[MS-BDCDPS2]:
Business Data Connectivity Database
Version 2 Protocol 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.
**Preliminary Documentation.** This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

**Revision Summary**

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/20/2012</td>
<td>0.1</td>
<td>New</td>
<td>Released new document.</td>
</tr>
<tr>
<td>04/11/2012</td>
<td>0.1</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>07/16/2012</td>
<td>0.1</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction .......................................................................................................... 9
   1.1 Glossary ........................................................................................................ 9
   1.2 References .................................................................................................... 11
      1.2.1 Normative References ............................................................................. 11
      1.2.2 Informative References ......................................................................... 11
   1.3 Overview ....................................................................................................... 11
   1.4 Relationship to Other Protocols ..................................................................... 12
   1.5 Prerequisites/Preconditions ........................................................................... 12
   1.6 Applicability Statement ................................................................................ 12
   1.7 Versioning and Capability Negotiation .......................................................... 13
   1.8 Vendor-Extensible Fields .............................................................................. 13
   1.9 Standards Assignments ................................................................................ 13

2 Messages ............................................................................................................... 14
   2.1 Transport ...................................................................................................... 14
   2.2 Common Data Types ...................................................................................... 14
      2.2.1 Common Fields ....................................................................................... 14
         2.2.1.1 Id ...................................................................................................... 14
         2.2.1.2 Name ................................................................................................ 14
         2.2.1.3 Namespace ....................................................................................... 14
         2.2.1.4 PartitionId ....................................................................................... 14
         2.2.1.5 IsCached ......................................................................................... 14
         2.2.1.6 SettingId ........................................................................................... 14
         2.2.1.7 MajorVersion ................................................................................... 15
         2.2.1.8 MinorVersion .................................................................................. 15
         2.2.1.9 BuildVersion .................................................................................... 15
         2.2.1.10 RevisionVersion ............................................................................. 15
         2.2.1.11 EstimatedInstanceCount ................................................................. 15
         2.2.1.12 IsActive .......................................................................................... 15
         2.2.1.13 CacheUsage .................................................................................... 15
         2.2.1.14 Position .......................................................................................... 16
         2.2.1.15 IsDisplayed ..................................................................................... 16
         2.2.1.16 IsOpenedInNewWindow .................................................................. 16
         2.2.1.17 Icon ................................................................................................ 16
         2.2.1.18 URL ................................................................................................ 16
         2.2.1.19 Index ............................................................................................... 16
         2.2.1.20 FilterType ......................................................................................... 16
         2.2.1.21 FilterField ....................................................................................... 17
         2.2.1.22 IdentifierTypeName ........................................................................ 18
         2.2.1.23 MethodInstanceType ...................................................................... 18
         2.2.1.24 Direction .......................................................................................... 20
         2.2.1.25 TypeDescriptorTypeName ............................................................... 20
         2.2.1.26 TypeDescriptorLobName ................................................................. 20
         2.2.1.27 TypeDescriptorFlags ...................................................................... 21
         2.2.1.28 TypeDescriptorInterpretation ......................................................... 21
         2.2.1.29 DefaultValue .................................................................................. 21
         2.2.1.30 SystemType .................................................................................... 21
         2.2.1.31 SystemData ..................................................................................... 22
         2.2.1.32 MetadataRights .............................................................................. 22
         2.2.1.33 IsStatic ............................................................................................. 22
         2.2.1.34 IsOpenedInNewWindow .................................................................. 16
         2.2.1.35 IsDisplayed ..................................................................................... 16
         2.2.1.36 IsOpenedInNewWindow .................................................................. 16
         2.2.1.37 IsDisplayed ..................................................................................... 16
2.2.5   Result Sets .................................................................................................... 34

2.2.3   Bit Fields and Flag Structures ........................................................................... 33

2.2.2   Simple Data Types and Enumerations ............................................................... 24

2.2.5.17   FilterDescriptor Result Set ........................................................................ 44

2.2.5.16   Entity Name Result Set ............................................................................ 44

2.2.5.15   Entity Result Set ..................................................................................... 43

2.2.5.14   DefaultValues Result Set .......................................................................... 42

2.2.5.13   DataClass Result Set ............................................................................... 41

2.2.5.12   TypeDescriptor Result Set ........................................................................ 39

2.2.5.10   AssociationReference Result Set ............................................................... 38

2.2.5.9   Association Member Result Set ................................................................... 38

2.2.5.8   Association Group Result Set ...................................................................... 37

2.2.5.7   Association Result Set ............................................................................... 36

2.2.5.6   Setting Result Set ..................................................................................... 36

2.2.5.4   LocalizedName Result Set .......................................................................... 36

2.2.5.3   MetadataCatalog Result Set ....................................................................... 35

2.2.5.2   Count Result Set ...................................................................................... 35

2.2.5.1   Action Result Set ...................................................................................... 34

2.2.1.34   MethodLobName ..................................................................................... 22

2.2.1.35   IsDefault ............................................................................................... 23

2.2.1.36   SessionId .............................................................................................. 23

2.2.1.37   IsReverse ............................................................................................... 23

2.2.1.38   ThrottleScope ......................................................................................... 23

2.2.1.39   ThrottleType .......................................................................................... 24

2.2.1.40   ThrottleConfigEnabled ............................................................................ 24

2.2.1.41   ActionParameterValue .......................................................................... 24

2.2.2   Simple Data Types and Enumerations ............................................................... 24

2.2.2.1   MetadataObject ........................................................................................ 24

2.2.2.2   Property .................................................................................................... 25

2.2.2.3   Localized Name ........................................................................................ 25

2.2.2.4   Access Control Entry .................................................................................. 25

2.2.2.5   Model ........................................................................................................ 26

2.2.2.6   LobSystem ............................................................................................... 26

2.2.2.7   LobSystemInstance .................................................................................... 26

2.2.2.8   DataClass ................................................................................................ 27

2.2.2.9   Entity ...................................................................................................... 27

2.2.2.10  Identifier ................................................................................................. 28

2.2.2.11  Method ..................................................................................................... 28

2.2.2.12  MethodInstance ....................................................................................... 29

2.2.2.13  Association ............................................................................................. 29

2.2.2.14  Parameter ............................................................................................... 29

2.2.2.15  TypeDescriptor ....................................................................................... 30

2.2.2.16  FilterDescriptor ...................................................................................... 30

2.2.2.17  DefaultValue ......................................................................................... 31

2.2.2.18  AssociationGroup .................................................................................... 31

2.2.2.19  AssociationReference ............................................................................. 31

2.2.2.20  Action ..................................................................................................... 31

2.2.2.21  ActionParameterValue .......................................................................... 32

2.2.2.22  Cache Version Stamp ............................................................................. 32

2.2.2.23  Throttle Configuration Setting .................................................................. 32

2.2.3   Bit Fields and Flag Structures ......................................................................... 33

2.2.3.1   CacheLine ................................................................................................ 33

2.2.4   Binary Structures ......................................................................................... 34

2.2.5   Result Sets .................................................................................................... 34

2.2.5.1   Action Result Set ...................................................................................... 34

2.2.5.2   Count Result Set ...................................................................................... 35

2.2.5.3   MetadataCatalog Result Set ....................................................................... 35

2.2.5.4   LocalizedName Result Set .......................................................................... 36

2.2.5.5   Partition Result Set ................................................................................... 36

2.2.5.6   Setting Result Set ..................................................................................... 36

2.2.5.7   Association Result Set ............................................................................... 36

2.2.5.8   Association Group Result Set .................................................................... 37

2.2.5.9   Association Member Result Set .................................................................. 38

2.2.5.10  AssociationReference Result Set ............................................................... 38

2.2.5.11  Cache Version Stamps Result Set ............................................................... 39

2.2.5.12  TypeDescriptor Result Set ....................................................................... 39

2.2.5.13  DataClass Result Set ............................................................................... 41

2.2.5.14  DefaultValues Result Set .......................................................................... 42

2.2.5.15  Entity Result Set ..................................................................................... 43

2.2.5.16  Entity Name Result Set ............................................................................ 44

2.2.5.17  FilterDescriptor Result Set ...................................................................... 44
2.2.5.18 Identifier Result Set .......................................................... 45
2.2.5.19 Property Result Set ............................................................ 45
2.2.5.20 Method Result Set ............................................................... 46
2.2.5.21 MethodInstance Result Set .................................................. 46
2.2.5.22 Model Result Set ................................................................. 47
2.2.5.23 Parameter Result Set ........................................................... 47
2.2.5.24 Throttle Setting Result Set ................................................... 48
2.2.5.25 System Result Set ................................................................. 49
2.2.5.26 System Data Result Set ......................................................... 49
2.2.5.27 SystemInstance Result Set .................................................... 49
2.2.5.28 Access Control Entry Result Set .......................................... 50
2.2.5.29 Id Result Set .................................................................. 50
2.2.5.30 Progress Result Set ............................................................. 50
2.2.5.31 Activation Errors Result Set ............................................... 51
2.2.5.32 Action Parameter Result Set ............................................... 51

2.2.6 Tables and Views ........................................................................ 55
2.2.7 XML Structures ........................................................................... 55
2.2.7.1 Namespaces .......................................................................... 55
2.2.7.2 Simple Types ......................................................................... 55
2.2.7.3 Complex Types ..................................................................... 55
2.2.7.4 Elements .............................................................................. 55
2.2.7.5 Attributes ............................................................................ 55
2.2.7.6 Groups ................................................................................. 55
2.2.7.7 Attribute Groups .................................................................. 55

3 Protocol Details ................................................................................. 56
3.1 Common Details ............................................................................. 56
3.2 Server Details ................................................................................ 56
3.2.1 Abstract Data Model ................................................................. 56
3.2.2 Timers ..................................................................................... 62
3.2.3 Initialization ............................................................................. 62
3.2.4 Higher-Layer Triggered Events ................................................. 62
3.2.5 Message Processing Events and Sequencing Rules ...................... 62
3.2.5.1 proc_ar_ActivateEntity ......................................................... 62
3.2.5.2 proc_ar_AddEntity ............................................................... 64
3.2.5.3 proc_ar_AddOrInsertLocalizedStringForMetadataObjectId .... 64
3.2.5.4 proc_ar_AddOrInsertPropertyForMetadataObjectId ............. 65
3.2.5.5 proc_ar_BulkSwitchActive .................................................... 66
3.2.5.6 proc_ar_BumpCacheInvalidationCounters ............................. 66
3.2.5.7 proc_ar_ClearAccessControlEntriesForMetadataObject ......... 69
3.2.5.8 proc_ar_CopyAccessControlEntriesForMetadataObjectId ...... 69
3.2.5.9 proc_ar_CopyAccessControlEntriesForSettings .................... 70
3.2.5.10 proc_ar_CreateAction ....................................................... 70
3.2.5.11 proc_ar_CreateActionParameter ....................................... 71
3.2.5.12 proc_ar_CreateAdministrationMetadataCatalog ................ 73
3.2.5.13 proc_ar_CreateAssociation ................................................. 73
3.2.5.14 proc_ar_CreateAssociationGroup ....................................... 75
3.2.5.15 proc_ar_CreateAssociationReference ................................ 76
3.2.5.16 proc_ar_CreateEntity ......................................................... 78
3.2.5.17 proc_ar_CreateFilterDescriptor ......................................... 79
3.2.5.18 proc_ar_CreateIdentifier .................................................. 81
3.2.5.19 proc_ar_CreateMethod ....................................................... 82
3.2.5.20 proc_ar_CreateMethodInstance ......................................... 83
3.2.5.21 proc_ar_CreateModel ................................................................. 85
3.2.5.22 proc_ar_CreateParameter .......................................................... 86
3.2.5.23 proc_ar_CreateSystem ............................................................... 87
3.2.5.24 proc_ar_CreateSystemInstance ................................................... 88
3.2.5.25 proc_ar_CreateTypeDescriptor .................................................... 89
3.2.5.26 proc_ar_DeactivateEntity ............................................................ 92
3.2.5.27 proc_ar_DeleteActionById ........................................................... 93
3.2.5.28 proc_ar_DeleteActionParameterById ............................................. 94
3.2.5.29 proc_ar_DeleteAdministrationMetadataCatalog ......................... 95
3.2.5.30 proc_ar_DeleteAssociationById ................................................... 96
3.2.5.31 proc_ar_DeleteAssociationGroupId .............................................. 97
3.2.5.32 proc_ar_DeleteAssociationReferenceById ..................................... 98
3.2.5.33 proc_ar_DeleteDefaultValue ....................................................... 99
3.2.5.34 proc_ar_DeleteEntityById ............................................................ 100
3.2.5.35 proc_ar_DeleteFilterDescriptorById .......................................... 101
3.2.5.36 proc_ar_DeleteIdentifierById ...................................................... 102
3.2.5.37 proc_ar_DeleteLocalizedNameForMetadataObjectByLCID ............... 103
3.2.5.38 proc_ar_DeleteLocalizedNamesByMetadataObjectId ...................... 104
3.2.5.39 proc_ar_DeleteMethodById .......................................................... 105
3.2.5.40 proc_ar_DeleteMethodInstanceById ............................................. 106
3.2.5.41 proc_ar_DeleteModelById ........................................................... 107
3.2.5.42 proc_ar_DeleteParameterById ...................................................... 108
3.2.5.43 proc_ar_DeletePropertiesById ..................................................... 109
3.2.5.44 proc_ar_DeletePropertyForMetadataObjectId ............................... 110
3.2.5.45 proc_ar_DeleteSystemById .......................................................... 111
3.2.5.46 proc_ar_DeleteSystemInstanceById ............................................. 112
3.2.5.47 proc_ar_DeleteTypeDescriptorById ............................................. 113
3.2.5.48 proc_ar_GetAccessControlEntriesForMetadataObject .................... 114
3.2.5.49 proc_ar_GetActionById ............................................................... 115
3.2.5.50 proc_ar_GetActionParameterById ................................................ 115
3.2.5.51 proc_ar_GetActionParametersForActionWithCount .......................... 116
3.2.5.52 proc_ar_GetActionsForEntityWithCount ...................................... 116
3.2.5.53 proc_ar_GetAdministrationMetadataCatalogById ......................... 117
3.2.5.54 proc_ar_GetAdministrationMetadataCatalogByPartitionId ............... 117
3.2.5.55 proc_ar_GetAllLocalizedNamesForMetadataObjectWithCount ............ 118
3.2.5.56 proc_ar_GetAllMergedLocalizedNamesForMetadataObjectWithCount .... 118
3.2.5.57 proc_ar_GetAllPartitionIds ........................................................ 118
3.2.5.58 proc_ar_GetAllSlicesForMetadataObjectId .................................... 118
3.2.5.59 proc_ar_GetAssociationById ....................................................... 119
3.2.5.60 proc_ar_GetAssociationGroupId ................................................ 119
3.2.5.61 proc_ar_GetAssociationGroupsForEntityWithCount ........................ 120
3.2.5.62 proc_ar_GetAssociationMembersInRoleWithCount .......................... 120
3.2.5.63 proc_ar_GetAssociationReferencesForAssociationGroupWithCount .... 121
3.2.5.64 proc_ar_GetAssociationsForDataClassWithCount .......................... 121
3.2.5.65 proc_ar_GetAssociationsForEntityAndRoleWithCount ..................... 122
3.2.5.66 proc_ar_GetAssociationsForMethodWithCount ................................ 123
3.2.5.67 proc_ar_GetCacheInvalidationCountersWithCount .......................... 123
3.2.5.68 proc_ar_GetChildTypeDescriptorsForTypeDescriptorWithCount .......... 123
3.2.5.69 proc_ar_GetDataClassById .......................................................... 124
3.2.5.70 proc_ar_GetDataClassesForSystemWithCount .................................. 124
3.2.5.71 proc_ar_GetDefaultValuesForTypeDescriptor .................................. 125
3.2.5.72 proc_ar_GetEntitiesForAssociationAndRoleWithCount .................... 126
3.2.5.73 proc_ar_GetEntitiesForSystemCount ........................................... 126
3.2.5.74 proc_ar_GetEntitiesForSystemWithCount .................................................. 127
3.2.5.75 proc_ar_GetEntitiesLikeNameAndNamespace ............................................ 127
3.2.5.76 proc_ar_GetEntitiesReferencedByModelId ................................................. 128
3.2.5.77 proc_ar_GetEntityById ............................................................................... 129
3.2.5.78 proc_ar_GetEntityNamesForAssociationAndRole ....................................... 130
3.2.5.79 proc_ar_GetEntityWithNamespaceAndVersion ............................................. 130
3.2.5.80 proc_ar_GetEntityWithNamespaceAndVersionAndVersion ............................ 131
3.2.5.81 proc_ar_GetFilterDescriptorById .............................................................. 131
3.2.5.82 proc_ar_GetFilterDescriptorsForMethodWithCount .................................... 132
3.2.5.83 proc_ar_GetIdentifierById ........................................................................ 132
3.2.5.84 proc_ar_GetIdentifiersForEntityWithCount ................................................. 133
3.2.5.85 proc_ar_GetMergedPropertiesForMetadataObject ....................................... 133
3.2.5.86 proc_ar_GetMethodById ............................................................................. 134
3.2.5.87 proc_ar_GetMethodInstanceById ............................................................... 134
3.2.5.88 proc_ar_GetMethodInstancesForDataClassWithCount .................................. 135
3.2.5.89 proc_ar_GetMethodsForDataClassWithCount ............................................. 135
3.2.5.90 proc_ar_GetModelById .............................................................................. 136
3.2.5.91 proc_ar_GetModelByEntityId ............................................................... 136
3.2.5.92 proc_ar_GetModelsByEntityId .............................................................. 136
3.2.5.93 proc_ar_GetModelsByName ...................................................................... 137
3.2.5.94 proc_ar_GetParameterById ................................................................. 137
3.2.5.95 proc_ar_GetParametersForMethodWithCount ............................................. 138
3.2.5.96 proc_ar_GetPropertiesForMetadataObject ................................................ 138
3.2.5.97 proc_ar_GetRootTypeDescriptorForParameter ........................................... 139
3.2.5.98 proc_ar_GetSafetyNetConfigs .................................................................. 140
3.2.5.99 proc_ar_GetSystemById ........................................................................... 140
3.2.5.100 proc_ar_GetSystemByMethodId ................................................................. 140
3.2.5.101 proc_ar_GetSystemDataBySystemId .......................................................... 141
3.2.5.102 proc_ar_GetSystemForParameterId ........................................................ 141
3.2.5.103 proc_ar_GetSystemForTypeDescriptorId .................................................. 141
3.2.5.104 proc_ar_GetSystemInstanceById ............................................................ 142
3.2.5.105 proc_ar_GetSystemInstancesForSystemWithCount ................................... 142
3.2.5.106 proc_ar_GetSystemsLikeNameWithCount ................................................. 143
3.2.5.107 proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId ............. 143
3.2.5.108 proc_arGetTypeDescriptorById .............................................................. 144
3.2.5.109 proc_arGetTypeDescriptorsByNamespaceAndVersion ....................... 144
3.2.5.110 proc_arGetTypeDescriptorsByNamespaceAndParameter ........................... 144
3.2.5.111 proc_ar_GetViewByMethodInstance ......................................................... 145
3.2.5.112 proc_ar_IsMethodInstantiated ............................................................... 146
3.2.5.113 proc_ar_IsParameterReferencedByMethodInstance .................................. 146
3.2.5.114 proc_ar_RemoveEntity ........................................................................... 147
3.2.5.115 proc_ar_RemoveSafetyNetConfig ............................................................. 148
3.2.5.116 proc_ar_RetrieveProgress ..................................................................... 148
3.2.5.117 proc_ar_SetAccessControlEntryForMetadataObject .............................. 149
3.2.5.118 proc_ar_SetDefaultAction ..................................................................... 149
3.2.5.119 proc_ar_SetDefaultValuesForTypeDescriptor ....................................... 149
3.2.5.120 proc_ar_SetSystemDataBySystemId ...................................................... 151
3.2.5.121 proc_ar_SetSafetyNetConfig ............................................................... 151
3.2.5.122 proc_ar_UpdateActionById ..................................................................... 152
3.2.5.123 proc_ar_UpdateActionParameterById ..................................................... 152
3.2.5.124 proc_ar_UpdateAssociationById ............................................................ 154
3.2.5.125 proc_ar_UpdateAssociationGroupById .................................................. 157
3.2.5.126 proc_ar_UpdateEntityById ..................................................................... 158
1 Introduction

