element. Any set of non-nullable, provider type declared properties can serve as the key.

The following is the XML schema definition of the **EntityKey** element.

```
<xs:complexType name ="TEntityKeyElement">
  <xs:sequence>
    <xs:element name="PropertyRef" type="edm:TPropertyRef" minOccurs="1"
maxOccurs="unbounded" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

2.1.5 PropertyRef

The **PropertyRef** element specifies a reference to a declared property of an EntityType. The **Name** attribute refers to the name of a Property that is specified in the declaring **EntityType**.

The following is the XML schema definition of the **PropertyRef** element.

```
<xs:complexType name="TPropertyRef">
  <xs:sequence>
    <xs:group ref="edm:GEmptyElementExtensibility" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="Name" type="edm:TSimpleIdentifier" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:group name="GEmptyElementExtensibility">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:group>
```

2.1.6 Association

The **Association** element specifies a peer-to-peer relationship between participating EntityType elements and can support different multiplicities at the two ends.

The following is the XML schema definition of the **Association** element.

```
<xs:complexType name="TAssociation">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:element name="End" type="edm:TAssociationEnd" minOccurs="2" maxOccurs="2"/>
    <xs:element name="ReferentialConstraint" type="edm:TConstraint" minOccurs="0"
maxOccurs="1" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

The following rules apply to the **Association** element:

- The **Name** attribute MUST be unique among all **EntityType**, **Association**, and Function names that are defined in the namespace in which **Association** is defined.
- An **Association** element MUST have exactly one ReferentialConstraint element specified.
- An **Association** element can contain any number of AnnotationAttribute attributes. The full names of the **AnnotationAttribute** attributes MUST be unique.
- An **Association** element can contain any number of AnnotationElement elements.

An example of an association is the relationship between Customer and Order entities. This relationship has the following characteristics:

- **Multiplicity**: Each Order entity is associated with exactly one Customer entity. Every Customer has zero or more Orders.
- **Operational behavior**: When an Order entity that has one or more OrderLines is deleted, the corresponding OrderLines are also deleted. This behavior is known as an OnDelete cascade.

2.1.7 AssociationEnd

The **AssociationEnd** element specifies the entity types that are related in an association, specifies the roles of those entity types in the association, and specifies the **cardinality** rules for each end of the association. Every association includes two **AssociationEnd** elements.

For a given Association, the **AssociationEnd** element specifies one side of the relationship. **AssociationEnd** specifies what type is participating in the relationship, specifies the multiplicity or the cardinality, and specifies whether there are any operation associations, such as cascading delete behavior.

The following is the XML schema definition of the **AssociationEnd** element.

```
<xs:complexType name="TAssociationEnd">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:group ref="edm:TOperations" minOccurs="0" maxOccurs="unbounded" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Type" type="edm:TQualifiedName" use="required" />
  <xs:attribute name="Role" type="edm:TSimpleIdentifier" use="optional" />
  <xs:attribute name="Multiplicity" type="edm:TMultiplicity" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

The following rules apply to the **AssociationEnd** element:

- The **Type** attribute MUST be a valid EntityType name.
- The **Type** attribute MUST be either a **namespace qualified name** or an **alias qualified name** of an **EntityType** element that is in scope.
- The **AssociationEnd** element MUST contain no more than one OnDelete element.

2.1.8 OnDelete

The **OnDelete** element is a trigger that is associated with a relationship. The action is performed on one end of the relationship when the state of the other side of the relationship changes.

OnDelete operational behavior can be specified at any end of the relationship, provided that the value of the multiplicity of that end is 1 or 0..1.

The following is the XML schema definition of the **OnDelete** element.

```
<xs:group name="TOperations">
  <xs:choice>
    <xs:element name="OnDelete" type="edm:TOnAction" maxOccurs="1" minOccurs="0" />
  </xs:choice>
</xs:group>
```

The following rules apply to the **OnDelete** element:

- The multiplicity value of the corresponding AssociationEnd element MUST be 1 or 0..1.
- Any subelements of **OnDelete** MUST appear in the following sequence:
  - Documentation
  - AnnotationElement

2.1.9 ReferentialConstraint

The **ReferentialConstraint** element is a constraint on a given association. This element specifies referential constraints through a Principal and Dependent end.

In the EDM, **ReferentialConstraint** elements can exist between the key of one entity type and the property (or properties) of another, associated entity type. (These two entity types are in a principal-to-dependent relationship, which can be considered a type of parent-child relationship.)

When defining a **ReferentialConstraint** element, the **Role** attribute MUST be used to indicate which end of the association is the principal and which end of the relationship is the dependent. Thus, the **ReferentialConstraint** MUST contain two **Role** definitions, the Principal and the Dependent role elements.

The following is the XML schema definition of the **ReferentialConstraint** element.

```
<xs:complexType name="TConstraint">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:element name="Principal" type="edm:TReferentialConstraintRoleElement" minOccurs="1"
maxOccurs="1" />
    <xs:element name="Dependent" type="edm:TReferentialConstraintRoleElement" minOccurs="1"
maxOccurs="1" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

2.1.9.1 Principal

The **Principal** element specifies the primary (or parent) end of a relationship. The entity that is specified as the **Principal** must exist in order for any Dependent entities to be associated to that parent.

The following is the XML schema definition of the **Principal** element.

```
<xs:complexType name="TReferentialConstraintRoleElement">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:element name="PropertyRef" type="edm:TPropertyRef" minOccurs="1"
maxOccurs="unbounded" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Role" type="edm:TSimpleIdentifier" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

The following rules apply to the **Principal** element:

- The PropertyRef element MUST be used to specify the properties that participate in the **Principal** role of a ReferentialConstraint element.
- Each **PropertyRef Name** attribute MUST be unique within a given **Principal** element.
- Each **PropertyRef** element on the **Principal** MUST correspond to a **PropertyRef** on the **Dependent**. Therefore, the **Principal** and the **Dependent** of the **ReferentialConstraint** MUST contain the same number of **PropertyRef** elements.
- The **Principal** of a **ReferentialConstraint** MUST specify all properties that constitute the EntityKey of the EntityType that forms the **Principal** of the **ReferentialConstraint**.
- The multiplicity value of the **Principal** role MUST be 1 or 0..1.
- The multiplicity value of the **Dependent** role MUST be 0..1 or *.
- The **Type** attribute of each Property that is specified in the **Principal** role MUST be the same as the **Type** of the corresponding **Property** that is specified in the **Dependent** role.

2.1.9.2 Dependent

The **Dependent** element specifies the dependent (or child) end of the relationship. The entity that is specified as the **Dependent** cannot exist unless the Principal entity exists.

The following is the XML schema definition of the **Dependent** element.

