Model Functions?

CollectionType/TypeRef?

ReferenceType/RowType?

Expressions

Annotations

dataservice attributes (i.e., to functions)

Model Reference/Annotation Reference

MARK’S TODOS

Link up the ABNF somehow  
Consider consolidating sections into coarser granularity  
Ensure all examples use customers-orders-products motif

Table of Contents

[Appendix A: Formal Common Schema Definition Language (CSDL) 5](#_Toc319851348)

[1 Common Schema Definition Language (CSDL) Namespaces 5](#_Toc319851349)

[1.1 Entity Data Model for Data Services Packaging (EDMX) Namespace 5](#_Toc319851350)

[1.2 Entity Data Model (EDM) Namespace 6](#_Toc319851351)

[1.3 Data Service Metadata Namespace 6](#_Toc319851352)

[2 Common Characteristics of Entity Model Elements 6](#_Toc319851353)

[2.1 Nominal Elements 6](#_Toc319851354)

[2.2 Structured Elements 7](#_Toc319851355)

[2.3 The edm:Documentation Element 7](#_Toc319851356)

[2.4 Custom Annotations 7](#_Toc319851357)

[2.5 Primitive Types 8](#_Toc319851358)

[3 Entity Model Wrapper Constructs 9](#_Toc319851359)

[3.1 The edmx:Edmx Element 9](#_Toc319851360)

[3.1.1 The Version Attribute 9](#_Toc319851361)

[3.2 The edmx:DataServices Element 9](#_Toc319851362)

[3.2.1 The metadata:DataServiceVersion Attribute 9](#_Toc319851363)

[3.3 The edmx:Reference Element 9](#_Toc319851364)

[3.3.1 The edmx:Url Attribute 10](#_Toc319851365)

[3.4 The edmx:AnnotationsReference Element 10](#_Toc319851366)

[3.4.1 The edmx:Url Attribute 11](#_Toc319851367)

[3.5 The edmx:Include Element 11](#_Toc319851368)

[3.5.1 The edmx:TermNamespace Attribute 11](#_Toc319851369)

[3.5.2 The edmx:Qualifier Attribute 11](#_Toc319851370)

[4 Schema Constructs 12](#_Toc319851371)

[4.1 The edm:Schema Element 12](#_Toc319851372)

[4.1.1 The edm:Namespace Attribute 12](#_Toc319851373)

[4.1.2 The edm:Alias Attribute 12](#_Toc319851374)

[4.2 The edm:Using Element 13](#_Toc319851375)

[4.2.1 The edm:Namespace Attribute 13](#_Toc319851376)

[4.2.2 The edm:Alias Attribute 13](#_Toc319851377)

[5 Properties 13](#_Toc319851378)

[5.1 The edm:Property Element 14](#_Toc319851379)

[5.1.1 The edm:Name Attribute 14](#_Toc319851380)

[5.2 The edm:Type Attribute 14](#_Toc319851381)

[5.2.1 The edm:Nullable Attribute 14](#_Toc319851382)

[5.2.2 The edm:MaxLength Attribute 14](#_Toc319851383)

[5.2.3 The edm:FixedLength Attribute 15](#_Toc319851384)

[5.2.4 The edm:Precision Attribute 15](#_Toc319851385)

[5.2.5 The edm:Scale Attribute 15](#_Toc319851386)

[5.2.6 The edm:Unicode Attribute 15](#_Toc319851387)

[5.2.7 The edm:Collation Attribute 15](#_Toc319851388)

[5.2.8 The edm:SRID Attribute 16](#_Toc319851389)

[5.2.9 The edm:Default Attribute 16](#_Toc319851390)

[5.2.10 The edm:ConcurrencyMode Attribute 16](#_Toc319851391)

[6 Entity Type Constructs 16](#_Toc319851392)

[6.1 The edm:EntityType Element 17](#_Toc319851393)

[6.1.1 The edm:Name Attribute 17](#_Toc319851394)

[6.1.2 The edm:BaseType Attribute 18](#_Toc319851395)

[6.1.3 The edm:Abstract Attribute 18](#_Toc319851396)

[6.1.4 The edm:OpenType Attribute 18](#_Toc319851397)

[6.2 The edm:Key Element 18](#_Toc319851398)

[6.3 The edm:PropertyRef Element 19](#_Toc319851399)

[6.4 The edm:NavigationProperty Element 19](#_Toc319851400)

[6.4.1 The edm:Name Attribute 19](#_Toc319851401)

[6.4.2 The edm:Relationship Attribute 19](#_Toc319851402)

[6.4.3 The edm:ToRole Attribute 19](#_Toc319851403)

[6.4.4 The edm:FromRole Attribute 20](#_Toc319851404)

[7 Complex Type Constructs 20](#_Toc319851405)

[7.1 The edm:ComplexType Element 20](#_Toc319851406)

[8 Enumeration Type Constructs 20](#_Toc319851407)

[8.1 The edm:Enum Element 21](#_Toc319851408)

[8.1.1 The edm:Name Attribute 21](#_Toc319851409)

[8.1.2 The edm:UnderlyingType Attribute 21](#_Toc319851410)

[8.1.3 The edm:IsFlags Attribute 21](#_Toc319851411)

[8.2 The edm:Member Element 21](#_Toc319851412)

[8.2.1 The edm:Name Attribute 21](#_Toc319851413)

[8.2.2 The edm:Value Attribute 22](#_Toc319851414)

[9 Association Constructs 22](#_Toc319851415)

[9.1 The edm:Association Element 22](#_Toc319851416)

[9.1.1 The edm:Name Attribute 22](#_Toc319851417)

[9.2 The edm:End Element 23](#_Toc319851418)

[9.2.1 The edm:Type Attribute 23](#_Toc319851419)

[9.2.2 The edm:Role Attribute 23](#_Toc319851420)

[9.2.3 The edm:Multiplicity Attribute 23](#_Toc319851421)

[9.3 The edm:OnDelete Element 23](#_Toc319851422)

[9.4 The edm:ReferentialConstraint Element 23](#_Toc319851423)

[9.5 The edm:Principal Element 24](#_Toc319851424)

[9.6 The edm:Dependent Element 24](#_Toc319851425)

[9.7 The edm:PropertyRef Element 24](#_Toc319851426)

[10 Entity Container Constructs 25](#_Toc319851427)

[10.1 The "edm:EntityContainer" Element 25](#_Toc319851428)

[10.2 The "edm:EntitySet" Element 26](#_Toc319851429)

[10.3 The "edm:AssociationSet" Element 26](#_Toc319851430)

[10.4 The "edm:FunctionImport" Element 26](#_Toc319851431)

[10.4.1 The "edm:Name" Attribute 26](#_Toc319851432)

[10.4.2 The "edm:ReturnType" Attribute 26](#_Toc319851433)