This document specifies the Business Data Connectivity Database Protocol. This protocol enables protocol clients to store and retrieve information about interfaces of line-of-business systems (LOB) systems and annotations of these interfaces.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

access control entry (ACE)
GUID
language code identifier (LCID)

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

AccessChecker
Action
ActionParameter
ActivityTrackingFilter
Association
AssociationGroup
AssociationNavigator
AssociationReference
Associator
BatchingPositionFilter
BatchingTerminationFilter
BinarySecurityDescriptorAccessor
bind
BulkAssociatedIdEnumerator
BulkAssociationNavigator
BulkIdEnumerator
BulkSpecificFinder
Business Logic Module
ChangedIdEnumerator
ComparisonFilter
Creator
DataClass
DefaultValue
DeletedIdEnumerator
Deleter
Disassociator
empty GUID
Entity
EntityInstance
field
FilterDescriptor
Finder
GenericInvoker
Identifier
IdEnumerator
InputFilter
InputOutputFilter
LastIdFilter
LimitFilter
line-of-business (LOB) system
LobSystem
LobSystemInstance
localized name
Metadata partition
metadata store
MetadataCatalog
MetadataModel
MetadataObject
MetadataObjectId
Method
MethodInstance
Model
OutputFilter
PageNumberFilter
Parameter
PasswordCredentialFilter
Property
result set
return code
ReturnTypeDescriptor
root TypeDescriptor
Scalar
security principal
Setting
SpecificFinder
SsoTicketFilter
StreamAccessor
throttle configuration setting
TimeStampFilter
TypeDescriptor
Uniform Resource Locator (URL)
Updater
UserContextFilter
UserCultureFilter
UsernameCredentialFilter
View
Web service
WildcardFilter

The following terms are specific to this document:

Open Data Protocol (OData): A web protocol for querying and updating data specified in [MS-ODATA].

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 Specifications documentation do not include a publishing year because links are to the latest version of the technical 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.


1.2.2 Informative References


[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

1.3 Overview

Enterprises have a variety of data stored in various line-of-business (LOB) systems. Typically, this data is accessible only through the proprietary programming interface of these software systems. It is desirable to be able to provide access to such data via a set of normalized interfaces so that users do not have to learn system-specific programming patterns for each LOB system.

To facilitate this, it is possible to store descriptions of the programmatic interface of the LOB systems using data structures such as Methods, Parameters, and TypeDescriptors, along with information about the LOB systems themselves (such as the server name, connection string and how to authenticate), using data structures such as LobSystem and LobSystemInstance. Methods can be considered to live within an Entity abstraction, representing a business data type, such as "customer" or "order". The LOB system interface definitions can then be transformed into normalized, stereotypical operations against Entities such as "Read-An-Entity-Instance-By-Id", "Read-Entity-Instances", and "Check-Entity-Instance-Permissions" by annotating the actual LOB system interface descriptions, with the annotations described by data structures such as...
**MethodInstance, Identifier, FilterDescriptor, and Association.** These data structures, collectively called *MetadataObjects*, can be grouped into related collections called *MetadataModels* that describe a single LOB system. Once a store of *MetadataModels* is made available, a runtime engine can use this information to convert stereotypical, normalized operations requested by an application that uses the protocol client into LOB system-specific invocations.

This protocol allows a protocol client to create, read, update and delete *MetadataObjects* in a *metadata store*. Additionally, it allows for partitioning of the metadata store such that an application can use the protocol client to store multiple *MetadataModels* that are isolated from *MetadataModels* of the other applications, provided each application is associated with a unique identifier that identifies a *Metadata partition*. Finally, for write operations, the protocol server will provide validation and diagnostic error messages such that protocol clients can maintain the *MetadataObjects* stored on the protocol server in a state that satisfies certain semantic constraints for *MetadataModels*.

1.4 **Relationship to Other Protocols**

The following diagram shows the transport stack that the protocol uses:

![Diagram](image)

Figure 1: This protocol in relation to other protocols

1.5 **Prerequisites/Preconditions**

This protocol operates between a protocol client and a protocol server on which the back-end databases are stored. The protocol client is expected to know the location and connection information for the databases.

This protocol requires that the protocol client has appropriate permissions to call the stored procedures in the back-end databases.

1.6 **Applicability Statement**

There are typically the following two types of applications that can be built using the protocol client, though an application that combines these functions in a single implementation is also feasible:

- **MetadataModel designers**, whose primary purpose is to create or edit a *MetadataModel*. These applications typically offer some graphical design surface and connectivity to LOB systems of known types to enable mining of the LOB system public interface definition and creation of corresponding *MetadataObjects* in the protocol server store.

- **MetadataModel consumers**, whose primary purpose is to read the *MetadataModel* in the protocol server store and use the information therein to convert uniform, stereotypical operations into LOB system-specific interface invocations.
This protocol does not specify how the stored MetadataObjects can be used to do the conversion from a stereotypical client request into a system-specific invocation; it is merely a MetadataObject storage and retrieval protocol.

This protocol is intended for use by protocol clients and protocol servers that are both connected by high-bandwidth, low latency network connections.

1.7 Versioning and Capability Negotiation

Security and authentication methods: This protocol supports the SSPI and SQL Authentication with the Protocol Server role in [MS-TDS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.
2 Messages

2.1 Transport

[MS-TDS] is the transport protocol used to call the stored procedures, return code, and return result sets.

2.2 Common Data Types

The following sections define the common data types that are used in this protocol.

2.2.1 Common Fields

The definitions of some data structures in this section make use of ABNF representation as specified in [RFC5234].

2.2.1.1 Id

Id: int NOT NULL. Identifies a MetadataObject uniquely within a metadata store. The value MUST be a positive integer.

2.2.1.2 Name

Name: nvarchar(255) NOT NULL. The name of a MetadataObject.

2.2.1.3 Namespace

Namespace: nvarchar(255) NOT NULL. The namespace of a DataClass.

2.2.1.4 PartitionId

PartitionId: uniqueidentifier NOT NULL. The identifier for the Metadata partition.

2.2.1.5 IsCached

IsCached: bit NOT NULL. A bit that specifies the frequency of the use of a MetadataObject by the protocol client. Protocol clients can use this as a recommendation as to whether to cache a MetadataObject. Whether the protocol client considers a MetadataObject to be frequently used or not is implementation-specific<1> and is outside the scope of this protocol.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The MetadataObject is infrequently used.</td>
</tr>
<tr>
<td>1</td>
<td>The MetadataObject is frequently used.</td>
</tr>
</tbody>
</table>

2.2.1.6 SettingId

SettingId: nvarchar(128) NULL. The name of the Setting to store a resource (Property, localized name, or access control entry (ACE)) in. If the resource is in the default Setting, the value MUST be NULL.
2.2.1.7 MajorVersion

**MajorVersion**: int NOT NULL. The part of the version of a DataClass tracking the changes done by an application that uses the protocol client. The value MUST be non-negative. If this value is different between any two DataClasses, values of **MinorVersion** (section 2.2.1.8), **BuildVersion** (section 2.2.1.9), and **RevisionVersion** (section 2.2.1.10) MUST be ignored for the purpose of comparison.

2.2.1.8 MinorVersion

**MinorVersion**: int NOT NULL. The part of the version of a DataClass tracking the changes done by an application that uses the protocol client. The value MUST be non-negative. If this value is different between any two DataClasses, values of **BuildVersion** (section 2.2.1.9) and **RevisionVersion** (section 2.2.1.10) MUST be ignored for the purpose of comparison.

2.2.1.9 BuildVersion

**BuildVersion**: int NOT NULL. The part of the version of a DataClass tracking the changes done by an application that uses the protocol client. The value MUST be -1 or non-negative. If this value is different between any two DataClasses, value of **RevisionVersion** (section 2.2.1.10) MUST be ignored for the purpose of comparison. The value -1 indicates the **BuildVersion** is not specified.

2.2.1.10 RevisionVersion

**RevisionVersion**: int NOT NULL. The part of the version of a DataClass tracking the changes done by an application that uses the protocol client. The value MUST be -1 or non-negative. The value -1 indicates the RevisionVersion is not specified. If the value of **BuildVersion** (section 2.2.1.9) is -1, the value of **RevisionVersion** MUST also be -1.

2.2.1.11 EstimatedInstanceCount

**EstimatedInstanceCount**: int NOT NULL. The estimated number of instances of the Entity contained by the line-of-business (LOB) system.

2.2.1.12 IsActive

**IsActive**: bit NOT NULL. A bit that specifies whether a DataClass is active.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The DataClass is not active.</td>
</tr>
<tr>
<td>1</td>
<td>The DataClass is active.</td>
</tr>
</tbody>
</table>

2.2.1.13 CacheUsage

**CacheUsage**: tinyint NOT NULL. The value which suggests how the protocol client creates, reads, updates and deletes EntityInstances against a line-of-business (LOB) system, when the protocol client implementation has provisions for an implementation-specific local cache of EntityInstances. The protocol client implementations MAY ignore this value. The value MUST be listed in the following table.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The protocol client MUST make an implementation-specific choice to use any one of the other behaviors listed in this table based on its capabilities.</td>
</tr>
<tr>
<td>1</td>
<td>The protocol client MUST bypass the EntityInstance data cache for all operations.</td>
</tr>
<tr>
<td>2</td>
<td>The protocol client MUST use the EntityInstance data cache to perform create, update and delete operations. If the requested data is available in the EntityInstance data cache, protocol client MUST use the data in the cache, otherwise the protocol client MUST directly interact with the LOB system to obtain the EntityInstances, and subsequently put the EntityInstances into the cache for future use.</td>
</tr>
<tr>
<td>3</td>
<td>The protocol client MUST use the EntityInstance data cache to perform create, read, update, and delete operations.</td>
</tr>
</tbody>
</table>

2.2.1.14 Position

**Position:** tinyint NOT NULL. The order of an Action among the other Actions for an Entity.<2>

2.2.1.15 IsDisplayed

**IsDisplayed:** bit NOT NULL. A bit that specifies whether an Action is represented in the user interface presented to the user.<3>

2.2.1.16 IsOpenedInNewWindow

**IsOpenedInNewWindow:** bit NOT NULL. A bit that specifies whether the results of executing an Action are presented in a new user interface context.<4>

2.2.1.17 Icon

**Icon:** nvarchar(2080). The implementation-specific location of the resource that is used to represent the Action in the user interface.<5>

2.2.1.18 URL

**URL:** nvarchar(2080) NOT NULL. The implementation-specific parameterized command associated with the Action. The parameters of the command MUST correspond to ActionParameters of this Action.<6>

2.2.1.19 Index

**Index:** tinyint NOT NULL. Index of the ActionParameter. This index corresponds to the parameter in the command of the Action that contains this ActionParameter. The index values of ActionParameters that are contained by the same Action MUST be greater than or equal to 0, and less than the number of ActionParameters that are contained by the Action. The index values of ActionParameters MUST be unique across all ActionParameters that are contained by the same Action.

2.2.1.20 FilterType

**FilterType:** tinyint NOT NULL. Type of the FilterDescriptor. The value MUST be in the following table.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison</td>
<td>1</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a ComparisonFilter.</td>
</tr>
<tr>
<td>LastId</td>
<td>3</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a LastIdFilter.</td>
</tr>
<tr>
<td>Limit</td>
<td>4</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a LimitFilter.</td>
</tr>
<tr>
<td>PageNumber</td>
<td>5</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a PageNumberFilter.</td>
</tr>
<tr>
<td>Password</td>
<td>6</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a PasswordCredentialFilter.</td>
</tr>
<tr>
<td>SsoTicket</td>
<td>8</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a SsoTicketFilter.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>9</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a TimeStampFilter.</td>
</tr>
<tr>
<td>UserContext</td>
<td>10</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a UserContextFilter.</td>
</tr>
<tr>
<td>UserName</td>
<td>11</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a UsernameCredentialFilter.</td>
</tr>
<tr>
<td>WildCard</td>
<td>13</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a WildcardFilter.</td>
</tr>
<tr>
<td>Input</td>
<td>14</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as an InputFilter.</td>
</tr>
<tr>
<td>Output</td>
<td>15</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as an OutputFilter.</td>
</tr>
<tr>
<td>InputOutput</td>
<td>16</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as an InputOutputFilter.</td>
</tr>
<tr>
<td>Batching</td>
<td>17</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a BatchingPositionFilter.</td>
</tr>
<tr>
<td>BatchingTermination</td>
<td>18</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a BatchingTerminationFilter.</td>
</tr>
<tr>
<td>UserCulture</td>
<td>19</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as a UserCultureFilter.</td>
</tr>
<tr>
<td>ActivityId</td>
<td>20</td>
<td>Indicates that the protocol client MUST interpret the FilterDescriptor as an ActivityTrackingFilter.</td>
</tr>
</tbody>
</table>

2.2.1.21  FilterField

**FilterField**: nvarchar(255) NULL. The implementation-specific representation of the field (4) to which the line-of-business (LOB) system applies the semantic represented by this FilterDescriptor. An application utilizing the protocol client typically uses this information to simulate behavior of the LOB system.
2.2.1.22 IdentifierTypeName

IdentifierTypeName: nvarchar(255) NOT NULL. The data type of the value corresponding to the Identifier. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.String</td>
<td>A string of Unicode text.</td>
</tr>
<tr>
<td>System.Int16</td>
<td>A number ranging from negative 32768 to positive 32767.</td>
</tr>
<tr>
<td>System.Int32</td>
<td>A number ranging from negative 2,147,483,648 to positive 2,147,483,647.</td>
</tr>
<tr>
<td>System.UInt16</td>
<td>A number ranging from 0 to 65535.</td>
</tr>
<tr>
<td>System.UInt32</td>
<td>A number ranging from 0 to 4,294,967,295.</td>
</tr>
<tr>
<td>System.UInt64</td>
<td>A number ranging from 0 to 18,446,744,073,709,551,615.</td>
</tr>
<tr>
<td>System.DateTime</td>
<td>A date and time ranging from 12:00:00 midnight, January 1, 1 A.D. (Common Era) to 11:59:59 P.M., December 31, 9999 A.D. (Common Era), in resolution of 100 nanoseconds.</td>
</tr>
<tr>
<td>System.TimeSpan</td>
<td>A duration ranging from negative 10675199 days 2 hours 48 minutes 5 seconds 477 milliseconds 800 nanoseconds to positive 10675199 days 2 hours 48 minutes 5 seconds 477 milliseconds 700 nanoseconds, in resolution of 100 nanoseconds.</td>
</tr>
<tr>
<td>System.Single</td>
<td>A single precision number ranging from negative 3.402823e38 to 3.402823e38.</td>
</tr>
<tr>
<td>System.Double</td>
<td>A double precision number ranging from negative 1.79769313486232e308 to positive 1.79769313486232e308 as well as positive zero, negative zero, positive infinity, negative infinity and NaN.</td>
</tr>
<tr>
<td>System.Decimal</td>
<td>A number ranging from negative 79,228,162,514,264,337,593,543,950,335 to positive 79,228,162,514,264,337,593,543,950,335.</td>
</tr>
<tr>
<td>System.Char</td>
<td>A Unicode character.</td>
</tr>
<tr>
<td>System.Byte</td>
<td>A number ranging from 0 to 255.</td>
</tr>
<tr>
<td>System.SByte</td>
<td>A number ranging from negative 128 to positive 127.</td>
</tr>
<tr>
<td>System.Guid</td>
<td>A <strong>GUID</strong>.</td>
</tr>
<tr>
<td>System.Boolean</td>
<td>A bit.</td>
</tr>
</tbody>
</table>