```
<xs:complexType name="TReferentialConstraintRoleElement">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:element name="PropertyRef" type="edm:TPropertyRef" minOccurs="1"
maxOccurs="unbounded" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Role" type="edm:TSimpleIdentifier" use="required" />
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

The following rules apply to the **Dependent** role element:

- Each PropertyRef **Name** attribute MUST be unique within a given **Dependent** element.
- The **PropertyRef** element MUST be used to specify the properties that participate in the **Dependent** role of the ReferentialConstraint element.
- Each **PropertyRef** element on the **Principal** MUST correspond to a **PropertyRef** on the **Dependent**. Therefore, the **Principal** and the **Dependent** of the **ReferentialConstraint** MUST contain the same number of **PropertyRef** elements.
- The **Type** attribute of each Property that is specified on the **Principal** role MUST be the same as the **Type** attribute of the corresponding **Property** that is specified on the **Dependent** role.

2.1.10 EntityContainer

The **EntityContainer** element is conceptually similar to a database or data source. This element groups EntitySet and AssociationSet subelements that represent a data source. All Entity Data Model

(EDM) instance-based concepts, such as **EntitySet** and **AssociationSet**, are defined within the scope of an **EntityContainer** element. Users can have one or more instances of an **EntityContainer**. Elements that are defined in an **EntityContainer** can reference one or more schema definitions.

The following is the XML schema definition of the **EntityContainer** element.

```
<xs:element name="EntityContainer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1"
/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="EntitySet">
            <!-- EntitySet schema covered EntitySet section of this document -->
        </xs:element>
        <xs:element name="AssociationSet">
            <!-- AssociationSet schema covered EntitySet section of this document -->
        </xs:element>
      </xs:choice>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"
/>
    </xs:sequence>
    <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>
```

The following rule applies to the **EntityContainer** element:

- The names of the **EntitySet** and **AssociationSet** elements within an **EntityContainer** MUST be unique.

2.1.11 EntitySet

The **EntitySet** element is a named set that contains instances of a specified EntityType. More than one **EntitySet** for a particular **EntityType** can be specified.

The following is the XML schema definition of the **EntitySet** element.

```
<xs:element name="EntitySet">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="DefiningQuery" type="edm:TCommandText" minOccurs="0" maxOccurs="1"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"
/>
    </xs:sequence>
    <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
    <xs:attribute name="EntityType" type="edm:TQualifiedName" use="required" />
    <xs:attribute name="Schema" type="edm:TSimpleIdentifier" use="optional" />
    <xs:attribute name="Table" type="edm:TSimpleIdentifier" use="optional" />
    <xs:attribute ref="gen:Type" use="optional"/>
    <xs:attribute ref="gen:Schema" use="optional"/>
    <xs:attribute ref="gen:Name" use="optional"/>
    <xs:anyAttribute processContents="lax" namespace="##other" />
```

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

The following rules apply to the **EntitySet** element:

- The **EntityType** of an **EntitySet** MUST be in scope of the Schema that declares the EntityContainer in which this **EntitySet** resides.
- Multiple **EntitySet** elements can be specified for a given **EntityType**.

2.1.12 DefiningQuery

The **DefiningQuery** element specifies an EntitySet element based on a database command that can be executed directly on the database. A **DefiningQuery** is analogous to a database view, but the view is specified in the SSDL instead of being physically specified on the database.

EntitySet elements that are backed by a **DefiningQuery** that is specified in SSDL can be mapped to **EntitySet** elements in the conceptual model by using the **EntitySetMapping** element that is specified in MSL. For more details about the **EntitySetMapping** element, see section 2.1.3 of [MS-MSL].

The following is the XML schema definition of the **DefiningQuery** element.

```
<xs:element name="DefiningQuery" type="edm:TCommandText" minOccurs="0" maxOccurs="1"/>
```

The following rules apply to the **DefiningQuery** element:

- The **DefiningQuery** element MUST NOT have any subelements or attributes.
- The **DefiningQuery** body MUST be a valid string, as specified in [XML1.0]. White spaces are allowed and MUST be preserved.

2.1.13 AssociationSet

The **AssociationSet** element contains relationship instances of the specified Association type. The **Association** specifies the EntityType elements of the two end points. The **AssociationSet** element specifies the EntitySet that corresponds to these **EntityType** elements directly or to derived **EntityType** elements. The association instances that are contained in the **AssociationSet** relate entity instances that belong to these **EntityType** elements.

The following is the XML schema definition of the **AssociationSet** element.

```
<xs:element name="AssociationSet">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="End" minOccurs="0" maxOccurs="2">
        <!– End schema covered in the Association Set End section of this document -->
      </xs:element>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"
/>
    </xs:sequence>
    <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
    <xs:attribute name="Association" type="edm:TQualifiedName" use="required" />
```

```
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>
```

The following rules apply to the **AssociationSet** element:

- The **AssociationSet** element MUST have the **Association** attribute specified. The value of the **Association** attribute MUST match the corresponding **Association** type name.
- The **Association** attribute of an **AssociationSet** element MUST be in scope of the Schema that declares the EntityContainer in which this **AssociationSet** resides.
- An **AssociationSet** element MUST have exactly two AssociationEnd elements specified.

2.1.13.1 End

The **End** element specifies the two sides of an AssociationSet element. This association is specified between the two entity sets that are declared in an EntitySet attribute.

The following is the XML schema definition of the **End** element.

```
<xs:element name="End" minOccurs="0" maxOccurs="2">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="edm:GEmptyElementExtensibility" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="Role" type="edm:TSimpleIdentifier" use="optional" />
    <xs:attribute name="EntitySet" type="edm:TSimpleIdentifier" use="required" />
  </xs:complexType>
</xs:element>
```

The following rules apply to **End** elements inside an **AssociationSet** element:

- The **EntitySet** attribute MUST be the **Name** of an **EntitySet** that is specified inside the EntityContainer that also defines the **AssociationSet** that defines the **End** element.
- The **End** element's **Role** attribute MUST map to a **Role** that is declared on one of the **End** elements of the Association that is referenced by the **End** element's declaring **AssociationSet**.
- Each **End** element that is declared by an **AssociationSet** MUST refer to a different **Role**.
- The **EntitySet** attribute for a particular **End** element MUST contain either the same EntityType as the related **End EntityType** on the **Association** element or an **EntityType** that is derived from that **EntityType**.