[10.4.3 The "edm:EntitySet" Attribute 26](#_Toc319851434)

[10.4.4 The "edm:EntityPath" Attribute 26](#_Toc319851435)

[10.5 The "edm:ReturnType" Element 26](#_Toc319851436)

[10.5.1 The "edm:Type" Attribute 26](#_Toc319851437)

[10.5.2 The "edm:EntitySet" Attribute 26](#_Toc319851438)

[10.6 The "edm:Parameter" Element 26](#_Toc319851439)

[10.6.1 The "edm:Name" Attribute 26](#_Toc319851440)

[10.6.2 The "edm:Type" Attribute 26](#_Toc319851441)

[10.6.3 The "edm:Mode" Attribute 26](#_Toc319851442)

[10.6.4 The "edm:MaxLength" Attribute 26](#_Toc319851443)

[10.6.5 The "edm:Precision" Attribute 26](#_Toc319851444)

[10.6.6 The "edm:Scale" Attribute 26](#_Toc319851445)

[11 Annotation Constructs 26](#_Toc319851446)

[11.1 The "edm:TypeAnnotation" Element 26](#_Toc319851447)

[11.2 The "edm:PropertyValue" Element 26](#_Toc319851448)

[11.2.1 The "edm:Path" Attribute 26](#_Toc319851449)

[11.2.2 The "edm:String" Attribute 27](#_Toc319851450)

[11.2.3 The "edm:Int" Attribute 27](#_Toc319851451)

[11.2.4 The "edm:Float" Attribute 27](#_Toc319851452)

[11.2.5 The "edm:Decimal" Attribute 27](#_Toc319851453)

[11.2.6 The "edm:DateTime" Attribute 27](#_Toc319851454)

[11.3 The "edm:ValueAnnotation" Element 27](#_Toc319851455)

[11.3.1 The "edm:Path" Attribute 27](#_Toc319851456)

[11.3.2 The "edm:String" Attribute 27](#_Toc319851457)

[11.3.3 The "edm:Int" Attribute 27](#_Toc319851458)

[11.3.4 The "edm:Float" Attribute 27](#_Toc319851459)

[11.3.5 The "edm:Decimal" Attribute 27](#_Toc319851460)

[11.3.6 The "edm:DateTime" Attribute 27](#_Toc319851461)

[11.4 The "edm:ValueTerm" Element 27](#_Toc319851462)

[11.4.1 The "edm:Name" Attribute 27](#_Toc319851463)

[11.4.2 The "edm:Type" Attribute 27](#_Toc319851464)

[12 Expression Constructs 27](#_Toc319851465)

[13 CSDL Examples 27](#_Toc319851466)

[13.1 Title for example 27](#_Toc319851467)

[14 ABNF for CSDL 29](#_Toc319851468)

# Appendix A: Formal Common Schema Definition Language (CSDL)

OData services are described by an Entity Data Model (EDM). Common Schema Definition Language (CSDL) defines an XML-based representation of the Entity Model exposed by an OData service.

# Common Schema Definition Language (CSDL) Namespaces

In addition to the default XML namespace, the elements and attributes used to describe the entity model of an OData service are defined in one of the following namespaces.

## Entity Data Model for Data Services Packaging (EDMX) Namespace

Elements and attributes associated with the top-level wrapper that contains the CSDL used to define the entity model for an OData Service are qualified with the Entity Data Model for Data Services Packaging namespace: http://schemas.microsoft.com/ado/2007/06/edmx.

In this specification the namespace prefix "edmx" is used to represent the Entity Data Model for Data Services Packaging namespace, however the prefix name is not prescriptive.

## Entity Data Model (EDM) Namespace

Elements and attributes that define the entity model exposed by the OData Service are qualified with the Entity Data Model namespace: http://schemas.microsoft.com/ado/2007/06/edm.

In this specification the namespace prefix "edm" is used to represent the Entity Data Model namespace, however the prefix name is not prescriptive.

## Data Service Metadata Namespace

Elements and attributes specific to how the entity model is exposed as an OData Service are qualified with the Data Service Metadata namespace: http://schemas.microsoft.com/ado/2007/08/DataServices/Metadata.

In this specification the namespace prefix "metadata" is used to represent the Data Service Metadata namespace, however the prefix name is not prescriptive.

# Common Characteristics of Entity Model Elements

A typical entity model for an OData service contains one or more model elements. Some of these elements share a few common characteristics.

## Nominal Elements

Model elements can be nominal in nature. A nominal element has a name of the type <simpleIdentifier> that in combination with a <namespace> produces a fully qualified name of the form <namespaceQualifiedIdentifier>. The <namespaceQualifiedIdentifier> MUST be unique as it facilitates references to the element from other parts of the model.

When referring to nominal elements, the reference can use either of the following:

* Fully qualified name
* Alias qualified name

If the nominal element is unambiguous, the reference can simply use the name of the element.

Consider the following example:

<Schema  
 xmlns=http://schemas.microsoft.com/ado/2006/04/edm  
 xmlns:m=http://schemas.microsoft.com/ado/2007/08/dataservices/metadata  
 xmlns:d=http://schemas.microsoft.com/ado/2007/08/dataservices  
 Namespace="org.odata" Alias="odata">  
 <ComplexType Name="Address">  
 </ComplexType>  
</Schema>

The various ways of referring to the nominal element are:

* References in any namespace can use the fully qualified name, for example, org.odata.Address
* References in any namespace can specify an alias and use an alias qualified name, for example, odata.Address
* References in org.odata can simply use the name, for example, Address

## Structured Elements

Structured elements are composed of other model elements. Structured elements are common in entity models as they are the typical means of representing entities in the OData service. edm:EntityType and edm:ComplexType are both structured elements. edm:RowType is less common but is also a structured element.

A structural property is an edm:Property that has one of the following types:

* Primitive
* edm:ComplexType
* edm:EnumType

## The edm:Documentation Element

The edm:Documentation element allows service authors to provide documentation for most model elements. Refer to the XML schema for details on which elements support edm:Documentation.

For example:

<EntityType Name="Product">   
 <Documentation> Product names, suppliers, prices, and units in stock.</Documentation>  
 ...  
</EntityType>

## Custom Annotations

CSDL allows custom annotations to be attached to many model elements. This allows CSDL to be extended with markup to help various runtimes.

For instance, the following annotations indicate that the Name property should be used as the syndication title in an atom feed:

<Property Name="Name" Nullable="true" Type="Edm.String" m:FC\_KeepInContent="true" m:FC\_ContentKind="text" m:FC\_TargetPath="SyndicationTitle"/>

Custom annotations can appear in attribute form or element form. Refer to the XML schema for details on which elements support custom annotations.