2.2.1.23 MethodInstanceType

MethodInstanceType: tinyint NOT NULL. Type of the MethodInstance. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finder</td>
<td>1</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a Finder.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SpecificFinder</td>
<td>2</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a SpecificFinder.</td>
</tr>
<tr>
<td>GenericInvoker</td>
<td>4</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a GenericInvoker.</td>
</tr>
<tr>
<td>IdEnumerator</td>
<td>5</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as an IdEnumerator.</td>
</tr>
<tr>
<td>Scalar</td>
<td>6</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a Scalar.</td>
</tr>
<tr>
<td>AccessChecker</td>
<td>7</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as an AccessChecker.</td>
</tr>
<tr>
<td>Creator</td>
<td>8</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a Creator.</td>
</tr>
<tr>
<td>Updater</td>
<td>9</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as an Updater.</td>
</tr>
<tr>
<td>Deleter</td>
<td>10</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a Deleter.</td>
</tr>
<tr>
<td>ChangedIdEnumerator</td>
<td>11</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a ChangedIdEnumerator.</td>
</tr>
<tr>
<td>DeletedIdEnumerator</td>
<td>12</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a DeletedIdEnumerator.</td>
</tr>
<tr>
<td>AssociationNavigator</td>
<td>13</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as an AssociationNavigator.</td>
</tr>
<tr>
<td>Associator</td>
<td>14</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as an Associator.</td>
</tr>
<tr>
<td>Disassociator</td>
<td>15</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a Disassociator.</td>
</tr>
<tr>
<td>StreamAccessor</td>
<td>16</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a StreamAccessor.</td>
</tr>
<tr>
<td>BinarySecurityDescriptorAccessor</td>
<td>17</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a BinarySecurityDescriptorAccessor.</td>
</tr>
<tr>
<td>BulkSpecificFinder</td>
<td>20</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a BulkSpecificFinder.</td>
</tr>
<tr>
<td>BulkAssociatedIdEnumerator</td>
<td>22</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a BulkAssociatedIdEnumerator.</td>
</tr>
<tr>
<td>BulkAssociationNavigator</td>
<td>23</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a BulkAssociationNavigator.</td>
</tr>
<tr>
<td>BulkIdEnumerator</td>
<td>24</td>
<td>Indicates that the protocol client MUST interpret the MethodInstance as a BulkIdEnumerator.</td>
</tr>
</tbody>
</table>
2.2.1.24 Direction

**Direction**: tinyint NOT NULL. The direction of the Parameter while calling the Method that contains the Parameter. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In</td>
<td>1</td>
<td>Used for input purposes only.</td>
</tr>
<tr>
<td>Out</td>
<td>2</td>
<td>Used for output purposes only.</td>
</tr>
<tr>
<td>InOut</td>
<td>3</td>
<td>Used for input purposes before calling the Method and then for reading the output data when the call is complete.</td>
</tr>
<tr>
<td>Return</td>
<td>4</td>
<td>Used to indicate the Parameter is the formal return Parameter.</td>
</tr>
</tbody>
</table>

2.2.1.25 TypeDescriptorTypeName

**TypeDescriptorTypeName**: nvarchar(255) NOT NULL. The implementation-specific identifier of the data type of the data structure that is represented by this TypeDescriptor.

2.2.1.26 TypeDescriptorLobName

**TypeDescriptorLobName**: nvarchar(255) NOT NULL. The line-of-business (LOB) system specified name of the data structure that is represented by the TypeDescriptor. An application that uses the protocol client MUST use this value when manipulating data structures represented by this TypeDescriptor. For example, an LOB system data structure named "CN1A" can be represented by a TypeDescriptor with Name attribute (section 2.2.1.2) equal to "Customer Name", whereas the TypeDescriptorLobName attribute (section 2.2.1.26) of this TypeDescriptor can be "CN1A".

2.2.1.27 TypeDescriptorInterpretation

**TypeDescriptorInterpretation**: nvarchar(512) NULL. Rules to apply to the values in the data structure represented by a TypeDescriptor. If there are no rules to be applied, the value MUST be NULL or empty string (""'). If there are rules to be applied, the value MUST be a rules structure. The following is the ABNF for the rules structure:

```
rules = rule *( %x00 rule)
rule = convertRule / ImplementationSpecificRule

correctRule = %x54 fromType HTAB toType CRLF culture

fromType = TypeName
toType = TypeName
ImplementationSpecificRule = *(%x01-%xFF)
```

**Culture**: A Unicode string representing the implementation-specific name of the culture.

**TypeName**: A Unicode string representing the implementation-specific name of the type.

**ImplementationSpecificRule**: An implementation-specific representation of an implementation-specific rule.
The rules MUST be stored with their order of execution from left to right, where the leftmost rule is first to execute. Occurrence of a `convertRule` indicates that the protocol client and the protocol server MUST interpret this rule as a replacement of the `TypeName` to the name indicated with `toType` to determine the name of the data type represented by the `TypeDescriptor`.

An application that uses the protocol client typically applies all the rules when interacting with the data structures that are returned from or are being prepared to be sent to the LOB system. For the structures that are being prepared to be sent to the LOB system, the rules are applied in reverse order to achieve operational symmetry and compatibility.

### 2.2.1.28 `TypeDescriptorFlags`

`TypeDescriptorFlags`: smallint NOT NULL. The flags for this `TypeDescriptor`. The value MUST consist of zero or more of the bitmask values from the following table.

**Bitmask values:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatorField</td>
<td>0x01</td>
<td>This <code>TypeDescriptor</code> MUST be considered as a field (4) in a Creator view.</td>
</tr>
<tr>
<td>UpdaterField</td>
<td>0x02</td>
<td>This <code>TypeDescriptor</code> MUST be considered as a field (4) in an Updater view.</td>
</tr>
<tr>
<td>PreUpdaterField</td>
<td>0x04</td>
<td>This <code>TypeDescriptor</code> MUST be used to send the latest value received from line-of-business (LOB) system corresponding to the field (4) with the same name as this <code>TypeDescriptor</code> when calling an <code>Updater</code>.</td>
</tr>
<tr>
<td>IsCollection</td>
<td>0x08</td>
<td>This <code>TypeDescriptor</code> MUST be interpreted as a collection of data structures.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>0x10</td>
<td>The protocol client MUST prevent values in the data structures corresponding to this <code>TypeDescriptor</code> from being modified.</td>
</tr>
<tr>
<td>Significant</td>
<td>0x20</td>
<td>The protocol client MUST use the values in the data structures corresponding to this <code>TypeDescriptor</code> when comparing values between structures or creating hash codes for comparison. When this flag is not set, The protocol client MUST ignore the values in the data structures corresponding to this <code>TypeDescriptor</code> when comparing values between structures or creating hash codes for comparison.</td>
</tr>
</tbody>
</table>

### 2.2.1.29 `DefaultValue`

`DefaultValue`: sql_variant NULL. Implementation specific representation of a `DefaultValue`. The applications that use protocol client MUST use this value during initialization of structures corresponding to the `TypeDescriptor`.

### 2.2.1.30 `SystemType`

`SystemType`: tinyint NOT NULL. The type of line-of-business (LOB) system that a `LobSystem` is representing. The value of this field MUST be one of the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>1</td>
<td>The represented LOB system is a database.</td>
</tr>
<tr>
<td>WebService</td>
<td>2</td>
<td>The represented LOB system is a Web service.</td>
</tr>
</tbody>
</table>
| Custom   | 6     | The represented LOB system is a LOB system for which business logic external to
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>the protocol implementation manages the connection and data transfer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wcf</td>
<td>8</td>
<td>The represented LOB system is a service for which the communication address, the bind process, and the contract are specified.</td>
</tr>
<tr>
<td>DotNet</td>
<td>9</td>
<td>The represented LOB system is a Business Logic Module.&lt;7&gt;</td>
</tr>
<tr>
<td>OData</td>
<td>10</td>
<td>The represented LOB system is a OData service that exposes data as per OData protocol.</td>
</tr>
</tbody>
</table>

### 2.2.1.31 SystemData

**SystemData:** image NULL. The implementation-specific representation of the data associated with the LobSystem. This data typically consists of implementation-specific Business Logic Modules.<8>

### 2.2.1.32 MetadataRights

**MetadataRights:** bigint NOT NULL. The permissions available to a security principal (2) to perform operations on or using a MetadataObject. The value MUST be a combination of bits in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>Ability to call implementation-specific logic to execute a MethodInstance.</td>
</tr>
<tr>
<td>0x02</td>
<td>Ability to change the attributes of a MetadataObject or its relationship to other MetadataObjects.</td>
</tr>
<tr>
<td>0x04</td>
<td>Ability to change the permissions associated with a MetadataObject.</td>
</tr>
<tr>
<td>Any other bit</td>
<td>Implementation-specific abilities.</td>
</tr>
</tbody>
</table>

### 2.2.1.33 IsStatic

**IsStatic:** bit NOT NULL. A bit that specifies whether the execution of the Method requires a context of an EntityInstance. The value MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Method operates in the context of a specific EntityInstance.</td>
</tr>
<tr>
<td>1</td>
<td>The Method operates out of the context of a specific EntityInstance.</td>
</tr>
</tbody>
</table>

This value is typically used by applications that use the protocol clients as guidance to enable or disable execution of certain methods based on whether an EntityInstance exists in the context of the application.

### 2.2.1.34 MethodLobName

**MethodLobName:** nvarchar(255) NOT NULL. The name of the line-of-business (LOB) system operation that is represented by this Method. An application that uses the protocol client MUST use this name when calling LOB system operations. For example, an LOB system operation named...
"GetCus_1" can be represented by a Method with Name attribute equal to "Get Customer". The MethodLobName attribute of this Method can be "GetCus_1".

2.2.1.35 IsDefault

IsDefault: bit NOT NULL. A bit that specifies whether a MethodInstance is the default among all MethodInstances sharing its MethodInstanceType within the containing DataClass. The application that uses the protocol client typically uses the default MethodInstance of the specified MethodInstanceType whenever additional specifications are not available. The value MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The MethodInstance is the default one.</td>
</tr>
<tr>
<td>1</td>
<td>The MethodInstance is not the default one.</td>
</tr>
</tbody>
</table>

2.2.1.36 SessionId

SessionId: uniqueidentifier NOT NULL. An identifier to distinguish simultaneous executions of proc_ar_ActivateEntity (section 3.2.5.1), proc_ar_BulkSwitchActive (section 3.2.5.5), and proc_ar_DeactivateEntity (section 3.2.5.26) stored procedures. These stored procedures MUST use this identifier to record their errors to avoid conflicts.

2.2.1.37 IsReverse

IsReverse: bit NOT NULL. A bit that specifies how the Association, referenced by the AssociationReference, is executed. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Association referenced by the AssociationReference requires data structures that correspond to AssociationGroup sources as input and returns a data structure that corresponds to the AssociationGroup's destination.</td>
</tr>
<tr>
<td>1</td>
<td>The Association referenced by the AssociationReference requires a data structure that corresponds to AssociationGroup's destination as input and returns a data structure that corresponds to AssociationGroup's source.</td>
</tr>
</tbody>
</table>

2.2.1.38 ThrottleScope

ThrottleScope: int NOT NULL. A value which specifies the kind of SystemType (section 2.2.1.30) a Throttle Configuration Setting (section 2.2.2.23) is applied against. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The setting is used globally independent from the SystemType of the LobSystem (section 2.2.2.6).</td>
</tr>
<tr>
<td>1</td>
<td>The setting is used for LobSystems that have a SystemType value of &quot;Database&quot;.</td>
</tr>
<tr>
<td>2</td>
<td>The setting is used for LobSystems that have a SystemType value of &quot;WebService&quot;.</td>
</tr>
<tr>
<td>3</td>
<td>The setting is used for LobSystems that have a SystemType value of &quot;Wcf&quot;.</td>
</tr>
</tbody>
</table>
### 2.2.1.39 ThrottleType

**ThrottleType**: int NOT NULL. The type of the Throttle Configuration Setting (section 2.2.2.23) that is used to restrict operations done against the line-of-business (LOB) system. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The setting is not used in any operations. The protocol client MUST ignore settings that have a ThrottleType attribute value of zero.</td>
</tr>
<tr>
<td>1</td>
<td>The setting is used to restrict the number of items retrieved from the LOB system.</td>
</tr>
<tr>
<td>2</td>
<td>The setting is used to restrict the number of bytes of the data retrieved from the LOB system.</td>
</tr>
<tr>
<td>3</td>
<td>The setting is used to restrict the number of simultaneous connections opened against the LOB system at a given time.</td>
</tr>
<tr>
<td>4</td>
<td>The setting is used to restrict the waiting time in milliseconds between the connection attempt to the LOB system and the time the connection is established.</td>
</tr>
</tbody>
</table>

### 2.2.1.40 ThrottleConfigEnabled

**ThrottleConfigEnabled**: bit NOT NULL. A bit that specifies whether a Throttle Configuration Setting (section 2.2.2.23) is enabled. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The setting is not enabled. Protocol client MUST ignore the settings with the Enabled attribute equal to 0.</td>
</tr>
<tr>
<td>1</td>
<td>The setting is enabled.</td>
</tr>
</tbody>
</table>

### 2.2.1.41 ActionParameterName

**ActionParameterName**: nvarchar(4000) NOT NULL. The name of an ActionParameter.

### 2.2.2 Simple Data Types and Enumerations

This section specifies the data structures used in this protocol specification along with their attributes.

#### 2.2.2.1 MetadataObject

This data type corresponds to a MetadataObject. This data type MUST contain all the attributes specified in the following table.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made to it through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
</tbody>
</table>

### 2.2.2.2 Property
This data type corresponds to a Property. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>SQL variant NULL. A value corresponding to the Property.</td>
</tr>
<tr>
<td>Name</td>
<td>nvarchar(255) NOT NULL. Name of the Property.</td>
</tr>
<tr>
<td>SettingId</td>
<td>A setting identifier.</td>
</tr>
</tbody>
</table>

### 2.2.2.3 Localized Name
This data type corresponds to a localized name. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCID</td>
<td>int NOT NULL. A language code identifier (LCID) corresponding to the localized name.</td>
</tr>
<tr>
<td>Value</td>
<td>nvarchar(255) NOT NULL. The localized name.</td>
</tr>
<tr>
<td>SettingId</td>
<td>A setting identifier.</td>
</tr>
</tbody>
</table>

### 2.2.2.4 Access Control Entry
This data type corresponds to an ACE. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rights</td>
<td>A MetadataRights (section 2.2.1.32).</td>
</tr>
<tr>
<td>Identity Name</td>
<td>nvarchar(255) NOT NULL. A name of the security principal (2) associated with this ACE.</td>
</tr>
<tr>
<td>SettingId</td>
<td>A setting identifier.</td>
</tr>
</tbody>
</table>
2.2.2.5 Model

This data type corresponds to a Model. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
</tbody>
</table>

2.2.2.6 LobSystem

This data type corresponds to an LobSystem. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Type</td>
<td>A SystemType (section 2.2.1.30).</td>
</tr>
</tbody>
</table>

2.2.2.7 LobSystemInstance

This data type corresponds to a LobSystemInstance. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
</tbody>
</table>
### 2.2.2.8 DataClass

This data type corresponds to a DataClass. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Version</td>
<td>A value represents the combined values of MajorVersion (section 2.2.1.7), MinorVersion (section 2.2.1.8), BuildVersion (section 2.2.1.9), and RevisionVersion (section 2.2.1.10).</td>
</tr>
<tr>
<td>Namespace</td>
<td>A namespace.</td>
</tr>
</tbody>
</table>

This data type has the states that are specified in the following table.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>This DataClass is available to be used by metadata consumers.</td>
</tr>
<tr>
<td>Not active</td>
<td>This DataClass is available to be used by metadata designers.</td>
</tr>
</tbody>
</table>

### 2.2.2.9 Entity

This data type corresponds to an Entity. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An identifier.</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Version</td>
<td>A value represents the combined values of MajorVersion (section 2.2.1.7), MinorVersion (section 2.2.1.8), BuildVersion (section 2.2.1.9), and RevisionVersion (section 2.2.1.10).</td>
</tr>
<tr>
<td>Namespace</td>
<td>A namespace.</td>
</tr>
<tr>
<td>EstimatedInstanceCount</td>
<td>An estimated instance count.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CacheUsage</td>
<td>A CacheUsage (section 2.2.13).</td>
</tr>
</tbody>
</table>

This data type has the states that are specified in the following table.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>This Entity is available to be used by metadata consumers.</td>
</tr>
<tr>
<td>Not active</td>
<td>This Entity is available to be used by metadata designers.</td>
</tr>
</tbody>
</table>

2.2.2.10 Identifier

This data type corresponds to an Identifier. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (2.2.1.1).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>TypeName</td>
<td>An IdentifierTypeName (section 2.2.1.22).</td>
</tr>
<tr>
<td>OrdinalNumber</td>
<td>An integer representing the index of the Identifiers within the containing Entity.</td>
</tr>
</tbody>
</table>

2.2.2.11 Method

This data type corresponds to a Method. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (2.2.1.1).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>LobName</td>
<td>A MethodLobName (section 2.2.1.34).</td>
</tr>
<tr>
<td>IsStatic</td>
<td>An IsStatic (section 2.2.1.33).</td>
</tr>
</tbody>
</table>
2.2.2.12 MethodInstance

This data type corresponds to a MethodInstance. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (<a href="#">2.2.1.1</a>).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section <a href="#">2.2.1.5</a>).</td>
</tr>
<tr>
<td>ObjectVersion</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Type</td>
<td>A MethodInstanceType (section <a href="#">2.2.1.23</a>).</td>
</tr>
<tr>
<td>IsDefault</td>
<td>An IsDefault (section <a href="#">2.2.1.35</a>).</td>
</tr>
</tbody>
</table>

2.2.2.13 Association

This data type corresponds to an Association. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (<a href="#">2.2.1.1</a>).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section <a href="#">2.2.1.5</a>).</td>
</tr>
<tr>
<td>ObjectVersion</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Type</td>
<td>A MethodInstanceType (section <a href="#">2.2.1.23</a>).</td>
</tr>
<tr>
<td>IsDefault</td>
<td>An IsDefault (section <a href="#">2.2.1.35</a>).</td>
</tr>
</tbody>
</table>

2.2.2.14 Parameter

This data type corresponds to a Parameter. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (<a href="#">2.2.1.1</a>).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section <a href="#">2.2.1.5</a>).</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Object version</strong></td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td><strong>PartitionId</strong></td>
<td>A partition identifier.</td>
</tr>
<tr>
<td><strong>Direction</strong></td>
<td><strong>A Direction</strong> (section 2.2.1.24).</td>
</tr>
<tr>
<td><strong>OrdinalNumber</strong></td>
<td>An integer representing the index of the Parameters within the containing Method.</td>
</tr>
</tbody>
</table>

### 2.2.2.15 TypeDescriptor

This data type corresponds to a TypeDescriptor. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
<td>An <strong>Id</strong> (2.2.1.1).</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>A name.</td>
</tr>
<tr>
<td><strong>IsCached</strong></td>
<td>An <strong>IsCached</strong> (section 2.2.1.5).</td>
</tr>
<tr>
<td><strong>Object version</strong></td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td><strong>PartitionId</strong></td>
<td>A partition identifier.</td>
</tr>
<tr>
<td><strong>TypeName</strong></td>
<td>A <strong>TypeDescriptorTypeName</strong> (section 2.2.1.25).</td>
</tr>
<tr>
<td><strong>LobName</strong></td>
<td>A <strong>TypeDescriptorLobName</strong> (section 2.2.1.26).</td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td>A <strong>TypeDescriptorFlags</strong> (section 2.2.1.28).</td>
</tr>
</tbody>
</table>

### 2.2.2.16 FilterDescriptor

This data type corresponds to a FilterDescriptor. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
<td>An <strong>Id</strong> (2.2.1.1).</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>A name.</td>
</tr>
<tr>
<td><strong>IsCached</strong></td>
<td>An <strong>IsCached</strong> (section 2.2.1.5).</td>
</tr>
<tr>
<td><strong>Object version</strong></td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td><strong>PartitionId</strong></td>
<td>A partition identifier.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>A <strong>FilterType</strong> (section 2.2.1.20).</td>
</tr>
<tr>
<td><strong>Field</strong></td>
<td>A <strong>FilterField</strong> (section 2.2.1.21).</td>
</tr>
</tbody>
</table>
2.2.2.17 DefaultValue

This data type stores a DefaultValue.