2.1.14 Documentation

The **Documentation** element is used to provide documentation of comments about the contents of an SSDL file. Elements that support the use of the **Documentation** element explicitly define this support in the XML schema fragments for those elements.

The following is the XML schema definition of the **Documentation** element.

```
<xs:complexType name="TDocumentation">
  <xs:annotation>
```

```
    <xs:documentation>The Documentation element is used to provide documentation of comments
on the contents of the XML file. It is valid under Schema, Type, Index and Relationship
elements.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Summary" type="edm:TText" minOccurs="0" maxOccurs="1" />
    <xs:element name="LongDescription" type="edm:TText" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
  <xs:anyAttribute processContents="lax" namespace="##other" />
</xs:complexType>
```

2.1.15 AnnotationElement

The **AnnotationElement** element is a custom XML element that is applied to an SSDL element. The **AnnotationElement** element and its subelements can belong to any XML namespace that is not in the list of reserved XML namespaces for SSDL. Elements that support the use of the **AnnotationElement** element explicitly define this support in the XML schema fragments for those elements.

The following is the XML schema definition of the **AnnotationElement** element.

```
<xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
```

The following rules apply to the **AnnotationElement** element:

- The namespace used in annotations MUST be declared, or the namespace declaration MUST be declared as part of the annotation.
- Annotations MUST follow all other subelements. For example, when annotating an EntityType element, the **AnnotationElement** element MUST follow all EntityKey and Property elements.
- More than one named annotation can be specified per SSDL element.
- For a given SSDL element, **AnnotationElement** element names can be non-unique, as long as the combination of namespace plus element name is unique for a particular element.
- **AnnotationElement** is an XML element. It MUST contain a valid XML structure.

2.1.16 Function

The **Function** element is used to specify or declare a store function or stored procedure. These functions are specified as subelements of the Schema element.

The following is the XML schema definition of the **Function** element.

```
<xs:complexType name="TFunction">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:element name="CommandText" type="edm:TCommandText" minOccurs="0" maxOccurs="1" />
    <xs:element name="Parameter" type="edm:TParameter" minOccurs="0" maxOccurs="unbounded" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
  <xs:attribute name="ReturnType" type="edm:TFunctionType" use="optional" />
```

```
    <xs:attribute name="Aggregate" type="xs:boolean" use="optional" />
    <xs:attribute name="BuiltIn" type="xs:boolean" use="optional" />
    <xs:attribute name="StoreFunctionName" type="xs:string" use="optional" />
    <xs:attribute name="NiladicFunction" type="xs:boolean" use="optional" />
    <xs:attribute name="IsComposable" type="xs:boolean" use="optional" default="true" />
    <xs:attribute name="ParameterTypeSemantics" type="edm:TParameterTypeSemantics"
  use="optional" default="AllowImplicitConversion" />
    <xs:attribute name="Schema" type="edm:TSimpleIdentifier" use="optional" />
    <xs:attribute ref="gen:Schema" use="optional"/>
    <xs:attribute ref="gen:Name" use="optional"/>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
  <xs:simpleType name="TParameterTypeSemantics">
    <xs:restriction base="xs:token">
      <xs:enumeration value="ExactMatchOnly" />
      <xs:enumeration value="AllowImplicitPromotion" />
      <xs:enumeration value="AllowImplicitConversion" />
    </xs:restriction>
  </xs:simpleType>
```

The following rules apply to the **Function** element:

- The **Name** attribute represents the name of the current **Function** as it relates to the store or database.
- The **ReturnType** attribute MUST be a store type or a collection of store types.
- The **Function** element can specify the **Aggregate** attribute to indicate whether the function returns an aggregate value. If the **Aggregate** attribute is unspecified, its value is false.
- The **Function** element can specify a **BuiltIn** attribute to indicate whether the function is a built-in function offered by the store. If the **BuiltIn** attribute is unspecified, its value is false.
- The **Function** element can specify a **NiladicFunction** attribute to indicate whether the function is a niladic function. Niladic functions do not accept parameters. If the **NiladicFunction** attribute is unspecified, its value is false.
- A **Function** element that has an **IsComposable** attribute that has a value of true MUST specify a **ReturnType** attribute.
- A **Function** element that has an **IsComposable** attribute that has a value of false MUST NOT specify the **ReturnType**, **Aggregate**, **NiladicFunction**, or **BuiltIn** attributes.
- A **Function** element that has an **Aggregate** attribute that has a value of true MUST have exactly one parameter. The type of this parameter MUST be a collection.
- A **Function** element that has a **CommandText** attribute MUST NOT specify a value of true for the **IsComposable** attribute and MUST NOT specify the **StoreFunctionName** attribute.
- The **Function** element can specify a **Schema** attribute to specify the schema on the store where the function resides. If the **Schema** attribute is unspecified, its value is "default".
- If the **ParameterTypeSemantics** attribute is unspecified, its value is "AllowImplicitConversion".
- **Function** elements are declared as global items inside the **Schema** element.

2.1.16.1 Parameter

The **Parameter** element specifies the parameters of a stored procedure or function in the database. The **Parameter** element allows the schema to specify whether the parameter is an Input, Output, or Input/Output parameter. The **Parameter** element in SSDL is a subelement of the Function element.