## Primitive Types

Model elements are composed of other model elements and primitive types. CSDL defines the following fully qualified primitive types:

* Edm.Binary
* Edm.Binary
* Edm.Boolean
* Edm.Byte
* Edm.DateTime
* Edm.Decimal
* Edm.Double
* Edm.Single
* Edm.Guid
* Edm.Int16
* Edm.Int32
* Edm.Int64
* Edm.SByte
* Edm.String
* Edm.Time
* Edm.DateTimeOffset
* Edm.Geography
* Edm.GeographyPoint
* Edm.GeographyLineString
* Edm.GeographyPolygon
* Edm.GeographyMultiPoint
* Edm.GeographyMultiLineString
* Edm.GeographyMultiPolygon
* Edm.GeographyCollection
* Edm.Geometry
* Edm.GeometryPoint
* Edm.GeometryLineString
* Edm.GeometryPolygon
* Edm.GeometryMultiPoint
* Edm.GeometryMultiLineString
* Edm.GeometryMultiPolygon
* Edm.GeometryCollection

# Entity Model Wrapper Constructs

An Entity Model Wrapper serves as the aggregation root for the schemas that describe the entity model exposed by the OData Service.

## The edmx:Edmx Element

An OData service exposes a single entity model. A CSDL description of the entity model can be requested from $metadata.

The document returned by $metadata MUST contain a single root edmx:Edmx element. This element MUST contain a single direct child edmx:DataServices element. edmx:DataServices describes the entity model(s) exposed by the OData service.

In addition, edmx:Edmx may have zero or more edmx:Reference elements and zero or more edmx:AnnotationsReference elements. edmx:Reference elements specify the location of schemas referenced by the OData service. edmx:AnnotationsReference elements specify the location of annotations to be applied to the OData service.

### The Version Attribute

The Version attribute MUST be present on the edmx:Edmx element. See <xmlschema> for details.

The Version attribute is a string value that specifies the version of the EDMX wrapper, and must be of the form <majorversion>.<minorversion>. This version of the specification defines version "1.0" of the EDMX Wrapper.

## The edmx:DataServices Element

The edmx:DataServices element contains zero or more edm:Schema elements which define the schema(s) exposed by the OData service.

### The metadata:DataServiceVersion Attribute

The metadata:DataServiceVersion attribute describes the version of OData protocol required to consume the service. This version of the specification defines the following valid data service version values: “1.0”, “2.0”, and “3.0”, corresponding to OData protocol versions 1.0, 2.0 and 3.0 respectively.

## The edmx:Reference Element

The edmx:Reference element specifies external entity models referenced by this EDMX. Referenced models are available in their entirety to referencing models. All entity types, complex types and other named elements in a referenced model can be accessed from a referencing model.

The following example demonstrates usage of edmx:Reference to reference entity models that contain entity types and complex types that are used as vocabulary terms:

<?xml version="1.0" encoding="UTF-8" standalone="true"?>  
<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">  
 <edmx:Reference Url="http://vocabs.odata.org/capabilities/v1.0" />  
 <edmx:Reference Url="http://vocabs.odata.org/display/v1.0" />  
 <edmx:DataServices ...>  
</edmx:Edmx>

### The edmx:Url Attribute

The edmx:Reference element MUST specify an edmx:Url attribute. The edmx:Url attribute uniquely identifies a model. The URL may be backed by a CSDL describing the referenced model. Alternatively, the URL may be used to load a well-known model from a different location.

## The edmx:AnnotationsReference Element

The edmx:AnnotationsReference element specifies the location of an external document that contains annotations for this entity model. Only edm:Annotation, edm:TypeAnnotation and edm:ValueAnnotation elements will be read from the referenced model.

The edmx:AnnotationsReference element MUST contain one or more edmx:Include elements that specify the annotations to include from the target document.

The following example demonstrates usage of edmx:AnnotationsReference to reference documents that contain annotations:

<?xml version="1.0" encoding="UTF-8" standalone="true"?>  
<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">  
 <edmx:AnnotationsReference Url="http://odata.org/ann/a">  
 <edmx:Include />  
 </edmx:AnnotationsReference>  
 <edmx:AnnotationsReference Url="http://odata.org/ann/b">  
 <edmx:Include TermNamespace="org.odata.validation" />  
 <edmx:Include TermNamespace="org.odata.display" Qualifier="Slate" />  
 </edmx:AnnotationsReference>  
 <edmx:DataServices ...>  
</edmx:Edmx>

All annotations from http://odata.org/ann/a are included. For http://odata.org/ann/b, only the following annotations are included:

* Annotations that use a term from org.odata.validation
* Annotations that use a term from org.odata.display and specify a Slate qualifier

### The edmx:Url Attribute

The edmx:AnnotationsReference element MUST specify an edmx:Url attribute. The value of the edmx:Url attribute uniquely identifies a model. The URL may be backed by a CSDL describing the referenced model. Alternatively, the URL may be used to load a well-known model from a different location.

## The edmx:Include Element

The edmx:Include element specifies which annotations to include from an edmx:AnnotationsReference. An edmx:Include that does not have an edmx:TermNamespace attribute or an edmx:Qualifier attribute includes all annotations within the document. If both TermNamespace and Qualifier have values, only annotations that meet both restrictions will be included.

### The edmx:TermNamespace Attribute

An edmx:Include element MAY have a value for the edmx:TermNamespace attribute. A term namespace is a string that disambiguates terms with the same name.

For instance, assume both org.schema and org.microformats define a term named Address. Although the terms have the same name, they are uniquely identifiable since each term is in a model with a unique namespace.

If a value is supplied, the edmx:Include element will add the set of annotations that apply terms from the namespace in the value. The edmx:TermNamespace attribute also provides consumers insight about what namespaces are used in the annotations document. If there are no edmx:Include elements that have a term namespace of interest to the consumer, the consumer can opt to not download the document.

### The edmx:Qualifier Attribute

An edmx:Include element MAY have a value for the edmx:Qualifier attribute. A qualifier is used to apply an annotation to a subset of consumers. For instance, a service author may want to supply a different set of annotations for various device form factors.

If a value is supplied, the edmx:Include element will add the set of annotations that apply the qualifier in the value. The edmx:Qualifier attribute also provides consumers insight about which qualifiers are used in the annotations document. If there are no edmx:Include elements that have a qualifier of interest to the consumer, the consumer can opt to not download the document.

# Schema Constructs

Each entity model exposed by the OData service is described by a schema. The schema contains all of the entity types, complex types and other model elements that make up an entity model.

## The edm:Schema Element

The edm:Schema is the root of an entity model exposed by an OData service. Although an edmx:DataServices element contains zero or more edm:Schema elements, many OData services will contain exactly one schema.

An edm:Schema element contains zero or more of the following elements:

* edm:Annotations
* edm:Association
* edm:ComplexType
* edm:EntityContainer
* edm:EntityType
* edm:EnumType
* edm:Function
* edm:Using
* edm:ValueTerm

### The edm:Namespace Attribute

The edm:Schema is identified by the value of the edm:Namespace attribute. The value of the edm:Namespace attribute is combined with the name of elements in the entity model to create unique names.