2.2.2.18 AssociationGroup

This data type corresponds to an AssociationGroup. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (2.2.1.1).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
</tbody>
</table>

2.2.2.19 AssociationReference

This data type corresponds to an AssociationReference. This data type MUST contain the IsReverse attribute (section 2.2.1.37).

2.2.2.20 Action

This data type corresponds to an Action. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (2.2.1.1).</td>
</tr>
<tr>
<td>Name</td>
<td>A name.</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Position</td>
<td>A Position (section 2.2.1.14).</td>
</tr>
<tr>
<td>IsDisplayed</td>
<td>An IsDisplayed (section 2.2.1.15).</td>
</tr>
<tr>
<td>IsOpenedInNewWindow</td>
<td>An IsOpenedInNewWindow (section 2.2.1.16).</td>
</tr>
<tr>
<td>Icon</td>
<td>An icon.</td>
</tr>
<tr>
<td>URL</td>
<td>A URL.</td>
</tr>
</tbody>
</table>
2.2.2.21 ActionParameter

This data type corresponds to an ActionParameter. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>An Id (^{2.2.1.1}).</td>
</tr>
<tr>
<td>Name</td>
<td>An ActionParameterName (section (^{2.2.1.41})).</td>
</tr>
<tr>
<td>IsCached</td>
<td>An IsCached (section (^{2.2.1.5})).</td>
</tr>
<tr>
<td>Object version</td>
<td>A numerical value representing the version of this data type tracking the changes made through this protocol.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Index</td>
<td>An Index (section (^{2.2.1.19})).</td>
</tr>
</tbody>
</table>

2.2.2.22 Cache Version Stamp

This data type represents the collective version of data structures or relationships of data structures tracking the changes made by the applications utilizing the protocol client. This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>A CacheLine (section (^{2.2.3.1})).</td>
</tr>
<tr>
<td>Version</td>
<td>A numeric value representing the version.</td>
</tr>
<tr>
<td>PartitionId</td>
<td>A partition identifier.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>An implementation-specific timestamp representing the latest time that the Cache Version Stamp was modified.</td>
</tr>
</tbody>
</table>

2.2.2.23 Throttle Configuration Setting

This data type represents a throttle configuration setting.

This data type MUST contain all the attributes specified in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThrottleScope</td>
<td>A ThrottleScope (section (^{2.2.1.38})).</td>
</tr>
<tr>
<td>ThrottleType</td>
<td>A ThrottleType (section (^{2.2.1.39})).</td>
</tr>
<tr>
<td>MaxValue</td>
<td>int NOT NULL. The maximum value permissible for this setting.</td>
</tr>
<tr>
<td>DefaultValue</td>
<td>int NOT NULL. The initial default value for this setting.</td>
</tr>
<tr>
<td>Enabled</td>
<td>A ThrottleConfigEnabled (section (^{2.2.1.40})).</td>
</tr>
<tr>
<td>ProxyId</td>
<td>uniqueidentifier NOT NULL. An implementation-specific non-empty GUID used to partition the set of configuration settings, such that multiple instances of protocol clients may use the same protocol server and have their implementation limited by differing...</td>
</tr>
</tbody>
</table>
amounts. For example, a search crawler crawling an LOB may be allowed to make more simultaneous calls and query larger quantities of data than a web server serving interactive users against the same LOB.

An empty GUID designates a fallback setting. For a given combination of ThrottleScope and ThrottleType, if a setting with a non-empty GUID ProxyId is not available, the fallback setting is used.

2.2.3 Bit Fields and Flag Structures

This section defines common flag structures used by this protocol specification.

2.2.3.1 CacheLine

CacheLine: bigint NOT NULL. A bit field which identifies one or more Cache Version Stamps (section 2.2.2.22). Each bit identifies a Cache Version Stamp corresponding to a data type or relationships between data types. The relationship exists if the data type is contained by, contains or referenced by another data type. The value MUST consist of one or more of the bits from the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00001</td>
<td>LobSystem</td>
</tr>
<tr>
<td>0x00002</td>
<td>LobSystemInstance</td>
</tr>
<tr>
<td>0x00004</td>
<td>DataClass</td>
</tr>
<tr>
<td>0x00008</td>
<td>Entity</td>
</tr>
<tr>
<td>0x00010</td>
<td>Identifier</td>
</tr>
<tr>
<td>0x00020</td>
<td>Method</td>
</tr>
<tr>
<td>0x00040</td>
<td>MethodInstance</td>
</tr>
<tr>
<td>0x00080</td>
<td>FilterDescriptor</td>
</tr>
<tr>
<td>0x00100</td>
<td>Parameter</td>
</tr>
<tr>
<td>0x00200</td>
<td>TypeDescriptor</td>
</tr>
<tr>
<td>0x00400</td>
<td>Action</td>
</tr>
<tr>
<td>0x00800</td>
<td>ActionParameter</td>
</tr>
<tr>
<td>0x01000</td>
<td>Association</td>
</tr>
<tr>
<td>0x08000</td>
<td>AssociationGroup</td>
</tr>
<tr>
<td>0x10000</td>
<td>MetadataCatalog</td>
</tr>
<tr>
<td>0x00010000</td>
<td>Relationship to LobSystem</td>
</tr>
<tr>
<td>0x00020000</td>
<td>Relationship to LobSystemInstance</td>
</tr>
<tr>
<td>0x00040000</td>
<td>Relationship to DataClass</td>
</tr>
</tbody>
</table>
### 2.2.4 Binary Structures

None.

### 2.2.5 Result Sets

This section defines common result sets that are used by this protocol specification.

The definitions of some result sets in this section make use of ABNF representation as specified in [RFC5234].

#### 2.2.5.1 Action Result Set

The **Action** result set contains information about Actions. Each row in the result set MUST contain all the attributes of a single **Action**.

- **Id** int,
- **EntityId** int,
- **Position** tinyint,
- **IsDisplayed** bit,
- **IsOpenedInNewWindow** bit,
- **Icon** nvarchar(2080),
- **Url** nvarchar(2080),
- **Name** nvarchar(255),
- **IsCached** bit,
- **PartitionId** uniqueidentifier,
- **Version** int,

**Id**: The **MetadataObjectId** of the **Action**. The value MUST be an **Id** (section 2.2.1.1).
**EntityId:** The MetadataObjectId of the Entity that contains this Action. The value MUST be an Id.

**Position:** The order of this Action among the other Actions represented in the user interface for this Entity. The value MUST be a Position (section 2.2.1.14).

**IsDisplayed:** A bit that provides a hint on whether this Action is represented in the user interface presented to the user. The value MUST be an IsDisplayed section 2.2.1.15).

**IsOpenedInNewWindow:** A bit that provides a hint on whether the results of executing this Action are represented in a new user interface context in the user interface presented to the user. The value MUST be an IsOpenedInNewWindow (section 2.2.1.16).

**Icon:** The URL of the resource associated with the Action. The value MUST be an Icon (section 2.2.1.17).

**Url:** The URL associated with the Action. The value MUST be a URL (section 2.2.1.18).

**Name:** The name of the Action. The value MUST be a Name (section 2.2.1.2).

**IsCached:** A bit that specifies whether the Action is frequently used. The value MUST be an IsCached (section 2.2.1.5).

**PartitionId:** The Metadata partition of the Action. The value MUST be a PartitionId (section 2.2.1.4).

**Version:** The object version of this Action.

### 2.2.5.2 Count Result Set

The Count result set contains the number of rows that satisfy the requested condition. If the stored procedure that returned this result set immediately returns another result set, data in the Count result set MUST be equal to number of rows returned in the following result set. This result set MUST have exactly one row.

```
UnnamedColumn0 int,
```

**UnnamedColumn0:** The number of rows that satisfy the requested condition.

### 2.2.5.3 MetadataCatalog Result Set

The MetadataCatalog result set contains data about a single MetadataCatalog. The result set MUST contain zero or one row.

```
Id int,
PartitionId uniqueidentifier,
Name nvarchar(255),
IsCached bit,
Version int,
```

**Id:** The MetadataObjectId of the MetadataCatalog. The value MUST be "Id" (2.2.1.1).

**PartitionId:** Metadata partition of the MetadataCatalog. The value MUST be a "PartitionId" (section 2.2.1.4).

**Name:** The name of the MetadataCatalog. The value MUST be "Name" (section 2.2.1.2).
IsCached: The bit that specifies whether the MetadataCatalog is frequently used. The value MUST be "IsCached" (section 2.2.1.5).

Version: The object version of this MetadataCatalog.

2.2.5.4 LocalizedName Result Set

The Localized Name result set contains information about localized names. Each row in the result set contains a single localized name of a MetadataObject in a specific locale and Setting.

Id int,
LCID int,
LocalizedName nvarchar(255),
MetadataObjectId int,
SettingId nvarchar(128),

Id: An implementation-specific identifier for the localized name.

LCID: The LCID corresponding to the localized name.

LocalizedString: The localized name of the specified MetadataObject corresponding to the LCID.

MetadataObjectId: The MetadataObjectId of the MetadataObject containing the localized name. The value MUST be an Id (2.2.1.1).

SettingId: The Setting of the localized name. The value MUST be a SettingId (section 2.2.1.6).

2.2.5.5 Partition Result Set

The Partition Result Set contains information about Metadata partitions of the metadata store. Each row of the result set identifies a single Metadata partition.

PartitionId uniqueidentifier,

PartitionId: The identifier of the Metadata partition. The value MUST be a PartitionId (section 2.2.1.4).

2.2.5.6 Setting Result Set

The Setting result set contains information about Settings. Each row in the result set identifies a single Setting.

SettingId nvarchar(128),

SettingId: The name of the Setting. The value MUST be a SettingId (section 2.2.1.6).

2.2.5.7 Association Result Set

The Association result set contains information about Associations. Each row in the result set contains all the attributes of a single Association.

Id int,
AssociationGroupId int,
Id: The MetadataObjectId of the Association. The value MUST be an Id (section 2.2.1.1).

AssociationGroupId: The MetadataObjectId of the AssociationGroup that contains the Association. If the DataClass that contains the Association is an active DataClass or the Association is referenced from an AssociationReference contained by an AssociationGroup which also is contained by an active Entity then the value MUST be an Id. Otherwise the value MUST be NULL or 0. The protocol client MUST NOT distinguish between the values NULL and 0.

MethodId: The MetadataObjectId of the Method that contains this Association. The value MUST be an Id.

ReturnTypeDescriptorId: The MetadataObjectId of the ReturnTypeDescriptor. If the Association has a ReturnTypeDescriptor the value MUST be an Id. Otherwise the value MUST be NULL or 0. The protocol client MUST NOT distinguish between the values NULL and 0.

Type: The type of the MethodInstance. The value MUST be a MethodInstanceType (section 2.2.1.23).

IsDefault: A bit that specifies if the Association is default. The value MUST be an IsDefault (section 2.2.1.35).

Name: The name of the Association. The value MUST be a Name (section 2.2.1.2).

IsCached: A bit that specifies whether the Association is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The metadata partition of the Association. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of this Association.

2.2.5.8 Association Group Result Set

The Association Group result set contains information about AssociationGroups (section 2.2.18). Each row in the result contains all the attributes of a single AssociationGroup.

Id int,
EntityId int,
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,

Id: The MetadataObjectId of the AssociationGroup. The value MUST be an Id (section 2.2.1.1).
**EntityId:** The **MetadataObjectId** of the **Entity** that contains the **AssociationGroup**. The value MUST be an **Id**.

**Name:** The name of the **AssociationGroup**. The value MUST be a **Name** (section 2.2.1.2).

**IsCached:** A bit that specifies whether the **AssociationGroup** is frequently used. The value MUST be an **IsCached** (section 2.2.1.5).

**PartitionId:** The metadata partition of the **AssociationGroup**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Version:** The object version of the **AssociationGroup**.

### 2.2.5.9 Association Member Result Set

The **Association Member** result set contains information about Association sources or destination of an **Association**. Each row in the result set contains attributes to identify a single **Entity**.

```
EntityId int,
_EntityName nvarchar(255),
_EntityNamespace nvarchar(255),
PartitionId uniqueidentifier,
```

**EntityId:** The MetadataObjectId of the **Entity**. If the **Entity** is active, the value MUST be an **Id** (section 2.2.1.1). Otherwise, the value MUST be 0 or NULL. The protocol client MUST NOT distinguish between the values NULL and 0.

**_EntityName:** The name of the **Entity**. If the **Entity** is not active, the value MUST be a **Name** (section 2.2.1.2). Otherwise the value MUST be NULL.

**_EntityNamespace:** The namespace of the **Entity**. If the **Entity** is not active, the value MUST be a **Namespace** (section 2.2.1.3). Otherwise the value MUST be NULL.

**PartitionId:** The Metadata partition of the **Entity**. The value MUST be a **PartitionId** (section 2.2.1.4).

### 2.2.5.10 AssociationReference Result Set

The **Association Reference** result set contains information about AssociationReferences (section 2.2.2.19) contained by an AssociationGroup (section 2.2.2.18). Each row in the result set contains attributes for a single **AssociationReference**.

```
Id int,
AssociationGroupId int,
AssociationId int,
_AssociationName nvarchar(255),
_AssociationEntityName nvarchar(255),
_AssociationEntityNamespace nvarchar(255),
IsReverse bit,
Version int,
PartitionId uniqueidentifier,
```

**Id:** An implementation-specific identifier for the **AssociationReference**.
**AssociationGroupId:** The MetadataObjectId of the **AssociationGroup** that contains the **AssociationReference**. The value MUST be an **Id** (section 2.2.1.1).

**AssociationId:** The MetadataObjectId of the Association the **AssociationReference** references to. If this **AssociationReference** refers to an **Association** contained by an active DataClass, the value MUST be an **Id**. Otherwise, the value MUST be NULL or 0. The protocol client MUST NOT distinguish between the values NULL and 0.

**_AssociationName:** The name of the **Association** that the **AssociationReference** references. The value MUST be a **Name** (section 2.2.1.2).

**_AssociationEntityName:** The name of the Entity that contains the **Association** referenced by the **AssociationReference**. The value MUST be a **Name**.

**_AssociationEntityNamespace:** The namespace of the Entity that contains the **Association** referenced by the **AssociationReference**. The value MUST be a **Namespace** (section 2.2.1.3).

**IsReverse:** The "IsReverse" attribute of the **AssociationReference**. Value MUST be an **IsReverse** (section 2.2.1.37).

**Version:** The object version of the **AssociationGroup** that contains the **AssociationReference**.

**PartitionId:** Metadata partition of the **AssociationGroup** that contains the **AssociationReference**. The value MUST be **PartitionId** (section 2.2.1.4).

### 2.2.5.11  Cache Version Stamps Result Set

The **Cache Version Stamps** result set returns information about the Cache Version Stamps (section 2.2.2.22). Each row in the result set represents a single Cache Version Stamp. The result set MUST be sorted by ascending order of value of the **PartitionId** column.

```sql
CacheLine bigint,
Counter int,
PartitionId uniqueidentifier,
LastModified bigint,
```

**CacheLine:** Identifier for the Cache Version Stamp. The value MUST be a **CacheLine** (section 2.2.3.1). This value MUST have only one bit set.

**Counter:** The value of the **Version** attribute of the Cache Version Stamp.

**PartitionId:** The Metadata partition of the Cache Version Stamp. The value MUST be a **PartitionId** (section 2.2.1.4).

**LastModified:** The value of the **Timestamp** attribute of the Cache Version Stamp.

### 2.2.5.12  TypeDescriptor Result Set

The **TypeDescriptor** result set contains information about TypeDescriptors. Each row in the result set MUST contain all the attributes of a single **TypeDescriptor**.

```sql
Id int,
ParameterId int,
ParentTypeDescriptorId int,
TypeName nvarchar(255),
Rules nvarchar(512),
```

---

[MS-BDCDPS2] — v20120630
*Business Data Connectivity Database Version 2 Protocol Specification*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 16, 2012*
ChildrenContainRules bit,
ContainsIdentifier bit,
IdentifierId int,
ContainsFilterDescriptor bit,
FilterDescriptorId int,
ContainsReadOnly bit,
Flags smallint,
LobName nvarchar(255),
AssociationId int,
_identifierName nvarchar(255),
_identifierEntityName nvarchar(255),
_identifierEntityNamespace nvarchar(255),
_associationName nvarchar(255),
_associationEntityName nvarchar(255),
_associationEntityNamespace nvarchar(255),
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,

Id: The MetadataObjectId of the TypeDescriptor. The value MUST be an Id (section 2.2.1.1).