The following is the XML schema definition of the **Parameter** element.

```
<xs:complexType name="TParameter">
  <xs:sequence>
    <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0" maxOccurs="1" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="xs:string" use="required" />
  <xs:attribute name="Type" type="edm:TFunctionType" use="required" />
  <xs:attribute name="Mode" type="edm:TParameterMode" use="optional" />
  <xs:attribute name="MaxLength" type="edm:TMaxLengthFacet" use="optional" />
  <xs:attribute name="Precision" type="edm:TPrecisionFacet" use="optional" />
  <xs:attribute name="Scale" type="edm:TScaleFacet" use="optional" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:simpleType name="TFunctionType">
  <xs:union memberTypes="edm:TQualifiedName ">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:pattern value="Collection\([^ \t]{1,}(\.[^ \t]{1,}){0,}\)" />
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
<xs:simpleType name="TParameterMode">
  <xs:restriction base="xs:token">
    <xs:enumeration value="In" />
    <xs:enumeration value="Out" />
    <xs:enumeration value="InOut" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TMaxLengthFacet">
  <xs:union memberTypes="edm:TMax xs:nonNegativeInteger"/>
</xs:simpleType>
<xs:simpleType name="TPrecisionFacet">
  <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>
<xs:simpleType name="TScaleFacet">
  <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>
```

The following rules apply to the **Parameter** element:

- The **Name** attribute is of type SimpleIdentifier and represents the name of the current **Parameter**.
- The **Type** attribute MUST be a store type or a collection of store types.
- If the **Mode** attribute is unspecified, its value is "In".
- The following facets are optional and can be specified on a **Parameter** element:

- MaxLength
- Precision
- Scale

- The set of facets that are applicable to the **Parameter** element depends on the base EDM type of the parameter, as specified in the provider manifest. For more details, see section 2.2 of [MC-CSDL].

2.1.16.2 CommandText

The **CommandText** element specifies a database command that can be used to specify a function. This database command is an SQL statement that is meaningful and executable on the database. The **CommandText** element in SSDL is a subelement of the Function element.

The following is the XML schema definition of the **CommandText** element.

```
<xs:simpleType name="TCommandText">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="preserve"/>
  </xs:restriction>
</xs:simpleType>
```

2.2 Attributes

2.2.1 Action

The **Action** attribute specifies the action to be taken when OnDelete behavior is triggered.

The following is the XML schema definition of the **Action** attribute.

```
<xs:simpleType name="TAction">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Cascade" />
    <xs:enumeration value="Restrict" />
    <xs:enumeration value="None" />
  </xs:restriction>
</xs:simpleType>
```

The **Cascade** action implies that the operation to delete an entity should delete the relationship instance and then apply the action on the entity instance at the other end of the relationship. For example, when a Customer is deleted, all Orders that belong to the Customer are deleted.

2.2.2 Multiplicity

The **Multiplicity** attribute of a relationship specifies the cardinality or number of instances of an EntityType that can be associated with the instances of another **EntityType**. The possible types of multiplicity are as follows:

- One-to-many
- Zero-or-one to one

- Zero-or-one to many

The following is the XML schema definition of the **Multiplicity** attribute.

```
<xs:simpleType name="TMultiplicity">
  <xs:restriction base="xs:token">
    <xs:enumeration value="0..1" />
    <xs:enumeration value="1" />
    <xs:enumeration value="*" />
  </xs:restriction>
</xs:simpleType>
```

2.2.3 QualifiedName

The **QualifiedName** attribute is a string-based representation of the name of an element or attribute.

The following is the XML schema definition of the **QualifiedName** attribute.

```
<xs:simpleType name="TQualifiedName">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

2.2.4 SimpleIdentifier

The **SimpleIdentifier** attribute specifies an identifier that conforms to the rules for the **String** data type as specified in [XMLSCHEMA2].

The following is the XML schema definition of the **SimpleIdentifier** attribute.

```
<xs:simpleType name="TSimpleIdentifier">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

2.2.5 AnnotationAttribute

An **AnnotationAttribute** attribute is a custom XML attribute that is applied to an SSDL element. The attribute MAY belong to any XML namespace (as specified in [XML Namespaces1.0]) that is not on the list of reserved XML namespaces for SSDL. Elements that support the use of the **AnnotationAttribute** element explicitly define this support in the XML schema fragments for those elements.

The following is the XML schema definition of the **AnnotationAttribute** element.

```
<xs:anyAttribute namespace="##other" processContents="lax" />
```

2.2.6 UndottedIdentifier

The **UndottedIdentifier** attribute is a string-based value. It is an XML string type that does not have a period (.) character.

The following is the XML schema definition of the **UndottedIdentifier** attribute.

```
<xs:simpleType name="TUndottedIdentifier">
  <xs:restriction base="xs:string">
    <!-- no periods -->
    <xs:pattern value="[^.]{1,}" />
  </xs:restriction>
</xs:simpleType>
```

3 Structure Examples

The following is an example of a storage schema model that is specified by using SSDL. This example specifies the schema for SSDL version 2.0.

```
<Schema Namespace="Model1" Alias="Self" Provider="System.Data.SqlClient"
ProviderManifestToken="2008"
xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
xmlns="http://schemas.microsoft.com/ado/2009/02/edm/ssdl">
        <EntityContainer Name="Model1Container">
          <EntitySet Name="Customers" EntityType="Model1.Customers" store:Type="Tables"
Schema="dbo" />
          <EntitySet Name="Orders" EntityType="Model1.Orders" store:Type="Tables"
Schema="dbo" />
          <AssociationSet Name="FK_Orders_Customers"
Association="Model1.FK_Orders_Customers">
            <End Role="Customers" EntitySet="Customers" />
            <End Role="Orders" EntitySet="Orders" />
          </AssociationSet>
        </EntityContainer>
        <EntityType Name="Customers">
          <Key>
            <PropertyRef Name="CustomerID" />
          </Key>
          <Property Name="CustomerID" Type="nchar" Nullable="false" MaxLength="5" />
          <Property Name="CompanyName" Type="nvarchar" Nullable="false" MaxLength="40" />
          <Property Name="ContactName" Type="nvarchar" MaxLength="30" />
          <Property Name="ContactTitle" Type="nvarchar" MaxLength="30" />
          <Property Name="Address" Type="nvarchar" MaxLength="60" />
          <Property Name="City" Type="nvarchar" MaxLength="15" />
          <Property Name="Country" Type="nvarchar" MaxLength="15" />
          <Property Name="Phone" Type="nvarchar" MaxLength="24" />
        </EntityType>
        <EntityType Name="Orders">
          <Key>
            <PropertyRef Name="OrderID" />
          </Key>
          <Property Name="OrderID" Type="int" Nullable="false" />
          <Property Name="CustomerID" Type="nchar" MaxLength="5" />
          <Property Name="OrderDate" Type="datetime" />
          <Property Name="ShipAddress" Type="nvarchar" MaxLength="60" />
          <Property Name="ShipCity" Type="nvarchar" MaxLength="15" />
          <Property Name="ShipPostalCode" Type="nvarchar" MaxLength="10" />
          <Property Name="ShipCountry" Type="nvarchar" MaxLength="15" />
        </EntityType>
        <Association Name="FK_Orders_Customers">
          <End Role="Customers" Type="Model1.Customers" Multiplicity="0..1" />
          <End Role="Orders" Type="Model1.Orders" Multiplicity="*" />
          <ReferentialConstraint>
            <Principal Role="Customers">
              <PropertyRef Name="CustomerID" />
            </Principal>
            <Dependent Role="Orders">
              <PropertyRef Name="CustomerID" />
            </Dependent>
          </ReferentialConstraint>
        </Association>
```

```
</Schema>
```