Identifiers that are used to name types MUST be unique within an edm:Namespace to prevent ambiguity. See Nominal Elements for more detail.

An edm:Schema that contains constructs other than vocabulary annotations MUST specify a value for the edm:Namespace attribute. A edm:Schema that only contains vocabulary annotations MAY specify a value for the edm:Namespace attribute.

The edm:Namespace attribute MUST NOT use the reserved values "System", "Transient" or "Edm".

### The edm:Alias Attribute

The edm:Schema MAY have a value for the edm:Alias attribute. The edm:Alias attribute allows OData services to substitute a short string for a long namespace. For instance, org.odata.vocabularies.display may simply have an alias of Self. An alias qualified name is resolved to a fully qualified name by examining aliases on edm:Using and edm:Schema.

An edm:Alias is scoped to the container in which it is declared. For example, a model referencing an annotations document cannot use any aliases defined in that annotations document. A referencing model defines its own aliases with the edm:Using element.

## The edm:Using Element

The edm:Using element imports the contents of a specified namespace. An edm:Using element binds an edm:Alias to the namespace of any entity model.

Importing the contents of another model with the edm:Using element may alter the importing model. For instance, a model may import an entity model containing a type derived from a type in the importing model. In that case an edm:EntitySet in the importing model may return either type.

### The edm:Namespace Attribute

The edm:Using element MUST declare an edm:Namespace attribute. The value of the edm:Namespace attribute SHOULD match the namespace of an entity model that is in scope.

### The edm:Alias Attribute

The edm:Using element MUST have a value for the edm:Alias attribute. An alias allows OData services to substitute a short string for a long namespace. For instance, org.odata.vocabularies.display may be bound to an alias of Self. An alias qualified name is resolved to a fully qualified name by examining aliases on edm:Using and edm:Schema.

# Properties

As mentioned in <ref>Structured Elements</ref>, structured elements are composed of other model elements. Structured elements expose a collection of zero or more edm:Property elements.

For example, the following edm:ComplexType has two properties:

<ComplexType Name="Measurement">  
 <Property Name="Dimension" Type="Edm.String" Nullable="false" MaxLength="50" DefaultValue="Unspecified"/>  
 <Property Name="Length" Type="Edm.Decimal" Nullable="false" Precision="18" Scale="2" />  
</ComplexType>

## The edm:Property Element

An edm:Property allows the construction of structured elements from a scalar value or collection of scalar values.

For instance, the following property could be used to hold zero or more strings representing the names of measurement units:

<Property Name="Units" Type="Collection(Edm.String)" Nullable="false"/>

A property MUST have an edm:Name attribute, an edm:Type attribute and zero or more facets. Facets are attributes that modify or constrain the acceptable values for an edm:Property.

### The edm:Name Attribute

An edm:Property MUST specify a value for the edm:Name attribute. The edm:Name attribute allows a name to be assigned to the edm:Property. This name is used when serializing or deserializing OData payloads and can be used for other purposes, such as code generation.

The value of the edm:Name attribute MUST be a <qualifiedTypeName>.

## The edm:Type Attribute

An edm:Property MUST specify a value for the edm:Type attribute. The value of this attribute determines the type for the value of the property on instances of the containing type.

The value of the edm:Type attribute MUST be a local, alias qualified or namespace qualified name. The value of the edm:Type attribute MUST resolve to an edm:ComplexType or a primitive type or a collection of complex or primitive types.

### The edm:Nullable Attribute

Any edm:Property MAY define a value for the edm:Nullable attribute. The value of this attribute determines whether the containing type can be instantiated if no value is provided for the property.

The value of the edm:Nullable attribute SHOULD be true or false. If no value is specified, the edm:Nullable attribute is set to true.

### The edm:MaxLength Attribute

A binary or string edm:Property MAY define a value for the edm:MaxLength attribute. The value of this attribute specifies the maximum length of the value of the property on a type instance.

The value of the edm:MaxLength attribute MUST be a non-negative integer less than or equal to 2,147,483,647.

### The edm:FixedLength Attribute

A binary or string edm:Property MAY define a value for the edm:FixedLength attribute. The value of this attribute specifies the size of the array used to store the value of the property on a type instance.

The value of the edm:FixedLength attribute MUST be a non-negative integer less than or equal to 2,147,483,647.

### The edm:Precision Attribute

A temporal or decimal edm:Property MAY define a value for the edm:Precision attribute. The value of this attribute specifies the maximum number of digits allowed in the value of the property on a type instance.

The value of the edm:Precision attribute MUST be a non-negative integer less than or equal to 2,147,483,647.

### The edm:Scale Attribute

A decimal edm:Property MAY define a value for the edm:Scale attribute. The value of this attribute specifies the maximum number of digits allowed to the right of the decimal point in the value of the property on a type instance.

The value of the edm:Scale attribute MUST be a non-negative integer less than or equal to the value of the edm:Precision attribute.

### The edm:Unicode Attribute

A string edm:Property MAY define a value for the edm:Unicode attribute. The value of this attribute indicates whether the value is stored as Unicode.

The value of the edm:Unicode attribute SHOULD be true or false.

### The edm:Collation Attribute

A string edm:Property MAY define a value for the edm:Collation attribute. The value of this attribute specifies a collation sequence that can be sued for comparison and ordering operations.

The value of the edm:Unicode attribute MUST be one of the following:

* Binary
* Boolean
* Byte
* DateTime
* DateTimeOffset
* Time
* Decimal
* Double
* Single
* Guid
* Int16
* Int32
* Int64
* String
* SByte

### The edm:SRID Attribute

A property that has a <spatialPrimitive> type MAY define a value for the edm:SRID attribute. The value of this attribute identifies which spatial reference system is applied to values of the property on type instances.

The value of the edm:SRID attribute MUST be a <nonNegativeInt32>. If no value is specified, the edm:SRID attribute defaults to 0 for Geometry types or 4326 for Geography types.

### The edm:Default Attribute

An edm:Property MAY define a value for the edm:Default attribute. The value of this attribute determines the value of the property on new type instances.

### The edm:ConcurrencyMode Attribute

Any edm:Property MAY define a value for the edm:ConcurrencyMode attribute. The value of this attribute indicates how concurrency should be handled for the property.

The value of the edm:ConcurrencyMode attribute MUST be None or Fixed. If no value is specified , the edm:ConcurrencyMode attribute defaults to None.

# Entity Type Constructs

Entity types are nominal structured model elements with a key that consists of one or more references to structural properties. An entity type by definition has an independent existence and can be created, updated or deleted independently of any other types. An entity type is the template for an entity: any uniquely identifiable record such as a customer or order.