ParameterId: The MetadataObjectId of the Parameter that contains the TypeDescriptor. The value MUST be an Id.

ParentTypeDescriptorId: The MetadataObjectId of the parent TypeDescriptor that contains the TypeDescriptor. If the TypeDescriptor is a root TypeDescriptor, the value MUST be NULL. Otherwise, the value MUST be an Id.

TypeName: The name of the data type that is represented by the TypeDescriptor. The value MUST be a TypeDescriptorTypeName (section 2.2.1.25).

Rules: The rules for the TypeDescriptor. The value MUST be a TypeDescriptorInterpretation (section 2.2.1.27).

ChildrenContainRules: A bit that specifies whether any descendant of the TypeDescriptor has rules. The value MUST be 1, if any descendant of the TypeDescriptor has TypeDescriptorInterpretation attribute as not NULL, otherwise the value MUST be 0.

ContainsIdentifier: A bit that specifies whether this or any descendant of this TypeDescriptor references an Identifier. The value MUST be 1, if this TypeDescriptor references an Identifier or there is a descendant of this TypeDescriptor which references an Identifier; otherwise, the value MUST be 0.

IdentifierId: The MetadataObjectId of the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an active Entity, the value MUST be an Id. Otherwise, the value MUST be NULL or 0. The protocol client MUST NOT distinguish between the values NULL and 0.

ContainsFilterDescriptor: A bit that specifies whether this or any descendant of this TypeDescriptor has an associated FilterDescriptor. The value MUST be 1, if this TypeDescriptor has an associated FilterDescriptor or there is a descendant of this TypeDescriptor which has an associated FilterDescriptor, otherwise the value MUST be 0.

FilterDescriptorId: The MetadataObjectId of the FilterDescriptor associated with the TypeDescriptor. If a FilterDescriptor is associated with this TypeDescriptor, the value MUST be an Id. Otherwise, the value MUST be NULL.
ContainsReadOnly: A bit that specifies whether this or any descendant of this TypeDescriptor has ReadOnly flag set. The value MUST be 1, if this TypeDescriptor has ReadOnly flag set or there is a descendant of this TypeDescriptor which has ReadOnly flag set. Otherwise, the value MUST be 0.

Flags: The flags of the TypeDescriptor. The value MUST be a TypeDescriptorFlags (section 2.2.1.28).

LobName: The name of the data structure that is represented by the TypeDescriptor. The value MUST be a TypeDescriptorLobName (section 2.2.1.26).

AssociationId: The MetadataObjectId of the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association defined on an active DataClass, the value MUST be an Id. Otherwise, the value MUST be NULL or 0. The protocol client MUST NOT distinguish between the values NULL and 0.

_identifierName: The name of the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Name (section 2.2.1.2). Otherwise, the value MUST be NULL.

_identifierEntityName: The name of the Entity that contains the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Name. Otherwise it MUST be NULL.

_identifierEntityNamespace: The namespace of the Entity that contains the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Namespace (section 2.2.1.3). Otherwise, it MUST be NULL.

_AssociationName: The name of the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Name. Otherwise, the value MUST be NULL.

_AssociationEntityName: The name of the Entity that contains the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Name. Otherwise, the value MUST be NULL.

_AssociationEntityNamespace: The namespace of the Entity that contains the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Namespace. Otherwise, the value MUST be NULL.

Name: The name of the Namespace. The value MUST be a Name.

IsCached: A bit that specifies whether the TypeDescriptor is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The metadata partition of the TypeDescriptor. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of the TypeDescriptor.

2.2.5.13 DataClass Result Set

The DataClass result set contains information about DataClasses. Each row in the result set contains all the attributes of a single DataClass.

Id int,
SystemId int,
Name nvarchar(255),
Namespace nvarchar(255),
MajorVersion int,
MinorVersion int,
BuildVersion int,
RevisionVersion int,
Active bit,
IsCached bit,
PartitionId uniqueidentifier,
Version int,

Id: The MetadataObjectId of the DataClass. The value MUST be an Id (section 2.2.1.1).

SystemId: The MetadataObjectId of the LobSystem which contains the DataClass. The value MUST be an Id.

Name: The name of the DataClass. The value MUST be a Name (section 2.2.1.2).

Namespace: The namespace of the DataClass. The value MUST be a Namespace (section 2.2.1.3).

MajorVersion: The major version of the DataClass. The value MUST be a MajorVersion (section 2.2.1.7).

MinorVersion: The minor version of the DataClass. The value MUST be a MinorVersion (section 2.2.1.8).

BuildVersion: The build version of the DataClass. The value MUST be a BuildVersion (section 2.2.1.9).

RevisionVersion: The revision version of the DataClass. The value MUST be a RevisionVersion (section 2.2.1.10).

Active: A bit that specifies whether the returned version of the DataClass is active. The value MUST be an IsActive (section 2.2.1.12).

IsCached: A bit that specifies whether the DataClass is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The metadata partition of the DataClass. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of this DataClass.

2.2.5.14 DefaultValues Result Set

The DefaultValues result set contains information about DefaultValues (section 2.2.2.17). Each row of the result set contains information about a single DefaultValues.

Id int,
Value sql_variant,
TypeDescriptorId int,
MethodInstanceId int,
MethodInstanceName nvarchar(255),

Id: An implementation-specific identifier for the DefaultValues.
Value: The DefaultValues.

**TypeDescriptorId:** The MetadataObjectId of the TypeDescriptor with which the DefaultValues is associated. The value MUST be an Id (section 2.2.1.1).

**MethodInstanceId:** The MetadataObjectId of the MethodInstance with which the DefaultValues is associated. The value MUST be an Id.

**MethodInstanceName:** The name of the MethodInstance with which the DefaultValues is associated. The value MUST be a Name (section 2.2.1.2).

### 2.2.5.15 Entity Result Set

The Entity result set contains information about Entities. Each row in the result set contains all the attributes of a single Entity.

```plaintext
Id int,
EstimatedInstanceCount int,
CacheUsage int,
SystemId int,
MajorVersion int,
MinorVersion int,
BuildVersion int,
RevisionVersion int,
Namespace nvarchar(255),
Active bit,
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,
```

**Id:** The MetadataObjectId of the Entity. The value MUST be an Id (section 2.2.1.1).

**EstimatedInstanceCount:** The maximum estimated number of instances of the Entity. The value MUST be an EstimatedInstanceCount (section 2.2.1.11).

**CacheUsage:** The CacheUsage attribute of the Entity. The value must be a CacheUsage (section 2.2.1.13).

**SystemId:** The MetadataObjectId of the LobSystem that contains the Entity. The value MUST be an Id.

**MajorVersion:** The major version of the Entity. The value MUST be a MajorVersion (section 2.2.1.7).

**MinorVersion:** The minor version of the Entity. The value MUST be a MinorVersion (section 2.2.1.8).

**BuildVersion:** The build version of the Entity. The value MUST be a BuildVersion (section 2.2.1.9).

**RevisionVersion:** The revision version of the Entity. The value MUST be a RevisionVersion (section 2.2.1.10).

**Namespace:** The namespace of the Entity. The value MUST be a Namespace (section 2.2.1.3).
Active: A bit that specifies whether the returned version of this Entity is active. The value MUST be an IsActive (section 2.2.1.12).

Name: The name of this Entity. The value MUST be a Name (section 2.2.1.2).

IsCached: A bit that specifies whether the Entity is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The Metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of this Entity.

2.2.5.16 Entity Name Result Set

The Entity Name result set contains information about Entities. Each row in the result set contains the "Name" and "Namespace" attributes of a single Entity.

Namespace nvarchar(255),
Name nvarchar(255),

Namespace: The namespace of the Entity. The value MUST be a Namespace (section 2.2.1.3).
Name: The name of the Entity. The value MUST be a Name (section 2.2.1.2).

2.2.5.17 FilterDescriptor Result Set

The FilterDescriptor result set contains information about FilterDescriptors (section 2.2.2.16). Each row in the result set contains all the attributes of a single FilterDescriptor.

Id int,
FilterType tinyint,
MethodId int,
FilterField nvarchar(255),
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,

Id: The MetadataObjectId of the FilterDescriptor. The value MUST be an Id (section 2.2.1.1).
FilterType: The type of the FilterDescriptor. The value MUST be a FilterType (section 2.2.1.20).
MethodId: The MetadataObjectId of the Method that contains this FilterDescriptor. The value MUST be an Id.
FilterField: The Field attribute of the FilterDescriptor. The value MUST be a FilterField (section 2.2.1.21).
Name: The name of this FilterDescriptor. The value MUST be a Name (section 2.2.1.2).
IsCached: A bit that specifies whether the FilterDescriptor is frequently used. The value MUST be an IsCached (section 2.2.1.5).
**PartitionId**: The metadata partition of the **FilterDescriptor**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Version**: The object version of this **FilterDescriptor**.

### 2.2.5.18 Identifier Result Set

The **Identifier** result set contains information about Identifier. Each row in the result set contains all the attributes of a single **Identifier**. The result set MUST be sorted by ascending order of value of the **OrdinalNumber** column.

```
Id int,
TypeName nvarchar(255),
EntityId int,
OrdinalNumber tinyint,
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,
```

**Id**: The MetadataObjectId of the **Identifier**. The value MUST be an **Id** (section 2.2.1.1).

**TypeName**: The data type of the value corresponding to the **Identifier**. The value MUST be an **IdentifierTypeName** (section 2.2.1.22).

**EntityId**: The MetadataObjectId of the Entity that contains the **Identifier**. The value MUST be an **Id**.

**OrdinalNumber**: The "OrdinalNumber" attribute of the **Identifier**.

**Name**: The name of the **Identifier**. The value MUST be a **Name** (section 2.2.1.2).

**IsCached**: A bit that specifies whether the **Identifier** is frequently used. The value MUST be an **IsCached** (section 2.2.1.5).

**PartitionId**: The Metadata partition of the **Identifier**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Version**: The object version of this **Identifier**.

### 2.2.5.19 Property Result Set

The **Property** result set contains the name and value of the Property associated with a MetadataObject. Each row represents one **Property**.

```
Name nvarchar(255),
Value sql_variant,
SettingId nvarchar(128),
```

**Name**: The name of the **Property**.

**Value**: The implementation-specific representation of the value of the **Property**.

**SettingId**: The Setting that contains the **Property**. The value MUST be a **SettingId** (section 2.2.1.6).
2.2.5.20 Method Result Set

The Method result set contains information about Methods. Each row in the result set contains all the attributes of a single Method.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id int</td>
<td>The MetadataObjectId of the Method. The value MUST be an Id (section 2.2.1.1)</td>
</tr>
<tr>
<td>ClassId int</td>
<td>The MetadataObjectId of the DataClass of the Method. The value MUST be an Id.</td>
</tr>
<tr>
<td>IsStatic bit</td>
<td>A bit that specifies whether the Method is associated with an EntityInstance. The value MUST be an IsStatic (section 2.2.1.33).</td>
</tr>
<tr>
<td>LobName nvarchar(255)</td>
<td>The name of the operation on the line-of-business (LOB) system that the Method corresponds to. The value MUST be a MethodLobName (section 2.2.1.34).</td>
</tr>
<tr>
<td>Name nvarchar(255)</td>
<td>The name of the Method. The value MUST be a Name (section 2.2.1.2).</td>
</tr>
<tr>
<td>IsCached bit</td>
<td>A bit that specifies whether this Method is frequently used. The value MUST be an IsCached (section 2.2.1.5).</td>
</tr>
<tr>
<td>PartitionId uniqueidentifier</td>
<td>The Metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).</td>
</tr>
<tr>
<td>Version int</td>
<td>The object version this Method.</td>
</tr>
</tbody>
</table>

2.2.5.21 MethodInstance Result Set

The MethodInstance result set contains information about MethodInstances. Each row in the result set contains all the attributes of a single MethodInstance.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id int</td>
<td>The MetadataObjectId of the MethodInstance. The value MUST be an Id (section 2.2.1.1).</td>
</tr>
<tr>
<td>MethodId int</td>
<td>The MetadataObjectId of the Method that contains the MethodInstance. The value MUST be an Id.</td>
</tr>
<tr>
<td>ReturnTypeDescriptorId int</td>
<td></td>
</tr>
<tr>
<td>Type tinyint</td>
<td></td>
</tr>
<tr>
<td>IsDefault bit</td>
<td></td>
</tr>
<tr>
<td>Name nvarchar(255)</td>
<td></td>
</tr>
<tr>
<td>IsCached bit</td>
<td></td>
</tr>
<tr>
<td>PartitionId uniqueidentifier</td>
<td></td>
</tr>
<tr>
<td>Version int</td>
<td></td>
</tr>
</tbody>
</table>
ReturnTypeDescriptorId: The MetadataObjectId of the ReturnTypeDescriptor. If the MethodInstance has a ReturnTypeDescriptor, the value MUST be an Id. Otherwise the value MUST be NULL or 0. The protocol client MUST NOT distinguish between the values NULL and 0.

Type: The type of the MethodInstance. The value MUST be a MethodInstanceType (section 2.2.1.23).

IsDefault: A bit that specifies whether the MethodInstance is a default one. The value MUST be an IsDefault (section 2.2.1.35).

Name: The name of the MethodInstance. The value MUST be a Name (section 2.2.1.2).

IsCached: A bit that specifies whether the MethodInstance is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The metadata partition of the MethodInstance. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of this MethodInstance.

2.2.5.22 Model Result Set

The Model result set contains information about Models. Each row in the result set contains all the attributes of a single Model.

Id int, 
Name nvarchar(255), 
IsCached bit, 
PartitionId uniqueidentifier, 
Version int,

Id: The MetadataObjectId of the Model. The value MUST be an Id (section 2.2.1.1).

Name: The name of the Model. The value MUST be a Name (section 2.2.1.2).

IsCached: A bit that specifies whether the Model is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The Metadata partition of the Model. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of the Model.

2.2.5.23 Parameter Result Set

The Parameter result set contains information about Parameters. Each row in the result set contains all the attributes of a single Parameter. The result set MUST be sorted by ascending order of value of the OrdinalNumber column.

Id int, 
MethodId int, 
Direction tinyint, 
OrdinalNumber tinyint, 
Name nvarchar(255), 
IsCached bit, 
PartitionId uniqueidentifier,
Id: The MetadataObjectId of the Parameter. The value MUST be an Id (section 2.2.1.1).

MethodId: The MetadataObjectId of the Method that contains the Parameter. The value MUST be an Id.

Direction: The direction of the Parameter while calling its containing Method. The value MUST be a Direction (section 2.2.1.24).

OrdinalNumber: The OrdinalNumber attribute of the Parameter.

Name: The name of the Parameter. The value MUST be a Name (section 2.2.1.2).

IsCached: A bit that specifies whether the Parameter is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The Metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of the Parameter.

RootTypeDescriptorId: The root TypeDescriptor associated with the Parameter. The value MUST be an Id.

2.2.5.24 Throttle Setting Result Set

The Throttle Setting result set contains information about Throttle Configuration Settings (section 2.2.2.23). Each row in the result set contains attributes for a single setting.

Id int,
ThrottleScope int,
ThrottleType int,
Max int,
Default int,
Enabled bit,
ProxyId uniqueidentifier,

Id: An implementation-specific identifier for the setting.

ThrottleScope: The scope of this setting. Value MUST be a ThrottleScope (section 2.2.1.38).

ThrottleType: The type of this setting. Value MUST be ThrottleType (section 2.2.1.39).

Max: The maximum level to which this setting can be increased.

Default: The default level of this setting.

Enabled: A bit that specifies whether this setting is enabled. The value MUST be a ThrottleConfigEnabled (section 2.2.1.40).

ProxyId: An implementation-specific value specified in the Throttle Configuration Setting.
2.2.5.25 System Result Set

The **System** result set contains information about LobSystems. Each row in the result set contains all the attributes of a single **LobSystem**.

```
Id int,
SystemType tinyint,
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,
```

**Id**: The MetadataObjectId of the **LobSystem**. The value MUST be an **Id** (section 2.2.1.1).

**SystemType**: The type of the **LobSystem**. The value MUST be a **SystemType** (section 2.2.1.30).

**Name**: The name of the **LobSystem**. The value MUST be a **Name** (section 2.2.1.2).

**IsCached**: A bit that specifies whether the **LobSystem** is frequently used. The value MUST be an **IsCached** (section 2.2.1.5).

**PartitionId**: The Metadata partition of the **LobSystem**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Version**: The object version of the **LobSystem**.

2.2.5.26 System Data Result Set

The **System Data** result set contains the information about **SystemData** (section 2.2.1.30) associated with a single LobSystem. The result set MUST contain zero or one row.

```
Length int,
Data image,
```

**Length**: The size of the **SystemData**, in bytes.

**Data**: The **SystemData** associated with the **LobSystem**.

2.2.5.27 SystemInstance Result Set

The **SystemInstance** result set contains information about LobSystemInstances (section 2.2.2.7). Each row in the result set contains all the attributes of a single **LobSystemInstance**.

```
Id int,
SystemId int,
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,
```

**Id**: The MetadataObjectId of the **LobSystemInstance**. The value MUST be an **Id** (section 2.2.1.1).

**SystemId**: The MetadataObjectId of the LobSystem which contains this **LobSystemInstance**. The value MUST be an **Id**.
**Name:** The name of the LobSystemInstance. The value MUST be a Name (section 2.2.1.2).

**IsCached:** A bit that specifies whether the LobSystemInstance is frequently used. The value MUST be an IsCached (section 2.2.1.5).