3.1 Schema Example

The following is an example of the Schema element.

```
<Schema Namespace="ExampleModel.Store"
        Alias="Self"
        Provider="System.Data.SqlClient"
        ProviderManifestToken="2008"
        xmlns="http://schemas.microsoft.com/ado/2009/02/edm/ssdl">
```

3.2 EntityType Example

The following is an example of the EntityType element.

```
<EntityType Name="Customers">
    <Key>
      <PropertyRef Name="CustomerID" />
    </Key>
    <Property Name="CustomerID" Type="nchar" Nullable="false" MaxLength="5" />
    <Property Name="CompanyName" Type="nvarchar" Nullable="false" MaxLength="40" />
    <Property Name="ContactName" Type="nvarchar" MaxLength="30" />
    <Property Name="ContactTitle" Type="nvarchar" MaxLength="30" />
    <Property Name="Address" Type="nvarchar" MaxLength="60" />
    <Property Name="City" Type="nvarchar" MaxLength="15" />
    <Property Name="Region" Type="nvarchar" MaxLength="15" />
    <Property Name="PostalCode" Type="nvarchar" MaxLength="10" />
    <Property Name="Country" Type="nvarchar" MaxLength="15" />
    <Property Name="Phone" Type="nvarchar" MaxLength="24" />
    <Property Name="Fax" Type="nvarchar" MaxLength="24" />
</EntityType>
```

3.3 Property Example

The following is an example of the Property element.

```
<Property Name="CompanyName" Type="nvarchar" Nullable="false" MaxLength="40" />
```

3.4 EntityKey Example

The following is an example of the EntityKey element.

```
<Key>
  <PropertyRef Name="CustomerID" />
</Key>
```

3.5 PropertyRef Example

The following is an example of the PropertyRef element.

```
<PropertyRef Name="CustomerId" />
```

3.6 Association Example

The following is an example of the Association element.

```
<Association Name="CustomerOrder">
  <End Type="Model1.Customer" Role="Customer" Multiplicity="1" />
  <End Type="Model1.Order" Role="Order" Multiplicity="*" />
  <ReferentialConstraint>
    <Principal Role="Customer">
      <PropertyRef Name="CustomerID" />
    </Principal>
    <Dependent Role="Order">
      <PropertyRef Name="OrderID" />
    </Dependent>
  </ReferentialConstraint>
</Association>
```

3.7 AssociationEnd Example

The following is an example of the AssociationEnd element.

```
<End Type="Model1.Customer" Role="Customer" Multiplicity="1" />
```

3.8 OnDelete Example

The following is an example of the OnDelete element.

```
<Association Name="ProductCategory">
  <End Role="Product" Type="Self.Product" Multiplicity="*" />
  <End Role="Category" Type="Self.Category" Multiplicity="0..1">
    <OnDelete Action="Cascade" />
  </End>
    <ReferentialConstraint>
      <Principal Role="Category">
        <PropertyRef Name="CategoryID" />
      </Principal>
      <Dependent Role="Product">
        <PropertyRef Name="ProductID" />
      </Dependent>
    </ReferentialConstraint>
  </Association>
```

3.9 ReferentialConstraint Example

The following is an example of the ReferentialConstraint element.

```
<Association Name="FK_Employee_Employee_ManagerID">
  <End Role="Employee" Type="Adventureworks.Store.Employee" Multiplicity="1" />
  <End Role="Manager" Type="Adventureworks.Store.Manager" Multiplicity="0..1" />
  <ReferentialConstraint>
```

```
    <Principal Role="Employee">
      <PropertyRef Name="EmployeeID" />
    </Principal>
    <Dependent Role="Manager">
      <PropertyRef Name="ManagerID" />
    </Dependent>
  </ReferentialConstraint>
</Association>
```

3.10 Principal Example

The following is an example of the usage of the Principal role element as part of a ReferentialConstraint definition that specifies Employee as the **Principal** role.

```
    <Principal Role="Employee">
      <PropertyRef Name="EmployeeID" />
    </Principal>
```

3.11 Dependent Example

The following is an example of using the Dependent role element to specify a ReferentialConstraint.

```
    <Dependent Role="Manager">
      <PropertyRef Name="ManagerID" />
    </Dependent>
```

3.12 EntityContainer Example

The following is an example of the EntityContainer element.

```
<EntityContainer Name="Model1Container" >
  <EntitySet Name="CustomerSet" EntityType="Model1.Customer" />
  <EntitySet Name="OrderSet" EntityType="Model1.Order" />
  <AssociationSet Name="CustomerOrder" Association="Model1.CustomerOrder">
    <End Role="Customer" EntitySet="CustomerSet" />
    <End Role="Order" EntitySet="OrderSet" />
  </AssociationSet>
</EntityContainer>
```

3.13 EntitySet Example

The following is an example of the EntitySet element.

```
  <EntitySet Name="CustomerSet" EntityType="NorthwindModel.Store.Customer" />
```

3.14 DefiningQuery Example

The following is an example of the DefiningQuery element.

```
<EntitySet Name="Tables" EntityType="Self.STable">
```

```
        <DefiningQuery>
          SELECT  TABLE_CATALOG,
                  'test' as TABLE_SCHEMA,
                  TABLE_NAME
          FROM    INFORMATION_SCHEMA.TABLES
        </DefiningQuery>
  </EntitySet>
```

3.15 AssociationSet Example

The following is an example of the AssociationEnd element.

```
    <AssociationSet Name="CustomerOrder" Association="Model1.CustomerOrder">
      <End Role="Customer" EntitySet="CustomerSet" />
      <End Role="Order" EntitySet="OrderSet" />
    </AssociationSet>
```