A key MUST be supplied if and only if the entity type does not specify a base type. The key consists of one or more references to structural properties of the entity type.

An entity type can define two types of properties. A structural property is a named reference to a primitive or complex type, or a collection of primitive or complex types. A navigation property is a named reference to another entity type or collection of entity types.

A simple example of an entity type is as follows:

<EntityType Name="Product">  
 <Key>  
 <PropertyRef Name="ID"/>  
 </Key>  
 <Property Name="ID" Type="Edm.Int32" Nullable="false"/>  
 <Property Name="Name" Type="Edm.String" Nullable="true" />  
 <Property Name="Description" Type="Edm.String" Nullable="true" />  
 <NavigationProperty Name="Category" Relationship="ODataDemo.Product\_Category\_Category\_Products" FromRole="Product\_Category" ToRole="Category\_Products"/>  
 <NavigationProperty Name="Supplier" Relationship="ODataDemo.Product\_Supplier\_Supplier\_Products" FromRole="Product\_Supplier" ToRole="Supplier\_Products"/>  
</EntityType>

The following example shows an entity type based on the previous example:

<EntityType Name="DiscontinuedProduct" BaseType="Product">  
 <Property Name="DiscontinuedDate" Type="Edm.DateTime" Nullable="true"/>  
</EntityType>

## The edm:EntityType Element

The edm:EntityType element represents an entity type in the entity model.

The edm:EntityType element MUST contain exactly one edm:Key element or specify a value for the edm:BaseType, but not both.

If no base type is specified, the edm:EntityType element MUST contain one or more edm:Property elements describing the properties of the entity type. The edm:EntityType element also can contain zero or more edm:NavigationProperty elements.

### The edm:Name Attribute

A value of the form <odataIdentifier> MUST be included for the Name attribute because edm:EntityType is a nominal element. The value identifies the edm:EntityType and MUST be unique within the namespace containing the edm:EntityType.

### The edm:BaseType Attribute

An edm:EntityType can inherit from another edm:EntityType by specifying a <qualifiedEntityTypeName> or an <aliasQualifiedEntityTypeName> for edm:BaseType. An edm:EntityType that has a value for edm:BaseType MUST NOT have a value for edm:Key.

### The edm:Abstract Attribute

An edm:EntityType MAY indicate that it cannot be instantiated by setting the edm:Abstract attribute to true.

### The edm:OpenType Attribute

An edm:EntityType MAY indicate that it can be freely extended by setting the edm:OpenType attribute to true. An open type allows entities to add properties dynamically simply by adding named values to the payload.

## The edm:Key Element

An entity type must be uniquely identifiable. If an edm:EntityType does not specify a base type, the entity type MUST contain exactly one edm:Key element.

The edm:Key element MUST contain one or more references to an edm:Property.

The following entity type has a simple key:

<EntityType Name="Category">  
 <Key>  
 <PropertyRef Name="ID"/>  
 </Key>  
 <Property Name="ID" Type="Edm.Int32" Nullable="false"/>  
 <Property Name="Name" Type="Edm.String" Nullable="true" />  
</EntityType>

The following entity type has a composite key:

<EntityType Name="OrderLine">  
 <Key>  
 <PropertyRef Name="OrderID"/>  
 <PropertyRef Name="LineNumber"/>  
 </Key>  
 <Property Name="OrderID" Type="Edm.Int32" Nullable="true"/>  
 <Property Name="LineNumber" Type="Edm.Int32" Nullable="true"/>  
</EntityType>

## The edm:PropertyRef Element

The edm:PropertyRef element provides a reference to a single property of an entity type. The value supplied to the edm:Name attribute MUST be an <odataIdentifier> that names one of the properties of the entity type.

## The edm:NavigationProperty Element

A navigation property allows navigation from an entity to one or more related entities.

In the following example, the Product entity type has a navigation property to a Category, which has a navigation link back to one or more products:

<EntityType Name="Product">  
 ...  
 <NavigationProperty Name="Category" ToRole="Category" FromRole="Product" Relationship="ODataDemo.ProductCategory"/>  
 <NavigationProperty Name="Supplier" ToRole="Supplier" FromRole="Product" Relationship="ODataDemo.ProductSupplier"/>  
</EntityType>  
<EntityType Name="Category">  
 ...  
 <NavigationProperty Name="Products" ToRole="Product" FromRole="Category" Relationship="ODataDemo.ProductCategory"/>  
</EntityType>

### The edm:Name Attribute

The name of the navigation property is a <simpleIdentifier> that provides a meaningful string to represent the relationship when navigating from the entity that declared the navigation property to the related entity. Each navigation property MUST specify a value for the name.

### The edm:Relationship Attribute

The value of the edm:Relationship attribute is a <qualifiedIdentifier> that provides a reference to an association. The value of the attribute must be the same as the name of an association in the entity model.

### The edm:ToRole Attribute

### The edm:FromRole Attribute

# Complex Type Constructs

Complex types are keyless nominal structured types. The lack of a key means that complex types cannot be created, updated or deleted independently of an entity type. Complex types allow entity models to group properties into common structures if the group of properties does not need to be managed independently.

The following example demonstrates a complex type named Dimensions that is used by two entity types:

<ComplexType Name="Dimensions">  
 <Property Name="Height" Nullable="false" Type="Edm.Decimal"/>  
 <Property Name="Weight" Nullable="false" Type="Edm.Decimal"/>  
 <Property Name="Length" Nullable="false" Type="Edm.Decimal"/>  
</ComplexType>  
<EntityType Name="Product">  
 ...   
 <Property Name="ProductDimensions" Type="Dimensions" />  
 <Property Name="ShippingDimensions" Type="Dimensions" />  
</EntityType>  
<EntityType Name="ShipmentBox">  
 ...  
 <Property Name="Dimensions" Type="Dimensions" />  
</EntityType>

## The edm:ComplexType Element

The edm:ComplexType element represents a complex type in an entity model.

The complex type MUST declare an <odataIdentifier> value for the edm:Name attribute as well as one or more edm:Property elements. Complex types MUST NOT have any navigation properties.

# Enumeration Type Constructs

Enumeration types are nominal scalar types that represent a series of related values. Enumeration types expose these related values as members of the enumeration. Enumeration types generally allow the selection of a single member. The edm:IsFlag attribute allows entity model authors to indicate that more than one value can be selected.

The following example shows a simple flags-enabled enum:

<EnumType Name="FileAccess" UnderlyingType="Edm.Int32" IsFlags="true">  
 <Member Name="Read" Value="1" />  
 <Member Name="Write" Value="2" />  
 <Member Name="Create" Value="4" />  
 <Member Name="Delete" Value="8" />  
</EnumType>

## The edm:Enum Element

The edm:Enum element represents an enumeration type in an entity model.

The edm:Enum element contains zero or more child edm:Member elements enumerating the members of the enum.