**PartitionId:** The Metadata partition of the LobSystemInstance. The value MUST be a PartitionId (section 2.2.1.4).

**Version:** The object version of the LobSystemInstance.

### 2.2.5.28 Access Control Entry Result Set

The Access Control Entry result set contains information about ACEs. Each row in the result set contains all the attributes of a single ACE.

```plaintext
MetadataObjectId int,
IdentityName nvarchar(255),
DisplayName nvarchar(255),
RawSid varbinary(512),
Rights bigint,
```

**MetadataObjectId:** The MetadataObjectId of the MetadataObject that the ACE is associated with.

**IdentityName:** The name of the security principal (2) associated with the ACE.

**DisplayName:** The name of the security principal (2) associated with the ACE. The applications that use the protocol client typically use this value to represent the security principal (2) in the user interface.

**RawSid:** This column value MUST be NULL and MUST be ignored by the protocol client.

**Rights:** The permissions available to the security principal (2) for the specified MetadataObject. It MUST be MetadataRights (section 2.2.1.32).

### 2.2.5.29 Id Result Set

The Id result set contains MetadataObjectIds. Each row in the result set contains a single MetadataObjectId.

```plaintext
Id int,
```

**Id:** The MetadataObjectId. The value MUST be an Id (section 2.2.1.1).

### 2.2.5.30 Progress Result Set

The Progress result set contains information about the finished fraction of an operation that is tracked by proc_ar_UpdateProgress (section 3.2.5.133) and proc_ar_RetrieveProgress (section 3.2.5.116) stored procedures.

```plaintext
Progress System.Single,
```

**Progress:** Indicates the fraction of the portion of the operation that is complete. The value MUST be between 0 and 1.
### 2.2.5.31 Activation Errors Result Set

The **Activation Errors** result set contains information about reference errors encountered during the process of marking one or more Entities as active. The **ErrorCode** value specifies the list of possible reference errors.

```plaintext
Id int,
SessionId uniqueidentifier,
ErrorCode int,
ContainingEntityNamespace nvarchar(255),
ContainingEntityName nvarchar(255),
ContainingEntityVersion nvarchar(255),
ContainingMethodName nvarchar(255),
ContainingParameterName nvarchar(255),
ContainingTypeDescriptorName nvarchar(255),
ContainingTypeDescriptorId int,
ContainingAssociationGroupName nvarchar(255),
TDIDReferenceName nvarchar(255),
TDIDReferenceTypeName nvarchar(255),
TDIDEntityReferenceName nvarchar(255),
TDIDEntityReferenceNamespace nvarchar(255),
TDAssociationReferenceName nvarchar(255),
TDAssociationEntityReferenceName nvarchar(255),
TDAssociationEntityReferenceNamespace nvarchar(255),
AGAssociationReferenceName nvarchar(255),
AGAssociationEntityReferenceName nvarchar(255),
AGAssociationEntityReferenceNamespace nvarchar(255),
```

- **Id**: Unique identifier of the error.
- **SessionId**: Session of the activation or deactivation. The value MUST be a "SessionId" (section 2.2.1.36).
- **ErrorCode**: The error code. This value MUST be in the following table.

Possible parameter values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1003</td>
<td>A TypeDescriptor is in error because it references an Identifier that doesn't exist in the specified Entity. For this error code, all the following MUST NOT be NULL: ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, ContainingMethodName, ContainingParameterName, ContainingTypeDescriptorName, ContainingTypeDescriptorId, TDIDReferenceName, TDIDReferenceTypeName, TDIDEntityReferenceName, and TDIDEntityReferenceNamespace. All other columns MUST be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>1004</td>
<td>A TypeDescriptor is in error because it references an Association that does not exist in the specified Entity. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, ContainingMethodName, ContainingParameterName, ContainingTypeDescriptorName, ContainingTypeDescriptorId, TDAssociationReferenceName, TDAssociationEntityReferenceName, and TDAssociationEntityReferenceNamespace must NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>1005</td>
<td>An Entity is in error because the TypeDescriptors that are contained in the Parameters of its Methods are referencing only non-empty strict subset of Identifiers of an active Entity. For this error code, ContainingEntityNamespace, ContainingEntityName,</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-1008</td>
<td>An AssociationReference is in error because it references an Association that doesn’t exist in the specified Entity. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, ContainingAssociationGroupName, AGAssociationReferenceName, AGAssociationEntityReferenceNamespace, and AGAssociationEntityReferenceNamespace all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>-800</td>
<td>An AssociationGroup is in error because all Associations referenced by the AssociationReferences with IsReverse (section 2.2.1.37) attribute set to 0 of this AssociationGroup do not have same sources. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, and ContainingAssociationGroupName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>-801</td>
<td>An AssociationGroup is in error because one of the following conditions is true: The Entity containing AssociationGroup is not the AssociationGroup destination of this AssociationGroup. The AssociationGroup contains AssociationReferences with IsReverse attribute is set to 1, but the AssociationGroup has more than one AssociationGroup source. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, and ContainingAssociationGroupName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>-802</td>
<td>An AssociationGroup is in error because there is more than one Association which has MethodInstanceType (section 2.2.1.23) set to “Associator” or “Disassociator” referenced from the AssociationReferences of this AssociationGroup. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, and ContainingAssociationGroupName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>-803</td>
<td>An AssociationGroup is in error because one of the following conditions is true: - There are more than one Association which has MethodInstanceType set to &quot;BulkAssociatedIdEnumerator&quot; referenced from the AssociationReferences of this AssociationGroup with IsReverse attribute is set to 1, than the number of AssociationGroups sources of this AssociationGroup. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, and ContainingAssociationGroupName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>-804</td>
<td>An AssociationGroup is in error because it contains an AssociationReference which has IsReverse attribute specified as 1, but the Association it references has a MethodInstanceType other than &quot;AssociationNavigator&quot;, &quot;BulkAssociationNavigator&quot;, or &quot;BulkAssociatedIdEnumerator&quot;. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, and ContainingAssociationGroupName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>-805</td>
<td>An Association is in error because its MethodInstanceType is “BulkAssociationNavigator” and it is not referenced from an AssociationReference that is contained in an AssociationGroup which has another AssociationReference that references an Association with MethodInstanceType “AssociationNavigator” and has the same value for IsReverse. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, ContainingMethodName, and AGAssociationReferenceName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>MUST NOT be NULL. If the association is referenced from an AssociationReference, ContainingAssociationGroupName also MUST NOT be NULL; otherwise, it MUST be NULL. All other columns MUST be ignored by the protocol client.</td>
<td></td>
</tr>
<tr>
<td>-806</td>
<td>An Association is in error because it is referenced by two or more AssociationReferences. For this error code, ContainingEntityNamespace, ContainingEntityName, ContainingEntityVersion, ContainingMethodName, and AGAssociationReferenceName all MUST NOT be NULL. All other columns MUST be ignored by the protocol client.</td>
</tr>
</tbody>
</table>

**ContainingEntityNamespace:** The namespace of the Entity that is in error or contains the MetadataObject in error. The value MUST be NULL or a Name (section 2.2.1.2) depending on the error code.

**ContainingEntityName:** The name of the Entity that is in error or contains the MetadataObject in error. The value MUST be NULL or a Namespace (section 2.2.1.3) depending on the error code.

**ContainingEntityVersion:** The string representation of the version of the Entity that is in error or contains the MetadataObject in error. Following is the ABNF for the ContainingEntityVersion structure:

```
ContainingEntityVersion = Major %x2E Minor *1(%x2E Build *1(%x2E Revision))
```

**Major** MUST be equal to MajorVersion (section 2.2.1.7) of the Entity. **Minor** MUST be equal to MinorVersion (section 2.2.1.8) of the Entity. **Build** MUST be equal to BuildVersion (section 2.2.1.9) of the Entity. **Revision** MUST be equal to RevisionVersion (section 2.2.1.10) of the Entity.

**ContainingMethodName:** The name of the Method that contains the MetadataObject in error. The value MUST be NULL or a Name depending on the error code.

**ContainingParameterName:** The name of the Parameter that contains the MetadataObject in error. The value MUST be NULL or a Name depending on the error code.

**ContainingTypeDescriptorName:** The name of the TypeDescriptor that is in error. The value MUST be NULL or a Name depending on the error code.

**ContainingTypeDescriptorId:** The MetadataObjectId of the TypeDescriptor that is in error. The value MUST be NULL or an Id depending on the error code.

**ContainingAssociationGroupName:** The name of the AssociationGroup that is in error or contains the AssociationReference in error. The value MUST be NULL or a Name, depending on the error code.

**TDIDReferenceName:** The name of the Identifier referenced by the TypeDescriptor that is in error. The value MUST be NULL or a Name, depending on the error code.
**TDIDReferenceTypeName**: The name of the data type that is represented by the TypeDescriptor that is in error. The value MUST be NULL or a TypeDescriptorTypeName (section 2.2.1.25), depending on the error code.

**TDIDEntityReferenceName**: The name of the Entity containing the Identifier referenced by the TypeDescriptor that is in error. The value MUST be NULL or a Name, depending on the error code.

**TDIDEntityReferenceNamespace**: The namespace of the Entity containing the Identifier referenced by the b that is in error. The value MUST be NULL or a Namespace (section 2.2.1.3), depending on the error code.

**TDAssociationReferenceName**: The name of the Association referenced by the TypeDescriptor that is in error. The value MUST be NULL or a Name, depending on the error code.

**TDAssociationEntityReferenceName**: The name of the Entity that contains the Association referenced by the TypeDescriptor that is in error. The value MUST be NULL or a Name, depending on the error code.

**TDAssociationEntityReferenceNamespace**: The namespace of the Entity that contains the Association referenced by the TypeDescriptor that is in error. The value MUST be NULL or a Namespace, depending on the error code.

**AGAssociationReferenceName**: The name of the Association referenced by the AssociationReference that is in error. The value MUST be NULL or a Name, depending on the error code.

**AGAssociationEntityReferenceName**: The name of the Entity containing the Association referenced by the AssociationReference that is in error. The value MUST be NULL or a Name, depending on the error code.

**AGAssociationEntityReferenceNamespace**: The namespace of the Entity containing the Association referenced by the AssociationReference that is in error. The value MUST be NULL or a Namespace, depending on the error code.

### 2.2.5.32 Action Parameter Result Set

The Action Parameter result set contains information about ActionParameters. Each row in the result set MUST contain all the attributes of a single ActionParameter.

```sql
Id int,
ActionId int,
Index tinyint,
Name nvarchar(255),
IsCached bit,
PartitionId uniqueidentifier,
Version int,
```

**Id**: The MetadataObjectId of the ActionParameter. The value MUST be an Id (section 2.2.1.1).

**ActionId**: The MetadataObjectId of the Action that contains this ActionParameter. The value MUST be an Id.

**Index**: A value indicating the position of this ActionParameter among the other ActionParameters in the Action that contains this ActionParameter. The value MUST be an Index (section 2.2.1.19).
Name: The name of the ActionParameter. The value MUST be an ActionParameterName (section 2.2.1.41).

IsCached: A bit that specifies whether the ActionParameter is frequently used. The value MUST be an IsCached (section 2.2.1.5).

PartitionId: The Metadata partition of the ActionParameter. The value MUST be a PartitionId (section 2.2.1.4).

Version: The object version of this ActionParameter.

2.2.6 Tables and Views

None.

2.2.7 XML Structures

This specification does not define any common XML structure definitions.

2.2.7.1 Namespaces

None.

2.2.7.2 Simple Types

This specification does not define any common XML Schema simple type definitions.

2.2.7.3 Complex Types

This specification does not define any common XML Schema complex type definitions.

2.2.7.4 Elements

This specification does not define any common XML Schema element definitions.

2.2.7.5 Attributes

This specification does not define any common XML Schema attribute definitions.

2.2.7.6 Groups

This specification does not define any common XML Schema group definitions.

2.2.7.7 Attribute Groups

This specification does not define any common XML Schema attribute group definitions.
3 Protocol Details

3.1 Common Details

None.

3.2 Server Details

The back-end database protocol server responds only to stored procedure calls from the protocol client. It returns result sets and return codes and never initiates communication with other endpoints of the protocol.

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

For this protocol the back-end database server maintains lists to store the attributes of each of the following data types:

- ACE
- Action
- ActionParameter
- Association
- AssociationGroup
- AssociationReference
- Cache Version Stamp
- DataClass
- DefaultValue
- Entity
- FilterDescriptor
- Identifier
- LobSystem
- LobSystemInstance
- Localized name
- MetadataObject
- Method
• MethodInstance
• Model
• Parameter
• Property
• Throttle Configuration Setting (section 2.2.23)
• TypeDescriptor

The implementations of the basic Create, Read, Update, and Delete stored procedures simply insert, read, update or delete items in each of these lists where the MetadataObjectId serves as the primary identifier.

The containment and reference relationships can be captured through additional lists that store the primary identifiers of the related data types.

The protocol server maintains the following relationships and restrictions.

The MetadataObject data type (section 2.2.1) contains the following:
• Zero or more Property data types (section 2.2.2).
• Zero or more localized name data types (section 2.2.3).
• Zero or more ACE data types (section 2.2.4).

The Model data type (section 2.2.5) contains the following:
• Zero or more Property data types.
• Zero or more localized name data types.
• Zero or more ACE data types.

The Model data type references the following:
• Zero or more DataClass data types (section 2.2.8).
• Zero or more Entity data types (section 2.2.9).

The LobSystem data type (section 2.2.6) contains the following:
• Zero or more Property data types.
• Zero or more localized name data types.
• Zero or more ACE data types.
• Zero or more DataClass data types.
• Zero or more Entity data types.
• Zero or more LobSystemInstance data types (section 2.2.7).
• Zero or one SystemData (section 2.2.131).

The LobSystemInstance data type contains the following:
- Zero or more Property data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- The LobSystemInstance data type is contained by exactly one LobSystem data type.

The DataClass data type contains the following:
- Zero or more Property data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more Method data types (section 2.2.11).
- Zero or more MethodInstance data types (section 2.2.12).

The DataClass data type has the following restrictions:
- At most one of the DataClasses or Entity can be active across all DataClasses and Entities that have the same Name and Namespace.
- The DataClass data type is contained by exactly one LobSystem data type.

The Entity data type contains the following:
- Zero or more Property data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more Method data types.
- Zero or more MethodInstance data types.
- Zero or more Identifier data types (section 2.2.10).
- Zero or more Action data types (section 2.2.20).
- Zero or more AssociationGroup data types (section 2.2.18).

The Entity data type has the following restrictions:
- At most one of the Entity or DataClass can be active across all Entities and DataClasses that have the same Name and Namespace.
- The Entity data type is contained by exactly one LobSystem data type.

The Identifier data type contains the following:
- Zero or more Property data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
The **Identifier** data type is contained by exactly one **Entity** data type.

The **Method** data type contains the following:
- Zero or more **Property** data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more **FilterDescriptor** data types (section 2.2.2.16).
- Zero or more **Parameter** data types (section 2.2.2.14).
- Zero or more **MethodInstance** data types.
- Zero or more **Association** data types (section 2.2.2.13).

The **Method** data type is contained by either exactly one data type or exactly one **DataClass** data type.

The **MethodInstance** data type contains the following:
- Zero or more **Property** data types.
- Zero or more localized name data types.
- Zero or more ACE data types.

The **MethodInstance** data type references zero or one **TypeDescriptor** data type (section 2.2.2.15).

The **MethodInstance** data type has the following restrictions:
- The **MethodInstance** data type is contained by exactly one **Method** data type.
- The **MethodInstance** data type is contained by either exactly one **Entity** data type or exactly one **DataClass** data type.
- If the **MethodInstance** has a ReturnTypeDescriptor the MethodInstance data type references the **TypeDescriptor** data type that corresponds to the **ReturnTypeDescriptor**. Otherwise, the **MethodInstance** data type cannot reference any TypeDescriptor data types.
- The **Type** attribute cannot be "AssociationNavigator", "Associator", "Disassociator", "BulkAssociationNavigator", or "BulkAssociatedIdenumerator".

The **Association** data type contains the following:
- Zero or more **Property** data types.
- Zero or more localized name data types.
- Zero or more ACE data types.

The **Association** data type references the following:
- Zero or one **TypeDescriptor** data type.
- Two or more **Entity** data types.
The **Association** data type has the following restrictions:

- The **Association** data type is contained by exactly one **Method** data type.
- The **Association** data type is contained by either exactly one **Entity** data type or exactly one **DataClass** data type.
- If the **Association** has a **ReturnTypeDescriptor** the **Association** data type references the **TypeDescriptor** data type that corresponds to the **ReturnTypeDescriptor**. Otherwise, the **Association** data type cannot reference any **TypeDescriptor** data types.

The **Association** data type references the **Entity** data type that corresponds to the destination of the **Association**.

- The **Association** data type references all the **Entity** data types that correspond to the sources of the **Association**.
- The **Association** data type cannot reference an **Entity** data type, if the **Entity** that corresponds to the **Entity** data type is not a destination or source for the **Association**.
- The **Type** attribute can only be "AssociationNavigator", "Associator", "Disassociator", "BulkAssociationNavigator", or "BulkAssociatedIdenumerator".

The **Parameter** data type contains the following:

- Zero or more **Property** data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more **TypeDescriptor** data types.

The **Parameter** data type has the following restrictions:

- The **Parameter** data type is contained by exactly one **Method** data type.
- If the **Parameter** data type contains one or more **TypeDescriptor** data types, exactly one **TypeDescriptor** data type cannot be contained by another **TypeDescriptor** data type. The **TypeDescriptor** data type that is not contained by another **TypeDescriptor** data type corresponds to the **ReturnTypeDescriptor** of the **Parameter**.

The **TypeDescriptor** data type contains the following:

- Zero or more **Property** data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more **TypeDescriptor** data types.
- Zero or more **DefaultValue** data types (section 2.2.2.17).

The **TypeDescriptor** data type references the following:

- Zero or one **Identifier** data type.
- Zero or one **Association** data type.
- Zero or one `FilterDescriptor` data type.
- The `TypeDescriptor` data type is contained by exactly one `Parameter` data type or `TypeDescriptor` data type.