3.16 End Example

The following is an example of the End element.

```
      <End Role="Customer" EntitySet="CustomerSet" />
```

3.17 Documentation Example

The following is an example of the Documentation element on the EntityContainer element.

```
<EntityContainer Name="TwoThreeContainer">
   <Documentation>
      <Summary>Summary: entity container for storing Northwind instances</Summary>
      <LongDescription>LongDescription: This entity container is for storing Northwind
instances</LongDescription>
   </Documentation>
  <EntitySet Name="Products" EntityType="Self.Product" />
</EntityContainer>
```

The following is an example of the **Documentation** element on the EntitySet element.

```
<EntitySet Name="Products" EntityType="Self.Product">
   <Documentation>
       <Summary>EntitySet Products is for storing instances of EntityType Product</Summary>
       <LongDescription>This EntitySet having name Products is for storing instances of
EntityType Product</LongDescription>
   </Documentation>
</EntitySet>
```

The following is an example of the **Documentation** element on the AssociationSet element and the End role.

```
<AssociationSet Name="CategoryProducts" Association="Self.CategoryProduct">
     <Documentation>
```

```
        <Summary>AssociationSet CategoryProducts is for storing instances of Association
CategoryProduct</Summary>
        <LongDescription>This AssociationSet having name=CategoryProducts is for storing
instances of Association CategoryProduct</LongDescription>
    </Documentation>
  <End Role="Category" EntitySet="Categories">
    <Documentation>
        <Summary>This end of the relationship-instance describes the Category role for
AssociationSet CategoryProducts</Summary>
    </Documentation>
  </End>
  <End Role="Product" EntitySet="Products">
    <Documentation>
        <LongDescription>This end of the relationship-instance describes the Product role
for AssociationSet CategoryProducts</LongDescription>
    </Documentation>
  </End>
</AssociationSet>
```

The following is an example of the **Documentation** element on the EntityType and Property elements.

```
<EntityType Name="Product">
  <Documentation>
    <Summary>Summary: EntityType named Product describes the content model for
Product</Summary>
    <LongDescription>LongDescription: The EntityType named Product describes the content
model for Product</LongDescription>
  </Documentation>
  <Key>
    <PropertyRef Name="ProductID" />
  </Key>
  <Property Name="ProductID" Type="Int32" Nullable="false">
      <Documentation>
          <Summary>Summary: This is the key property of EntityType Product</Summary>
          <LongDescription>LongDescription: This is the key property of EntityType
Product</LongDescription>
      </Documentation>
  </Property>
  <Property Name="ProductName" Type="String">
      <Documentation>
          <Summary>Summary: This property describes the name of the Product</Summary>
      </Documentation>
  </Property>
  <Property Name="QuantityPerUnit" Type="String">
      <Documentation>
        <LongDescription>LongDescription: This property describes the quantity per unit
corresponding to a product</LongDescription>
      </Documentation>
  </Property>
  <Property Name="PriceInfo" Nullable="false" Type="Self.PriceInfo" />
  <Property Name="StockInfo" Nullable="false" Type="Self.StockInfo" />
</EntityType>
```

The following is an example of the **Documentation** element on the Association element.

```
<Association Name="CategoryProduct">
```

```
   <Documentation>
      <Summary>Association CategoryProduct describes the participating end of the
CategoryProduct relationship</Summary>
   </Documentation>
  <End Role="Category" Type="Self.Category" Multiplicity="1">
     <Documentation>
          <Summary>This end of the relationship-instance describes the Category role for
Association CategoryProduct</Summary>
     </Documentation>
  </End>
  <End Role="Product" Type="Self.Product" Multiplicity="*">
     <Documentation>
         <LongDescription>This end of the relationship-instance describes the Product role
for Association CategoryProduct</LongDescription>
     </Documentation>
  </End>
    <ReferentialConstraint>
      <Documentation>ReferentialConstraint representing the Foreign Key relationship in the
store</Documentation>
       <Principal Role="Category">
           <PropertyRef Name="CategoryID" />
       </Principal>
       <Dependent Role="Product">
           <PropertyRef Name="ProductID" />
       </Dependent>
    </ReferentialConstraint>
</Association>
```

3.18 AnnotationElement Example

The following is an example of the AnnotationElement element.

```
<EntityType Name="Content">
  <Key>
    <PropertyRef Name="ID" />
  </Key>
  <Property Name="ID" Type="Guid" Nullable="false" />
  <Property Name="HTML" Type="String" Nullable="false" MaxLength="Max" Unicode="true"
     FixedLength="false" />
  <CLR:Attributes>
    <CLR:Attribute TypeName="System.Runtime.Serialization.DataContract"/>
    <CLR:Attribute TypeName="MyNamespace.MyAttribute"/>
  </CLR:Attributes>
  <RS:Security>
    <RS:ACE Principal="S-0-123-1321" Rights="+R+W"/>
    <RS:ACE Principal="S-0-123-2321" Rights="-R-W"/>
  </RS:Security>
</EntityType>
```

3.19 Function Example

The following is an example of the Function element.

```
        <Function Name="UpdateOrderQuantity"
          Aggregate="false"
          BuiltIn="false"
```

```
    NiladicFunction="false"
    IsComposable="false"
    ParameterTypeSemantics="AllowImplicitConversion"
    Schema="dbo">
    <Parameter Name="orderId" Type="int" Mode="In" />
    <Parameter Name="newQuantity" Type="int" Mode="In" />
  </Function>
```

3.20 Parameter Example

The following is an example of the Parameter element.

```
    <Parameter Name="orderId" Type="int" Mode="In" />
    <Parameter Name="newQuantity" Type="int" Mode="In" />
```

4 Security Considerations

None.

5 Appendix A: Full XML Schema Definition for Store Schema Definition Language

The following is the complete XML schema definition for store schema definition language (SSDL).