### The edm:Name Attribute

The enumeration type MUST provide a <simpleIdentifier> as the value of the name attribute.

### The edm:UnderlyingType Attribute

An enumeration type has an underlying type which specifies the allowable values for member mapping.

The underlying type MUST be an <integralPrimitive>. If the underlying type is not specified, a 32-bit integer MUST be used as the underlying type.

### The edm:IsFlags Attribute

An enumeration type MAY specify a <boolean> value for the edm:IsFlags attribute. A value of <true> indicates that the enumeration type allows multiple members to be selected simultaneously.

## The edm:Member Element

An enumeration type typically has two or more members. Members represent discrete options for the enumeration type.

For example, the following enumeration type has three discrete members:

<EnumType Name=" ShippingMethod">  
 <Member Name="FirstClass" />  
 <Member Name="TwoDay" />  
 <Member Name="Overnight" />  
</EnumType>

### The edm:Name Attribute

The name of an enum member is a <simpleIdentifier> that allows the enum member to be referenced. The enumeration type MUST NOT declare two members with the same name.

### The edm:Value Attribute

The value of an enum member allows entity instances to be sorted by a property that has an enum member for its value. If the value is not explicitly set, the value MUST be assigned to zero for the first member or one plus the previous member value for any subsequent members.

In the example that follows, FirstClass MUST be assigned a value of 0, TwoDay a value of 4, and Overnight a value of 5.

<EnumType Name="ShippingMethod">  
 <Member Name="FirstClass" />  
 <Member Name="TwoDay" Value="4" />  
 <Member Name="Overnight" />  
</EnumType>

# Association Constructs

Associations provide the fundamental definition for a relationship between two entity types. An association MUST have a name and two ends, each with their own cardinality. In some cases, an association also has a referential constraint. A referential constraint asserts that the entity type playing the principal role in an association MUST exist for the entity type playing the dependent role to exist.

A simple association might look like this:

<Association Name="ProductCategory">   
 <End Type="ODataDemo.Product" Multiplicity="\*" Role="Product"/>  
 <End Type="ODataDemo.Category" Multiplicity="0..1" Role="Category"/>  
</Association>

## The edm:Association Element

The edm:Association element represents an association in an entity model.

The edm:Association element MUST contain exactly two edm:End elements that represent the respective ends of the association. The association can also specify zero or one edm:ReferentialConstraint elements.

### The edm:Name Attribute

The association MUST provide a <simpleIdentifier> as the value of the name attribute.

## The edm:End Element

An association has exactly two ends, each of which plays a role, has its own multiplicity and refers to the entity type at that end of the association.

Each end can also have zero or one edm:OnDelete elements.

### The edm:Type Attribute

Each end of the association MUST specify the entity type attached to that end. The value of the edm:Type attribute is a <qualifiedIdentifier> that names an entity type.

### The edm:Role Attribute

The edm:Role attribute allows the association end to be bound to a navigation property. The value of the attribute MUST be a <simpleIdentifier>.

### The edm:Multiplicity Attribute

The edm:Multiplicity attribute defines the cardinality of the association end. The value of the attribute MUST be one of the following:

* 0..1 – zero or one
* 1 – exactly one
* \* – zero or more

## The edm:OnDelete Element

The edm:OnDelete element prescribes the action that should be taken when the entity on the opposing end of the association is deleted.

If present, the edm:OnDelete element MUST define a value for the edm:Action attribute. There are two valid values for the action: Cascade or None.

## The edm:ReferentialConstraint Element

A referential constraint asserts that the entity on the principal end of the referential constraint must exist in order for the entity on the dependent end to exist. This assertion is established by the edm:ReferentialConstraint element.

An edm:ReferentialConstraint MUST contain exactly one edm:Principal element and one edm:Dependent element.

In the example that follows, the category must exist for a product in that category to exist:

<Association Name="ProductCategory">   
 <End Type="ODataDemo.Product" Multiplicity="\*" Role="Product"/>  
 <End Type="ODataDemo.Category" Multiplicity="0..1" Role="Category"/>  
 <ReferentialConstraint>  
 <Principal Role="Category">  
 <PropertyRef Name="CategoryID" />  
 </Principal>  
 <Dependent Role="Product">  
 <PropertyRef Name="CategoryID" />  
 </ Dependent>  
 </ReferentialConstraint>  
</Association>

## The edm:Principal Element

The edm:Principal element represents the dominant end of the association. The entity on this end of the association may exist independently of the entity on the dependent end of the association.

The edm:Principal element MUST provide a <simpleIdentifier> value for the edm:Role attribute. This value MUST correspond to the edm:Role attribute of one of the association ends.

## The edm:Dependent Element

The edm:Dependent element represents the subordinate end of the association. The entity on this end of the association may only exist if an entity on the opposite end of the association exists.

The edm:Dependent element MUST provide a <simpleIdentifier> value for the edm:Role attribute. This value MUST correspond to the edm:Role attribute of one of the association ends.

## The edm:PropertyRef Element

The edm:PropertyRef elements indicate the fields that take part in the referential constraint on each entity.

The edm:Principal element and the edm:Dependent element MUST each contain the same number of edm:PropertyRef elements. The property references MUST use the same <simpleIdentifier> and be ordered consistently in the edm:Principal and edm:Dependent elements.

The property references for the principal entity MUST be the same property references specified in the edm:Key of the principal entity type, however they need not share the same order.

The following example is NOT valid because the order of the edm:PropertyRef elements is inconsistent:

<Association Name="OrderLineDetails">   
 <End Type="ODataDemo.OrderLineDetail" Multiplicity="\*" Role="Detail"/>  
 <End Type="ODataDemo.OrderLine" Multiplicity="0..1" Role="OrderLine"/>  
 <ReferentialConstraint>  
 <Principal Role="OrderLine">  
 <PropertyRef Name="OrderID" />  
 <PropertyRef Name="LineNumber" />  
 </Principal>  
 <Dependent Role="Detail">  
 <PropertyRef Name="LineNumber" />  
 <PropertyRef Name="OrderID" />  
 </ Dependent>  
 </ReferentialConstraint>  
</Association>

# Entity Container Constructs

An entity model can also describe how entities are logically grouped and even model the store or stores from which the entities can be retrieved. This is achieved through the declaration of entity containers, entity sets and association sets.

An entity set is a nominal element that allows access to entity instances. Simple entity models frequently have one entity set per entity type, for example:

<EntitySet Name="Products" EntityType="ODataDemo.Product"/>  
<EntitySet Name="Categories" EntityType="ODataDemo.Category"/>

Other entity models may expose multiple entity sets per type. For instance, an entity model may have the following entity sets:

<EntitySet Name="Products" EntityType="ODataDemo.Product"/>  
<EntitySet Name="DiscontinuedProducts" EntityType="ODataDemo.Product"/>

In this case the Products entity set could expose products that have not been discontinued and the DiscontinuedProducts entity set could expose products that have been discontinued.