The `FilterDescriptor` data type contains the following:
- Zero or more `Property` data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- The `FilterDescriptor` data type is contained by exactly one `Method` data type.

The `DefaultValue` data type references either exactly one `MethodImpl` data type or exactly one `Association` data type.

The `DefaultValue` data type is contained by exactly one `TypeDescriptor` data type.

The `AssociationGroup` data type contains the following:
- Zero or more `Property` data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more `AssociationReference` data types (section 2.2.2.19).
- The `AssociationGroup` data type is contained by exactly one `Entity` data type.

The `AssociationReference` data type references exactly one `Association` data type.

The `AssociationReference` data type is contained by exactly one `AssociationGroup` data type.

The `Action` data type contains the following:
- Zero or more `Property` data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- Zero or more `ActionParameter` data types (section 2.2.21).
- The `Action` data type is contained by exactly one `Entity` data type.

The `ActionParameter` data type contains the following:
- Zero or more `Property` data types.
- Zero or more localized name data types.
- Zero or more ACE data types.
- The `ActionParameter` data type is contained by either exactly one `Action` data type.

The `Property` data type is contained by exactly one `MetadataObject` data type.
The localized name data type is contained by exactly one MetadataObject data type.
The ACE data type is contained by exactly one MetadataObject data type.
The Cache Version Stamp data type (section 2.2.2.22) does not have any relationships or restrictions.
The Throttle Configuration Setting data type does not have any relationships or restrictions.

3.2.2 Timers
None.

3.2.3 Initialization
None.

3.2.4 Higher-Layer Triggered Events
None.

3.2.5 Message Processing Events and Sequencing Rules
The T-SQL syntax for each stored procedure and result set, and the variables they are composed of, is defined in [MSDN-TSQL-Ref]. In the T-SQL syntax, the variable name is followed by the type of the variable which can optionally have a length value in brackets and can optionally have a default value indicated by an equals sign followed by the default value. Unless otherwise specified, all stored procedures defined in this section are located in the metadata store.

The definitions of some stored procedures, parameters and result sets in this section make use of ABNF representation as specified in [RFC5234].

3.2.5.1 proc_ar_ActivateEntity

The proc_ar_ActivateEntity stored procedure is called to set a version of an Entity active as follows.

    PROCEDURE proc_ar_ActivateEntity (    
        @Name nvarchar(255),    
        @Namespace nvarchar(255),    
        @PartitionId uniqueidentifier,    
        @MajorVersion int,    
        @MinorVersion int,    
        @BuildVersion int,    
        @RevisionVersion int,    
        @UniqueSessionId uniqueidentifier,    
        @Version int OUTPUT,    
        @ErrorCode int OUTPUT    
    );

@Name: The name of the Entity to activate. The value MUST be a Name (section 2.2.1.2).
@Namespace: The namespace of the Entity to activate. The value MUST be a Namespace (section 2.2.1.3).
@PartitionId: The Metadata partition that the Entity is obtained from. The value MUST be a PartitionId (section 2.2.1.4).

@MajorVersion: The major version of the Entity to activate. The value MUST be a MajorVersion (section 2.2.1.7).

@MinorVersion: The minor version of the Entity to activate. The value MUST be a MinorVersion (section 2.2.1.8).

@BuildVersion: The build version of the Entity to activate. The value MUST be a BuildVersion (section 2.2.1.9).

@RevisionVersion: The revision version of the Entity to activate. The value MUST be a RevisionVersion (section 2.2.1.10).

@UniqueSessionId: The session of the activation. The value MUST be a SessionId (section 2.2.1.36).

@Version: The object version of the Entity. The protocol client MUST set the value to the object version of the Entity at the time the Entity was last read by the protocol client. The protocol server MUST increment the object version of the Entity upon successful execution of this stored procedure. If the incremented object version of the Entity is equal to 2147483646, the protocol server MUST set the object version of the Entity to 0. The protocol server MUST return the object version of the Entity on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1009</td>
<td>The specified Entity is already active.</td>
</tr>
<tr>
<td>-1002</td>
<td>Another version of this Entity is already active.</td>
</tr>
<tr>
<td>-1000</td>
<td>Operation failed because of an inconsistency in the metadata store. This inconsistency identifies an error in the implementation of the protocol server.</td>
</tr>
<tr>
<td>-999</td>
<td>A reference error as specified in section 2.2.5.31 has been encountered during activation.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Entity has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Entity. For example, this error can be triggered when a thread reads the given Entity, after which another thread updates the same Entity, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>Entity does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>
Return Values: An integer that MUST be 0.
Result Sets: MUST NOT return any result sets.

3.2.5.2 proc_ar_AddEntity

The proc_ar_AddEntity stored procedure is called to add the specified DataClass to the specified Model. If the Model with the specified MetadataObjectId, already contains the DataClass with the specified MetadataObjectId, the state of the data in the metadata store is not considered to be in an error state. In this case, the proc_ar_AddEntity stored procedure MUST NOT change the state of the data in the metadata store. This stored procedure is defined as follows.

PROCEDURE proc_ar_AddEntity (  
    @ModelId int,
    @ClassId int OUTPUT
);

@ModelId: The MetadataObjectId of the Model to add the DataClass to. The value MUST be an Id.

@ClassId: The MetadataObjectId of the DataClass to be added to the Model. The value MUST be an Id.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.
Result Sets: MUST NOT return any result sets.

3.2.5.3 proc_ar_AddOrInsertLocalizedNameForMetadataObjectId

The proc_ar_AddOrInsertLocalizedNameForMetadataObjectId stored procedure is called to add a localized name for a MetadataObject for the specified LCID, in the specified Metadata partition. If a localized name already exists for the specified locale in the specified Setting, it MUST be replaced by the specified localized name. This stored procedure is defined as follows.

PROCEDURE proc_ar_AddOrInsertLocalizedNameForMetadataObjectId (  
    @MetadataObjectId int,
    @LocalizedName nvarchar(255)
);
@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (section 2.2.1.1).

@LocalizedName: The localized name of this MetadataObject for the specified locale.

@LCID: The LCID representing the locale of the specified localized name.

@SettingId: The Setting to which to write the localized name. The value MUST be a SettingId (section 2.2.1.6).

@PartitionId: The Metadata partition of the MetadataObject that contains the localized name to be added. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY&lt;13&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The specified MetadataObject contains implementation-specific maximum number of localized names.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified MetadataObject does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY&lt;14&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

| A positive integer | A T-SQL error code |

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.4 proc_ar_AddOrInsertPropertyForMetadataObjectId

The proc_ar_AddOrInsertPropertyForMetadataObjectId stored procedure is called to add a Property for a MetadataObject, in the specified Metadata partition. If a Property with the specified name already exists for the specified MetadataObject in the specified Setting, its value MUST be replaced by the specified value. This stored procedure is defined as follows.

PROCEDURE proc_ar_AddOrInsertPropertyForMetadataObjectId ( 
@MetadataObjectId int,
,@Name nvarchar(255),
,@Value sql_variant
@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (section 2.2.1.1).

@Name: The name of the Property.

@Value: The value of the Property.

@SettingId: The Setting to which to write the Property. The value MUST be a SettingId (section 2.2.1.6).

@PartitionId: The Metadata partition of the MetadataObject that contains the Property to be added. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;15&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The specified MetadataObject contains implementation-specific maximum number of Properties.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified MetadataObject does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;16&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

A positive integer A T-SQL error code

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

### 3.2.5.5 proc_ar_BulkSwitchActive

The proc_ar_BulkSwitchActive stored procedure is called to update the active version of the Entities. This stored procedure MUST set previously active versions of the Entities as not active. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_BulkSwitchActive ( @EntityIdList varchar(7000) ,@UniqueSessionId uniqueidentifier ,@PartitionId uniqueidentifier ,@Mode bit ,@ModelId int
```
@ErrorCode int OUTPUT,
@ErrorEntityId int OUTPUT,
@UpdatedEntityIdList varchar(8000) OUTPUT;

@EntityIdList: The list of Entity MetadataObjectId and corresponding object versions to set as active. Following is the ABNF for EntityIdList structure:

EntityIdList = 1*EntityVersionPair
EntityVersionPair = EntityId %x2d MOV %x2c
EntityId = 1*DIGIT
MOV = 1*DIGIT

EntityId MUST be the MetadataObjectId of the Entity. This value MUST be an Id (section 2.2.1.1). MOV MUST be the object version of the Entity. If the same b is specified multiple times in @EntityIdList, the protocol server MUST activate only the b with the highest version identified by MajorVersion (section 2.2.1.7), MinorVersion (section 2.2.1.8), BuildVersion (section 2.2.1.9), and RevisionVersion (section 2.2.1.10) fields, ignoring other versions of the same entity.

@UniqueSessionId: The session of the activation. The value MUST be a equal to SessionId (section 2.2.1.36).

@PartitionId: The Metadata partition that the Entities are obtained from. Value MUST be a PartitionId (section 2.2.1.4).

@Mode: A bit that specifies whether to change the state of the data stored in the protocol server. The value must be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This stored procedure MUST change the active versions of the Entities.</td>
</tr>
<tr>
<td>1</td>
<td>This stored procedure MUST verify that the Entities can be marked active without any reference errors, but MUST NOT change the state of the data stored in the protocol server.</td>
</tr>
</tbody>
</table>

@ModelId: The MetadataObjectId of the Model to add the active Entities to. If the value of this parameter is not NULL and is different from 0, this stored procedure MUST add the Entities it sets active to the Model. If the value of this parameter is NULL or 0, this stored procedure MUST NOT add the entities it sets active to any Model.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-999</td>
<td>A reference error as specified in section 2.2.5.31 has been encountered during activation.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;17&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6</td>
<td>One of the <strong>Entities</strong> has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>Entity</strong>. For example, this error can be triggered when a thread reads the given <strong>Entity</strong>, after which another thread updates the same <strong>Entity</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>One or more of the <strong>Entities</strong> do not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;18&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

#### A positive integer

A T-SQL error code

**@ErrorEntityId:** MetadataObjectId of the **Entity** that has an error. The value MUST be an **Id**.

**@UpdatedEntityIdList:** The stored procedure MUST set the value of this parameter to the list of **Entity MetadataObjectIds** and corresponding object versions after activation if the value of **@Mode** is 0. The stored procedure MUST set the value of this parameter to the value of **@EntityIdList** if the value of **@Mode** is 1. Following is the ABNF for **UpdatedEntityIdList** structure:

```
UpdatedEntityIdList = 1*EntityVersionPair
EntityVersionPair = EntityId %x2d MOV %x2c
EntityId = 1*DIGIT
MOV = 1*DIGIT
```

**EntityId** MUST be the **MetadataObjectId** of the **Entity**. This value MUST be an **Id**. MOV MUST be the object **Version** of the **Entity**.

**Return Values:** An integer that MUST be 0.

**Result Sets:**

If there are reference errors encountered this stored procedure MUST return an Activation Errors result set (section 2.2.5.31). Otherwise, this stored procedure MUST NOT return any result sets.

#### 3.2.5.6 proc_ar_BumpCacheInvalidationCounters

The **proc_ar_BumpCacheInvalidationCounters** stored procedure is called to increment the **Version** attribute of the Cache Version Stamps (section 2.2.2.22) stored in the metadata store. For each of the specified **Version** attributes, if the value of the attribute is at the implementation-specific maximum value before this stored procedure is called, the stored procedure MUST set the attribute value to 0. Otherwise, this stored procedure MUST increment the attribute value by 1. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_BumpCacheInvalidationCounters (   
```
@CacheLines bigint,
,@LastModified bigint,
,@PartitionId uniqueidentifier
);

@CacheLines: A bit mask representing which Cache Version Stamps to increment. The value MUST be a CacheLine (section 2.2.3.1).

@LastModified: Implementation specific timestamp of the operation.

@PartitionId: The Metadata partition of the Cache Version Stamps. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.7  proc_ar_ClearAccessControlEntriesForMetadataObject

The proc_ar_ClearAccessControlEntriesForMetadataObject stored procedure is called to delete all ACEs associated with both the specified MetadataObject and the specified Setting. This stored procedure is defined as follows.

PROCEDURE proc_ar_ClearAccessControlEntriesForMetadataObject (  
    @MetadataObjectId int
    ,@SettingId nvarchar(128)
);

@MetadataObjectId: The MetadataObjectId of the MetadataObject whose ACEs will be deleted. The value MUST be an Id (2.2.1.1).

@SettingId: The Setting to delete the ACEs from. The value MUST be a SettingId (section 2.2.1.6).

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.8  proc_ar_CopyAccessControlEntriesForMetadataObjectId

The proc_ar_CopyAccessControlEntriesForMetadataObjectId stored procedure is called to copy ACEs associated with a MetadataObject to another MetadataObject in the same Metadata partition. If @SourceMetadataObjectId and @DestinationMetadataObjectId are equal, this stored procedure MUST make no changes. If @SourceMetadataObjectId and @DestinationMetadataObjectId are not equal, this stored procedure MUST first delete all ACEs associated with the MetadataObject identified by the @DestinationMetadataObjectId, then, this stored procedure MUST duplicate the ACEs associated with the MetadataObject identified by the @SourceMetadataObjectId and associate the newly created ACEs with the MetadataObject identified by the @DestinationMetadataObjectId. This stored procedure is defined as follows.

PROCEDURE proc_ar_CopyAccessControlEntriesForMetadataObjectId (  
    @SourceMetadataObjectId int
    ,@DestinationMetadataObjectId int

@SourceMetadataObjectId: The MetadataObjectId of the MetadataObject from which the ACEs will be copied. The value MUST be an Id (2.2.1.1).

@DestinationMetadataObjectId: The MetadataObjectId of the MetadataObject with which the newly created ACEs will be associated. The value MUST be an Id.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.9 proc_ar_CopyAccessControlEntriesForSettings

The proc_ar_CopyAccessControlEntriesForSettings stored procedure is called to copy ACEs from the default Setting of a MetadataObject to the specified non-default Setting for the same MetadataObject. This stored procedure MUST delete all ACEs for the specified non-default Setting before the copying the ACEs. This stored procedure is defined as follows.

PROCEDURE proc_ar_CopyAccessControlEntriesForSettings (  
  @MetadataObjectId int,  
  @SettingId nvarchar(128)  
);

@MetadataObjectId: The MetadataObjectId for the MetadataObject for which ACEs values will be copied from default Setting to non-default Setting. The value MUST be an Id (2.2.1.1)

@SettingId: Setting to write the ACEs to. Value MUST be a SettingId (section 2.2.1.6).

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.10 proc_ar_CreateAction

The proc_ar_CreateAction stored procedure is called to create an Action in the specified Entity. This stored procedure is defined as follows.

PROCEDURE proc_ar_CreateAction (  
  @Name nvarchar(255),  
  @IsCached bit,  
  @PartitionId uniqueidentifier,  
  @EntityId int,  
  @Position tinyint,  
  @IsDisplayed bit,  
  @IsOpenedInNewWindow bit,  
  @Icon nvarchar(2080),  
  @Url nvarchar(2080),  
  @CreatedId int OUTPUT,  
  @ErrorCode int OUTPUT  
);

@Name: The name of the Action. The value MUST be a Name (section 2.2.1.2).
@IsCached: A bit that specifies whether the Action is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@Position: The Position attribute of the Action. The value MUST be a Position (section 2.2.1.14).

@IsDisplayed: The IsDisplayed attribute of the Action. The value MUST be an IsDisplayed (section 2.2.1.15).

@IsOpenedInNewWindow: The IsOpenedInNewWindow attribute of the Action. The value MUST be an IsOpenedInNewWindow (section 2.2.1.16).

@Icon: The Icon attribute of the Action. The value MUST be an Icon (section 2.2.1.17).

@Url: The URL attribute of the Action. The value MUST be a URL (section 2.2.1.18).

@CreatedId: The MetadataObjectId of the newly created Action. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the MetadataObjectId of the newly created Action. If so, the value MUST be an Id. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;19&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The Entity already contains the implementation-specific maximum allowed number of Actions.</td>
</tr>
<tr>
<td>-1</td>
<td>An Action with the specified name already exists within the specified Entity.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY&lt;20&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.11 proc_ar_CreateActionParameter

The proc_ar_CreateActionParameter stored procedure is called to create an ActionParameter in the specified Action. This stored procedure is defined as follows.
PROCEDURE proc_ar_CreateActionParameter (  
@Name nvarchar(255),  
@IsCached bit  
,@PartitionId uniqueidentifier  
,@ActionId int  
,@Index tinyint  
,@CreatedId int OUTPUT  
,@ErrorCode int OUTPUT  
);  

@Name: The name of the ActionParameter. The value MUST be an ActionParameterName (section 2.2.1.41).

@IsCached: A bit that specifies whether the ActionParameter is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition to create the ActionParameter for. The value MUST be a PartitionId (section 2.2.1.4).

@ActionId: The MetadataObjectId of the Action. The value MUST be an Id (2.2.1.1).

@Index: The Index attribute of the ActionParameter. The value MUST be an Index (section 2.2.1.19).

@CreatedId: The MetadataObjectId of the newly created ActionParameter. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the MetadataObjectId of the newly created ActionParameter. If so, the value MUST be an Id. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;21&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The Action already contains the implementation-specific maximum allowed number of ActionParameters.</td>
</tr>
<tr>
<td>-1</td>
<td>An ActionParameter with the specified name already exists within the specified Action.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;22&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.
3.2.5.12  proc_ar_CreateAdministrationMetadataCatalog

The proc_ar_CreateAdministrationMetadataCatalog stored procedure is called to create a MetadataCatalog for the specified metadata partition. This stored procedure is defined as follows.

PROCEDURE proc_ar_CreateAdministrationMetadataCatalog (  @PartitionId uniqueidentifier  ,@CreatedId int OUTPUT  ,@ErrorCode int OUTPUT  );

@PartitionId: The metadata partition for which to create the MetadataCatalog. The value MUST be a PartitionId (section 2.2.1.4).