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:edm="http://schemas.microsoft.com/ado/2009/02/edm/ssdl"

xmlns:gen="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
           targetNamespace="http://schemas.microsoft.com/ado/2009/02/edm/ssdl">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            Common Data Model Schema Definition Language.
            Copyright (c) Microsoft Corp. All rights reserved.
        </xs:documentation>
    </xs:annotation>
  <xs:import
namespace="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
schemaLocation="System.Data.Resources.EntityStoreSchemaGenerator.xsd" />
  <xs:element name="Schema" type="edm:TSchema"/>
    <xs:complexType name="TSchema">
        <xs:sequence>
            <xs:group ref="edm:GSchemaBodyElements" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Namespace" type="edm:TQualifiedName" use="required" />
        <xs:attribute name="Alias" type="edm:TSimpleIdentifier" use="optional" />
        <xs:attribute name="Provider" type="edm:TSimpleIdentifier" use="required" />
        <xs:attribute name="ProviderManifestToken" type="edm:TSimpleIdentifier"
use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:group name="GSchemaBodyElements">
        <xs:choice>
            <xs:element name="Association" type="edm:TAssociation" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="EntityType" type="edm:TEntityType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element ref="edm:EntityContainer" minOccurs="1" maxOccurs="1"/>
            <xs:element name="Function" type="edm:TFunction" minOccurs="0"
maxOccurs="unbounded" />
        </xs:choice>
    </xs:group>
    <xs:simpleType name="TMax">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Max"/>
        </xs:restriction>
    </xs:simpleType>
    <!-- Facets for Primitive types -->
    <xs:simpleType name="TMaxLengthFacet">
        <xs:union memberTypes="edm:TMax xs:nonNegativeInteger"/>
    </xs:simpleType>
    <xs:simpleType name="TIsFixedLengthFacet">
        <xs:restriction base="xs:boolean"/>
    </xs:simpleType>
```

```
    <xs:simpleType name="TKindFacet">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Utc"/>
            <xs:enumeration value="Local"/>
            <xs:enumeration value="Unspecified"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="TPrecisionFacet">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <xs:simpleType name="TScaleFacet">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <xs:simpleType name="TIsUnicodeFacet">
        <xs:restriction base="xs:boolean"/>
    </xs:simpleType>
    <xs:simpleType name="TCollationFacet">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <!--
        types of the top level elements
    -->
   <xs:complexType name="TDocumentation">
    <xs:annotation>
      <xs:documentation>The Documentation element is used to provide documentation of
comments on the contents of the XML file. It is valid under Schema, Type, Index
and Relationship elements.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Summary" type="edm:TText" minOccurs="0" maxOccurs="1" />
      <xs:element name="LongDescription" type="edm:TText" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
    <xs:anyAttribute processContents="lax" namespace="##other" />
  </xs:complexType>
  <xs:complexType name="TText" mixed="true">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"
/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax" namespace="##other" />
  </xs:complexType>
    <xs:complexType name="TAssociation">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:element name="End" type="edm:TAssociationEnd" minOccurs="2" maxOccurs="2"/>
            <xs:element name="ReferentialConstraint" type="edm:TConstraint" minOccurs="0"
maxOccurs="1" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
        <!--<xs:attribute name="Identifying" type="xs:boolean" use="optional" default="false"
/>-->
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:complexType name="TConstraint">
        <xs:sequence>
```

```
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:element name="Principal" type="edm:TReferentialConstraintRoleElement"
minOccurs="1" maxOccurs="1" />
            <xs:element name="Dependent" type="edm:TReferentialConstraintRoleElement"
minOccurs="1" maxOccurs="1" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="TReferentialConstraintRoleElement">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:element name="PropertyRef" type="edm:TPropertyRef" minOccurs="1"
maxOccurs="unbounded" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Role" type="edm:TSimpleIdentifier" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="TEntityType">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:element name="Key" type="edm:TEntityKeyElement" minOccurs="0" maxOccurs="1"/>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="Property" type="edm:TEntityProperty" minOccurs="0"
maxOccurs="unbounded" />
            </xs:choice>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name ="TEntityKeyElement">
        <xs:sequence>
            <xs:element name="PropertyRef" type="edm:TPropertyRef" minOccurs="1"
maxOccurs="unbounded" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TPropertyRef">
        <xs:sequence>
            <xs:group ref="edm:GEmptyElementExtensibility" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="Name" type="edm:TSimpleIdentifier" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:group name="GEmptyElementExtensibility">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
```

```
    </xs:group>
    <!--
        base types
    -->
    <xs:complexType name="TAssociationEnd">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:group ref="edm:TOperations" minOccurs="0" maxOccurs="unbounded" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Type" type="edm:TQualifiedName" use="required" />
        <xs:attribute name="Role" type="edm:TSimpleIdentifier" use="optional" />
        <xs:attribute name="Multiplicity" type="edm:TMultiplicity" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:group name="TOperations">
        <xs:choice>
            <xs:element name="OnDelete" type="edm:TOnAction" maxOccurs="1" minOccurs="0" />
        </xs:choice>
    </xs:group>
    <xs:complexType name="TOnAction">
        <xs:sequence>
            <xs:group ref="edm:GEmptyElementExtensibility" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="Action" type="edm:TAction" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:complexType name="TEntityProperty">
        <xs:sequence>
            <xs:group ref="edm:GEmptyElementExtensibility" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attributeGroup ref="edm:TCommonPropertyAttributes"/>
        <xs:attribute name="StoreGeneratedPattern" type="edm:TStoreGeneratedPattern"
use="optional" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:attributeGroup name="TCommonPropertyAttributes">
        <xs:attribute name="Name" type="edm:TSimpleIdentifier" use="required" />
        <xs:attribute name="Type" type="edm:TPropertyType" use="required" />
        <xs:attribute name="Nullable" type="xs:boolean" default="true" use="optional" />
        <xs:attribute name="DefaultValue" type="xs:string" use="optional" />
        <!-- Start Facets -->
        <xs:attribute name="MaxLength" type="edm:TMaxLengthFacet" use="optional" />
        <xs:attribute name="FixedLength" type="edm:TIsFixedLengthFacet" use="optional" />
        <xs:attribute name="Precision" type="edm:TPrecisionFacet" use="optional" />
        <xs:attribute name="Scale" type="edm:TScaleFacet" use="optional" />
        <xs:attribute name="Unicode" type="edm:TIsUnicodeFacet" use="optional" />
        <xs:attribute name="Collation" type="edm:TCollationFacet" use="optional" />
        <!--End Facets -->
    </xs:attributeGroup>
    <xs:element name="EntityContainer">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="EntitySet">
```