An entity set can expose instances of the specified entity type as well as any entity type inherited from the specified entity type.

An association set is a nominal element that disambiguates the entity set used by an association end. In the preceding example an ODataDemo.Product could be exposed through the Products entity set or the DiscontinuedProducts entity set. An association set is required to clarify which entity set is used for any association end that refers to ODataDemo.Product.

A function import is used to expose functions that are defined in a data store. For example, the following function import exposes a stored procedure that returns the top ten revenue generating products for a given fiscal year:

<FunctionImport Name="TopTenProductsByRevenue" EntitySet="Products" ReturnType="Collection(ODataDemo.Product)">  
 <Parameter Name="fiscalyear" Mode="In" Type="String" />  
</FunctionImport>

An entity container is the entity model equivalent of a single data store. An entity container aggregates entity sets, association sets and function imports.

A full example of an entity container is as follows:

<EntityContainer Name="DemoService">  
 <EntitySet Name="Products" EntityType="ODataDemo.Product"/>  
 <EntitySet Name="Categories" EntityType="ODataDemo.Category"/>  
 <EntitySet Name="Suppliers" EntityType="ODataDemo.Supplier"/>  
 <AssociationSet Name="ProductCategorySet" Association="ODataDemo.ProductCategory">  
 <End Role="Product" EntitySet="Products"/>  
 <End Role="Category" EntitySet="Categories"/>  
 </AssociationSet>  
 <AssociationSet Name="ProductSupplierSet" Association="ODataDemo.ProductSupplier">  
 <End Role="Product" EntitySet="Products"/>  
 <End Role="Supplier" EntitySet="Suppliers"/>  
 </AssociationSet>  
 <FunctionImport Name="GetProductsByRating" EntitySet="Products" ReturnType="Collection(ODataDemo.Product)">  
 <Parameter Name="rating" Type="Edm.Int32" Mode="In"/>  
 </FunctionImport>  
</EntityContainer>

## The edm:EntityContainer Element

The edm:EntityContainer element represents an entity container in an entity model. It corresponds to a logical data store and contains zero or more edm:EntitySet, edm:AssociationSet and edm:FunctionImport elements.

The edm:EntityContainer MUST provide a <simpleIdentifier> value for the edm:Name attribute.

The edm:EntityContainer can also provide a <qualifiedIdentifier> value for the edm:Extends attribute.

## The edm:EntitySet Element

The edm:EntitySet element is a nominal element that represents an entity set in an entity model.

An entity set MUST provide a <simpleIdentifier> value for the edm:Name attribute. An entity set also has a edm:EntityType attribute that MUST be provided with a <qualifiedIdentifier> that resolves to an entity type in the model.

## The edm:AssociationSet Element

The edm:AssociationSet element is a nominal element that represents an association set in an entity model.

An association set MUST provide a <simpleIdentifier> value for the edm:Name attribute. An association set also has an edm:Association attribute that MUST be provided with a <qualifiedIdentifier> that resolves to an association in the entity model.

An association set can contain zero, one or two edm:End elements.

### The edm:End Element

An edm:End element MUST be specified if the corresponding end of the association refers to an entity type that is exposed in more than one entity set.

The edm:End element MUST provide a <simpleIdentifier> value for the edm:Role attribute that is the same as the value of one of the association ends’ edm:Role attribute.

The edm:End element MUST provide a <simpleIdentifier> value for the edm:EntitySet attribute. The entity set that is named MUST expose the entity type bound by the corresponding end of the association.

## The edm:FunctionImport Element

The edm:FunctionImport element is a nominal element that represents a store function in an entity model. The function may return a single or collection of primitive types or nominal structured types.

The return type for the function import MUST be specified. The return type can be specified using either attribute notation or element notation.

### The edm:Name Attribute

A function import MUST provide a <simpleIdentifier> value for the edm:Name attribute.

### The edm:ReturnType Attribute

If the return type is specified using attribute notation, an <anyTypeIdentifier> value must be provided for the edm:ReturnType attribute.

If a value is provided for the edm:ReturnType attribute, the edm:FunctionImport element MUST NOT contain a edm:ReturnType element.

### The edm:EntitySet Attribute

If the return type is an entity or a collection of entities, a <simpleIdentifier> value must be defined for the edm:EntitySet attribute that names the entity set to which the returned entities belong.

If the return type is not an entity or a collection of entities, a value MUST NOT be defined for the edm:EntitySet attribute.

### The edm:EntitySetPath Attribute

The function import MAY specify a value for the edm:EntitySetPath attribute if determination of the entity set for the return type is contingent on a parameter.

The value for the edm:EntitySetPath attribute consists of a series of segments joined together with forward slashes.

The first segment of the entity set path MUST have the same name as one of the function import’s parameters. The remaining segments of the entity set path MUST represent navigation or type casts.

A navigation segment simply names the <simpleIdentifier> of the navigation property to be traversed. A type cast segment names the <qualifiedIdentifier> of the entity type that should be returned from the type cast.

## The edm:ReturnType Element

If the return type is specified using element notation, the function import MUST contain one or more edm:ReturnType elements. The presence of multiple edm:ReturnType elements may occur when the store returns more than one result set for a given function.

### The edm:Type Attribute

### The edm:EntitySet Attribute

## The edm:Parameter Element

### The edm:Name Attribute

### The edm:Type Attribute

### The edm:Mode Attribute

### The edm:MaxLength Attribute

### The edm:Precision Attribute

### The edm:Scale Attribute

# Annotation Constructs

## The edm:TypeAnnotation Element

## The edm:PropertyValue Element

### The edm:Path Attribute

### The edm:String Attribute

### The edm:Int Attribute

### The edm:Float Attribute

### The edm:Decimal Attribute

### The edm:DateTime Attribute

## The edm:ValueAnnotation Element

### The edm:Path Attribute

### The edm:String Attribute

### The edm:Int Attribute

### The edm:Float Attribute

### The edm:Decimal Attribute

### The edm:DateTime Attribute

## The edm:ValueTerm Element

### The edm:Name Attribute

### The edm:Type Attribute

# Expression Constructs

Expressions are used to specify values in annotations. They may appear as a direct child of an edm:PropertyValue or an edm:ValueAnnotation.

# CSDL Examples

Something about these two examples.

## Title for example

<edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">

<edmx:DataServices xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"m:DataServiceVersion="2.0">

<Schema xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"xmlns="http://schemas.microsoft.com/ado/2007/05/edm" Namespace="ODataDemo">

<EntityType Name="Product">

<Key>

<PropertyRef Name="ID"/>

</Key>

<Property Name="ID" Type="Edm.Int32" Nullable="false"/>

<Property Name="Name" Type="Edm.String" Nullable="true" m:FC\_TargetPath="SyndicationTitle"m:FC\_ContentKind="text" m:FC\_KeepInContent="false"/>

<Property Name="Description" Type="Edm.String" Nullable="true" m:FC\_TargetPath="SyndicationSummary"m:FC\_ContentKind="text" m:FC\_KeepInContent="false"/>

<Property Name="ReleaseDate" Type="Edm.DateTime" Nullable="false"/>

<Property Name="DiscontinuedDate" Type="Edm.DateTime" Nullable="true"/>

<Property Name="Rating" Type="Edm.Int32" Nullable="false"/>

<Property Name="Price" Type="Edm.Decimal" Nullable="false"/>

<NavigationProperty Name="Category" Relationship="ODataDemo.Product\_Category\_Category\_Products"FromRole="Product\_Category" ToRole="Category\_Products"/>

<NavigationProperty Name="Supplier" Relationship="ODataDemo.Product\_Supplier\_Supplier\_Products"FromRole="Product\_Supplier" ToRole="Supplier\_Products"/>

</EntityType>

<EntityType Name="Category">

<Key>

<PropertyRef Name="ID"/>

</Key>

<Property Name="ID" Type="Edm.Int32" Nullable="false"/>

<Property Name="Name" Type="Edm.String" Nullable="true" m:FC\_TargetPath="SyndicationTitle"m:FC\_ContentKind="text" m:FC\_KeepInContent="true"/>

<NavigationProperty Name="Products" Relationship="ODataDemo.Product\_Category\_Category\_Products"FromRole="Category\_Products" ToRole="Product\_Category"/>

</EntityType>

<EntityType Name="Supplier">

<Key>

<PropertyRef Name="ID"/>

</Key>

<Property Name="ID" Type="Edm.Int32" Nullable="false"/>

<Property Name="Name" Type="Edm.String" Nullable="true" m:FC\_TargetPath="SyndicationTitle"m:FC\_ContentKind="text" m:FC\_KeepInContent="true"/>

<Property Name="Address" Type="ODataDemo.Address" Nullable="false"/>

<Property Name="Concurrency" Type="Edm.Int32" Nullable="false" ConcurrencyMode="Fixed"/>

<NavigationProperty Name="Products" Relationship="ODataDemo.Product\_Supplier\_Supplier\_Products"FromRole="Supplier\_Products" ToRole="Product\_Supplier"/>

</EntityType>

<ComplexType Name="Address">

<Property Name="Street" Type="Edm.String" Nullable="true"/>

<Property Name="City" Type="Edm.String" Nullable="true"/>

<Property Name="State" Type="Edm.String" Nullable="true"/>

<Property Name="ZipCode" Type="Edm.String" Nullable="true"/>

<Property Name="Country" Type="Edm.String" Nullable="true"/>

</ComplexType>

<Association Name="Product\_Category\_Category\_Products">

<End Role="Product\_Category" Type="ODataDemo.Product" Multiplicity="\*"/>

<End Role="Category\_Products" Type="ODataDemo.Category" Multiplicity="0..1"/>

</Association>

<Association Name="Product\_Supplier\_Supplier\_Products">

<End Role="Product\_Supplier" Type="ODataDemo.Product" Multiplicity="\*"/>

<End Role="Supplier\_Products" Type="ODataDemo.Supplier" Multiplicity="0..1"/>

</Association>

<EntityContainer Name="DemoService" m:IsDefaultEntityContainer="true">

<EntitySet Name="Products" EntityType="ODataDemo.Product"/>

<EntitySet Name="Categories" EntityType="ODataDemo.Category"/>

<EntitySet Name="Suppliers" EntityType="ODataDemo.Supplier"/>

<AssociationSet Name="Products\_Category\_Categories"Association="ODataDemo.Product\_Category\_Category\_Products">

<End Role="Product\_Category" EntitySet="Products"/>

<End Role="Category\_Products" EntitySet="Categories"/>

</AssociationSet>

<AssociationSet Name="Products\_Supplier\_Suppliers"Association="ODataDemo.Product\_Supplier\_Supplier\_Products">

<End Role="Product\_Supplier" EntitySet="Products"/>

<End Role="Supplier\_Products" EntitySet="Suppliers"/>

</AssociationSet>

<FunctionImport Name="GetProductsByRating" EntitySet="Products"ReturnType="Collection(ODataDemo.Product)" m:HttpMethod="GET">

<Parameter Name="rating" Type="Edm.Int32" Mode="In"/>

</FunctionImport>

</EntityContainer>

</Schema>

</edmx:DataServices>

</edmx:Edmx>

# ABNF for CSDL

identifierLeadingCharacter = ; Any character from the Unicode classes L or Nl

identifierCharacter = ; Any character from the Unicode classes L, Nl, Nd, Mn, Mc, Pc or Cf

simpleIdentifier = identifierLeadingCharacter \*identifierCharacter

qualifiedIdentifier = namespaceQualifiedIdentifier /  
 aliasQualifiedIdentifier

anyTypeReference = "collection(" anySingleTypeReference ")" /  
 anySingleTypeReference

anyNonKeyedTypeReference = "collection(" singleNonKeyedTypeReference ")" /  
 singleNonKeyedTypeReference

anySingleTypeReference = singleEntityTypeReference /  
 singleNonKeyedTypeReference

singleEntityTypeReference = qualifiedIdentifier / simpleIdentifier  
 ; identifies an EntityType by name

singleNonKeyedTypeReference = qualifiedIdentifier / simpleIdentifier  
 ; identifies a primitive, ComplexType or EnumType by name

namespaceSegment = simpleIdentifier

alias = simpleIdentifier

namespaceQualifiedIdentifier = namespace "." simpleIdentifier

aliasQualifiedIdentifier = alias "." simpleIdentifier

namespace = namespaceSegment \*("." namespaceSegment)

coreModel = "edm."

true = "true" / "1"

false = "false" / "0"

boolean = true / false

integralPrimitive = [coreModel] (  
 "byte" /  
 "int16" /  
 "int32" /  
 "int64" /  
 "sbyte"  
 )

spatialPrimitive = [coreModel] (  
 "geography" /  
 "geographypoint" /  
 "geographylinestring" /  
 "geographypolygon" /  
 "geographymultipoint" /  
 "geographymultilinestring" /  
 "geographymultipolygon" /  
 "geographycollection" /  
 "geometry" /  
 "geometrypoint" /  
 "geometrylinestring" /  
 "geometrypolygon" /  
 "geometrymultipoint" /  
 "geometrymultilinestring" /  
 "geometrymultipolygon" /  
 "geometrycollection" /  
 )

nonNegativeInt32 = 1\*10DIGIT ; 0-2147483647

primitive = ["edm."] (  
 integralPrimitive /  
 spatialPrimitive /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 "" /  
 )