```
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Documentation" type="edm:TDocumentation"
minOccurs="0" maxOccurs="1" />
                                <xs:element name="DefiningQuery" type="edm:TCommandText"
minOccurs="0" maxOccurs="1"/>
                                <xs:any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded" />
                            </xs:sequence>
                          <xs:attribute name="Name" type="edm:TUndottedIdentifier"
use="required" />
                          <xs:attribute name="EntityType" type="edm:TQualifiedName"
use="required" />
                          <xs:attribute name="Schema" type="edm:TSimpleIdentifier"
use="optional" />
                          <xs:attribute name="Table" type="edm:TSimpleIdentifier"
use="optional" />
                          <xs:attribute ref="gen:Type" use="optional"/>
                          <xs:attribute ref="gen:Schema" use="optional"/>
                          <xs:attribute ref="gen:Name" use="optional"/>
                          <xs:anyAttribute processContents="lax" namespace="##other" />
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="AssociationSet">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Documentation" type="edm:TDocumentation"
minOccurs="0" maxOccurs="1" />
                                <xs:element name="End" minOccurs="0" maxOccurs="2">
                                    <!-- 1. The number of Ends has to match with ones defined
in AssociationType
                                         2. Value for attribute Name should match the defined
ones and EntitySet should be of the
                                            defined Entity Type in AssociationType
                                    -->
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:group ref="edm:GEmptyElementExtensibility"
minOccurs="0" maxOccurs="1"/>
                                        </xs:sequence>
                                        <xs:attribute name="Role"
type="edm:TSimpleIdentifier" use="optional" />
                                        <xs:attribute name="EntitySet"
type="edm:TSimpleIdentifier" use="required" />
                                    </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded" />
                            </xs:sequence>
                            <xs:attribute name="Name" type="edm:TUndottedIdentifier"
use="required" />
                            <xs:attribute name="Association" type="edm:TQualifiedName"
use="required" />
                            <xs:anyAttribute namespace="##other" processContents="lax" />
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
              <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
            </xs:sequence>
```

```
            <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
            <xs:anyAttribute namespace="##other" processContents="lax" />
        </xs:complexType>
    </xs:element>
    <xs:complexType name="TFunction">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:element name="CommandText" type="edm:TCommandText" minOccurs="0"
maxOccurs="1" />
            <xs:element name="Parameter" type="edm:TParameter" minOccurs="0"
maxOccurs="unbounded" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Name" type="edm:TUndottedIdentifier" use="required" />
        <xs:attribute name="ReturnType" type="edm:TFunctionType" use="optional" />
        <xs:attribute name="Aggregate" type="xs:boolean" use="optional" />
        <xs:attribute name="BuiltIn" type="xs:boolean" use="optional" />
        <xs:attribute name="StoreFunctionName" type="xs:string" use="optional" />
        <xs:attribute name="NiladicFunction" type="xs:boolean" use="optional" />
        <xs:attribute name="IsComposable" type="xs:boolean" use="optional" default="true" />
        <xs:attribute name="ParameterTypeSemantics" type="edm:TParameterTypeSemantics"
use="optional" default="AllowImplicitConversion" />
        <xs:attribute name="Schema" type="edm:TSimpleIdentifier" use="optional" />
        <xs:attribute ref="gen:Schema" use="optional"/>
        <xs:attribute ref="gen:Name" use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:complexType name="TParameter">
        <xs:sequence>
            <xs:element name="Documentation" type="edm:TDocumentation" minOccurs="0"
maxOccurs="1" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" use="required" />
        <xs:attribute name="Type" type="edm:TFunctionType" use="required" />
        <xs:attribute name="Mode" type="edm:TParameterMode" use="optional" />
        <!-- Start Facets -->
        <xs:attribute name="MaxLength" type="edm:TMaxLengthFacet" use="optional" />
        <xs:attribute name="Precision" type="edm:TPrecisionFacet" use="optional" />
        <xs:attribute name="Scale" type="edm:TScaleFacet" use="optional" />
        <!--End Facets -->
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <!--
    general (more or less) purpose simple types
    -->
  <xs:simpleType name="TCommandText">
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="TQualifiedName">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
  <xs:simpleType name="TUndottedIdentifier">
    <xs:restriction base="xs:string">
```

```
      <!-- no periods -->
      <xs:pattern value="[^.]{1,}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="TSimpleIdentifier">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="TPropertyType">
        <xs:union memberTypes="edm:TQualifiedName">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <!-- The below pattern represents the allowed identifiers in ECMA
specification plus the '.' for namespace qualification -->
                    <xs:pattern
value="[\p{L}\p{Nl}][\p{L}\p{Nl}\p{Nd}\p{Mn}\p{Mc}\p{Pc}\p{Cf}]{0,}(\.[\p{L}\p{Nl}][\p{L}\p{N
l}\p{Nd}\p{Mn}\p{Mc}\p{Pc}\p{Cf}]{0,})
{0,}" />
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
    </xs:simpleType>
    <xs:simpleType name="TAction">
        <xs:restriction base="xs:token">
            <xs:enumeration value="Cascade" />
            <xs:enumeration value="Restrict" />
            <xs:enumeration value="None" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="TMultiplicity">
    <xs:restriction base="xs:token">
      <xs:enumeration value="0..1" />
      <xs:enumeration value="1" />
      <xs:enumeration value="*" />
    </xs:restriction>
  </xs:simpleType>
    <xs:simpleType name="TStoreGeneratedPattern">
        <xs:restriction base="xs:token">
            <xs:enumeration value="None" />
            <xs:enumeration value="Identity" />
            <xs:enumeration value="Computed" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="TParameterMode">
        <xs:restriction base="xs:token">
            <xs:enumeration value="In" />
            <xs:enumeration value="Out" />
            <xs:enumeration value="InOut" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="TFunctionType">
        <xs:union memberTypes="edm:TQualifiedName ">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:pattern value="Collection\([^ \t]{1,}(\.[^ \t]{1,}){0,}\)" />
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
    </xs:simpleType>
    <xs:simpleType name="TParameterTypeSemantics">
```

```
        <xs:restriction base="xs:token">
            <xs:enumeration value="ExactMatchOnly" />
            <xs:enumeration value="AllowImplicitPromotion" />
            <xs:enumeration value="AllowImplicitConversion" />
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

6 Appendix B: Product Behavior

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

- Microsoft® .NET Framework 3.5
- Microsoft® .NET Framework 4.0

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

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

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index