@CreatedId: The MetadataObjectId of the newly created MetadataCatalog. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the MetadataCatalog. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter value is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-1</td>
<td>There is already a MetadataCatalog for the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.13  proc_ar_CreateAssociation

The proc_ar_CreateAssociation stored procedure is called to create an Association in the specified Method. The stored procedure MUST copy the ACEs of the Entity containing the specified Method to the newly created Association. This stored procedure is defined as follows.

PROCEDURE proc_ar_CreateAssociation (  @Name nvarchar(255)  ,@IsCached bit  ,@PartitionId uniqueidentifier  ,@MethodId int  ,@ReturnTypeDescriptorId int  );
The name of the Association. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies if the Association is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

@MethodId: The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

@ReturnTypeDescriptorId: The MetadataObjectId of the ReturnTypeDescriptor. If the Association has a ReturnTypeDescriptor the value MUST be an Id, otherwise the value MUST be NULL.

@Type: The type of the Association. The value MUST be a MethodInstanceType (section 2.2.1.23).

@SourceEntities: A list of name and namespaces of the sources of the Association. The following is ABNF for the SourceEntities structure:

SourceEntities = 1*(Entity %x2C)Entity = Namespace %x2C NameNamespace = EscapedStringName = EscapedStringEscapedString = 1*((%x00-%x2B) / (%x2D-%x5B) / (%x5D-%xFF)/ EscapedComma / EscapedSlash)EscapedComma = %5C %x2CEscapedSlash = %5C %x5C

For each Association source there MUST be a single Entity structure. The namespace and the name of the Association source MUST be equal to the Namespace and Name structures respectively when the EscapedComma and EscapedSlash rules are changes as follows:

EscapedComma = %x2C
EscapedSlash = %x5C

@DestinationEntity: The name and namespace of the destination of an Association. The following is the ABNF for the DestinationEntity structure:

DestinationEntity = Entity

The Entity structure is specified in the preceding @SourceEntities parameter. The namespace and the name of the destination of an Association MUST be equal to the Namespace and Name structures respectively when the EscapedComma and EscapedSlash rules are changes as follows:

EscapedComma = %x2C
EscapedSlash = %x5C

@CreatedId: The MetadataObjectId of the newly created Association. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the
**MetadataObjectId** of the newly created **Association**. If so, the value MUST be an **Id**. Upon return from this stored procedure with an **@ErrorCode** set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

**@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-300</td>
<td>The specified <strong>Association</strong> sources contain same <strong>Entity</strong> more than once.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY send retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The <strong>Association</strong> cannot be added to an active <strong>Entity</strong>.</td>
</tr>
<tr>
<td>-3</td>
<td>The number of MethodInstances associated with the specified Method is greater than an implementation-specific maximum limit.</td>
</tr>
<tr>
<td>-1</td>
<td>An <strong>Association</strong> with the specified name already exists within the <strong>Entity</strong> that contains the specified <strong>Method</strong>.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

**Return Values**: An integer that MUST be 0.

**Result Sets**: MUST NOT return any result sets.

### 3.2.5.14 proc_ar_CreateAssociationGroup

The **proc_ar_CreateAssociationGroup** stored procedure is called to create an AssociationGroup in the specified Entity. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateAssociationGroup (
  @Name nvarchar(255),
  @IsCached bit,
  @PartitionId uniqueidentifier,
  @EntityId int,
  @CreatedId int OUTPUT,
  @ErrorCode int OUTPUT );
```

**@Name**: The name of the MetadataObject. The value MUST be a **Name** (section 2.2.1.2).

**@IsCached**: A bit that specifies whether this **AssociationGroup** is frequently used. The value MUST be an **IsCached** (section 2.2.1.5).

**@PartitionId**: The Metadata partition of the **Entity**. The value MUST be a **PartitionId** (section 2.2.1.4).
@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@CreatedId: The MetadataObjectId of the newly created AssociationGroup. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the newly created AssociationGroup. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The AssociationGroup cannot be added to an active Entity.</td>
</tr>
<tr>
<td>-3</td>
<td>The Entity already contains the implementation-specific maximum number of AssociationGroups.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified Entity does not exist.</td>
</tr>
<tr>
<td>-1</td>
<td>The Entity already contains another AssociationGroup with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.15 proc_ar_CreateAssociationReference

The proc_ar_CreateAssociationReference stored procedure is called to create an AssociationReference in the specified AssociationGroup. The AssociationReference references the specified Association. This stored procedure is defined as follows.

PROCEDURE proc_ar_CreateAssociationReference ( 
  @_AssociationName nvarchar(255), 
  @_AssociationEntityName nvarchar(255), 
  @_AssociationEntityNamespace nvarchar(255), 
  @IsReverse bit, 
  @PartitionId uniqueidentifier, 
  @AssociationGroupId int, 
  @Version int OUTPUT, 
  @CreatedId int OUTPUT, 
  @ErrorCode int OUTPUT 
);
@_AssociationName: The name of the Association. The value MUST be a Name (section 2.2.1.2).

@_AssociationEntityName: The name of the Entity containing the Association. The value MUST be a Name.

@_AssociationEntityNamespace: The namespace of the Entity containing the Association. The value MUST be a Namespace (section 2.2.1.3).

@IsReverse: The IsReverse attribute the AssociationReference. Value MUST be IsReverse (section 2.2.1.37).

@PartitionId: The Metadata partition of the AssociationGroup. Value MUST be a PartitionId (section 2.2.1.4).

@AssociationGroupId: The MetadataObjectId of the AssociationGroup. The value MUST be an Id (2.2.1.1).

@Version: The object version of the AssociationGroup with the specified MetadataObjectId. The protocol client MUST set the value to the object version of the AssociationGroup at the time the AssociationGroup was last read by the protocol client. The protocol server MUST increment the object version of the AssociationGroup upon successful execution of this stored procedure. If the incremented object version of the AssociationGroup is equal to 2147483646, the protocol server MUST set the object version of the AssociationGroup to 0. The protocol server MUST return the object version of the AssociationGroup on output.

@CreatedId: The MetadataObjectId of the newly created AssociationReference. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the created AssociationReference. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The specified AssociationGroup cannot be modified because it belongs to an active Entity.</td>
</tr>
<tr>
<td>-6</td>
<td>The AssociationGroup with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the AssociationGroup. For example, this error can be triggered when a thread reads the given AssociationGroup, after which another thread updates the same AssociationGroup, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-3</td>
<td>The AssociationGroup already contains the implementation-specific maximum number of AssociationReferences.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified AssociationGroup does not exist.</td>
</tr>
<tr>
<td>-1</td>
<td>The AssociationGroup already contains another AssociationReference referencing the specified Association.</td>
</tr>
</tbody>
</table>
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

#### 3.2.5.16 proc_ar_CreateEntity

The **proc_ar_CreateEntity** stored procedure is called to create an Entity in the specified LobSystem. The stored procedure MUST copy the list of ACEs of the specified LobSystem to the newly created Entity. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateEntity (  
    @Name nvarchar(255),  
    @Namespace nvarchar(255),  
    @IsCached bit,  
    @PartitionId uniqueidentifier,  
    @MajorVersion int,  
    @MinorVersion int,  
    @BuildVersion int,  
    @RevisionVersion int,  
    @SystemId int,  
    @EstimatedInstanceCount int,  
    @CacheUsage int,  
    @ModelId int,  
    @CreatedId int OUTPUT,  
    @ErrorCode int OUTPUT);  
```

@Name: The name of the Entity. The value MUST be a Name (section 2.2.1.2).

@Namespace: The namespace of the Entity. The value MUST be a Namespace (section 2.2.1.3).

@IsCached: A bit that specifies if the Entity is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

@MajorVersion: Major version of the Entity. The value MUST be a MajorVersion (section 2.2.1.7).

@MinorVersion: Minor version of the Entity. The value MUST be a MinorVersion (section 2.2.1.8).

@BuildVersion: Build version of the Entity. The value MUST be a BuildVersion (section 2.2.1.9).
@RevisionVersion: Revision version of the Entity. The value MUST be a RevisionVersion (section 2.2.1.10).

@SystemId: The MetadataObjectId of the LobSystem. The value must be an Id (2.2.1.1).

@EstimatedInstanceCount: The EstimatedInstanceCount attribute of the Entity. The value must be an EstimatedInstanceCount (section 2.2.1.11).

@CacheUsage: The cache usage mode to be used in the Entity. The value must be a CacheUsage (section 2.2.1.13).

@ModelId: The MetadataObjectId of the Model with which to associate the Entity. The protocol server MUST verify that the passed in MetadataObjectId is neither equal to 0, nor NULL and ignore it otherwise. The value MUST be the MetadataObjectId of a Model that currently exists in the metadata store.

@CreatedId: The MetadataObjectId of the newly created Entity. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the MetadataObjectId of the newly created Entity. If so, the value MUST be an Id. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The number of Entities associated with the specified LobSystem is greater than an implementation-specific maximum limit.</td>
</tr>
<tr>
<td>-1</td>
<td>An Entity with the specified name, namespace, and version already exists within the specified LobSystem.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.17 proc_ar_CreateFilterDescriptor

The proc_ar_CreateFilterDescriptor stored procedure is called to create a FilterDescriptor in the specified Method. This stored procedure is defined as follows.

PROCEDURE proc_ar_CreateFilterDescriptor (  
    @Name nvarchar(255)  
)
@Name: The name of the FilterDescriptor. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether the b is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

@MethodId: The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

@FilterType: The type of the FilterDescriptor. The value MUST be a FilterType (section 2.2.1.20).

@FilterField: The implementation-specific identifier of the field (4) affected by the FilterDescriptor. The value MUST be a FilterField (section 2.2.1.21).

@CreatedId: The MetadataObjectId of the newly created FilterDescriptor. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the MetadataObjectId of the newly created FilterDescriptor. If so, the value MUST be an Id. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>The specified type is &quot;Timestamp&quot; and another FilterDescriptor with type &quot;Timestamp&quot; already exists for the specified Method.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The number of FilterDescriptors associated with the specified Method is greater than an implementation-specific maximum limit.</td>
</tr>
<tr>
<td>-1</td>
<td>A FilterDescriptor with the specified name already exists within the specified Method.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>
Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.18 proc_ar_CreateIdentifier

The proc_ar_CreateIdentifier stored procedure is called to create an Identifier in the specified Entity. This stored procedure MUST set the OrdinalNumber attribute of the created Identifier to 1 plus the current maximum OrdinalNumber attribute of all Identifiers contained by the specified Entity. This stored procedure is defined as follows.

PROCEDURE proc_ar_CreateIdentifier ( 
  @Name nvarchar(255)
  ,@IsCached bit
  ,@PartitionId uniqueidentifier
  ,@EntityId int
  ,@TypeName nvarchar(255)
  ,@CreatedId int OUTPUT
  ,@ErrorCode int OUTPUT
 )

@Name: The name of the Identifier. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether the Identifier is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@TypeName: The type name of the Identifier. The value MUST be an IdentifierTypeName (section 2.2.1.22).

@CreatedId: The MetadataObjectId of the newly created Identifier. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter value MUST be set to the MetadataObjectId of the newly created Identifier. If so, the value MUST be an Id. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;35&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>Identifier could not be added to the active Entity.</td>
</tr>
<tr>
<td>-3</td>
<td>The number of Identifiers associated with the specified Entity is greater than an implementation-specific maximum limit.</td>
</tr>
<tr>
<td>-1</td>
<td>An Identifier with the specified name already exists within the specified Entity.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

### 3.2.5.19 proc_ar_CreateMethod

The **proc_ar_CreateMethod** stored procedure is called to create a **Method** in the specified **DataClass**. The stored procedure MUST copy the list of ACEs of the specified **DataClass** to the newly created **Method**. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateMethod ( 
    @Name nvarchar(255) ,
    @IsCached bit ,
    @PartitionId uniqueidentifier ,
    @ClassId int ,
    @IsStatic bit ,
    @LobName nvarchar(255) ,
    @CreatedId int OUTPUT ,
    @ErrorCode int OUTPUT 
);
```

@**Name**: The name of the **Method**. The value MUST be a Name (section 2.2.1.2).

@**IsCached**: A bit that specifies whether the **Method** is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@**PartitionId**: The metadata partition of the **DataClass**. The value MUST be a PartitionId (section 2.2.1.4).

@**ClassId**: The MetadataObjectId of the **DataClass**. The value MUST be an Id (2.2.1.1).

@**IsStatic**: A IsStatic attribute of the **Method**. The value MUST be an IsStatic (section 2.2.1.33).

@**LobName**: The name of the operation on the line-of-business (LOB) system that corresponds to the **Method**. The value MUST be a MethodLobName (section 2.2.1.34).

@**CreatedId**: The MetadataObjectId of the newly created **Method**. Upon return from this stored procedure with an @**ErrorCode** set to 0, this parameter value MUST be set to the MetadataObjectId of the newly created **Method**. If so, the value MUST be an Id. Upon return from this stored procedure with an @**ErrorCode** set to a value other than 0, this parameter value is set to a value that MUST be ignored by the protocol client.

@**ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY(^&lt;37&gt;) retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The number of Methods associated with the specified DataClass is greater than an implementation-specific maximum limit.</td>
</tr>
<tr>
<td>-1</td>
<td>A Method with the specified name already exists within the specified DataClass.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY(^&lt;38&gt;) retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.20 proc_ar_CreateMethodInstance

The `proc_ar_CreateMethodInstance` stored procedure is called to create a MethodInstance in the specified Method. The stored procedure MUST copy the list of ACEs of the DataClass containing the specified Method to the newly created MethodInstance. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateMethodInstance (  
    @Name nvarchar(255),  
    @IsCached bit,  
    @PartitionId uniqueidentifier,  
    @MethodId int,  
    @ReturnTypeDescriptorId int,  
    @Type tinyint,  
    @IsDefault bit,  
    @CreatedId int OUTPUT,  
    @ErrorCode int OUTPUT  
);  
```

**@Name:** The name of the MetadataObject. The value MUST be a Name (section 2.2.1.2).

**@IsCached:** A bit that specifies whether the MethodInstance is frequently used. The value MUST be an IsCached (section 2.2.1.5).

**@PartitionId:** The Metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

**@MethodId:** The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

**@ReturnTypeDescriptorId:** The MetadataObjectId of the ReturnTypeDescriptor. If the MethodInstance has a ReturnTypeDescriptor the value MUST be an Id. Otherwise the value MUST be NULL.
@Type: The type of the MethodInstance. The value MUST be a MethodInstanceType (section 2.2.1.23).

@IsDefault: A bit that specifies if the MethodInstance is a default one. The value MUST be an IsDefault. When this value is set to 1, this stored procedure MUST set IsDefault attribute of all other MethodInstances that have the same MethodInstanceType attribute within the DataClass of the specified Method to 0. If this value is set to 0 and the DataClass of the specified Method does not contain any other MethodInstance with the specified MethodInstance type, the IsDefault attribute of the specified MethodInstance MUST be set to 1.

@CreatedId: The identifier of the newly created MethodInstance. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the newly created MethodInstance. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-217</td>
<td>The MethodInstance of the specified type requires input Parameter.</td>
</tr>
<tr>
<td>-216</td>
<td>An Entity or DataClass of the Method cannot contain more than one MethodInstance of type BulkIdEnumerator.</td>
</tr>
<tr>
<td>-215</td>
<td>The Method with the specified MetadataObjectId does not contain exactly one TimeStampFilter.</td>
</tr>
<tr>
<td>-214</td>
<td>The ReturnTypeDescriptor is required not to contain any TypeDescriptors for the specified type for the MethodInstance, however the specified ReturnTypeDescriptor contains one or more TypeDescriptors.</td>
</tr>
<tr>
<td>-211</td>
<td>An Entity or DataClass of the Method with the specified MetadataObjectId cannot contain more than one MethodInstance of type DeletedIdEnumerator.</td>
</tr>
<tr>
<td>-210</td>
<td>An Entity or DataClass of the Method with the specified MetadataObjectId cannot contain more than one MethodInstance of type ChangedIdEnumerator.</td>
</tr>
<tr>
<td>-209</td>
<td>An Entity or DataClass of the Method with the specified MetadataObjectId cannot contain more than one MethodInstance of type Deleter.</td>
</tr>
<tr>
<td>-208</td>
<td>The MethodInstance of the specified type requires ReturnTypeDescriptor to have &quot;IsCollection&quot; flag to be not set.</td>
</tr>
<tr>
<td>-207</td>
<td>The MethodInstance of the specified type requires ReturnTypeDescriptor to have &quot;IsCollection&quot; flag to be set.</td>
</tr>
<tr>
<td>-206</td>
<td>The MethodInstance of the specified type requires ReturnTypeDescriptor.</td>
</tr>
<tr>
<td>-205</td>
<td>An Entity or DataClass of the Method with the specified MetadataObjectId cannot contain more than one MethodInstance of type AccessChecker.</td>
</tr>
<tr>
<td>-204</td>
<td>The Parameter that contains the specified MetadataObjectId cannot have a Direction (section 2.2.1.24) set to &quot;In&quot;.</td>
</tr>
<tr>
<td>-203</td>
<td>The specified Method does not contain the Parameter that contains the specified ReturnTypeDescriptor.</td>
</tr>
<tr>
<td>-202</td>
<td>An Entity or DataClass of the Method with the specified MetadataObjectId cannot</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;39&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The Method with the specified MetadataObjectId already contains the implementation-specific maximum allowed number of MethodInstances.</td>
</tr>
<tr>
<td>-1</td>
<td>The DataClass of the Method with the specified MetadataObjectId already contains another MethodInstance with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors occurred.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;40&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.21   proc_ar_CreateModel

The proc_ar_CreateModel stored procedure is called to create a new Model. It MUST copy the list of ACEs of the MetadataCatalog of the specified Metadata partition to the newly created Model. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateModel ( @Name nvarchar(255), @IsCached bit, @PartitionId uniqueidentifier, @CreatedId int OUTPUT, @ErrorCode int OUTPUT );
```

@Name: The name of the Model. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this Model is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition to create the Model for. The value MUST be a PartitionId (section 2.2.1.4).

@CreatedId: The identifier for the newly created Model. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the newly created Model. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-1</td>
<td>A Model with the specified name already exists in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors occurred.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.22 proc_ar_CreateParameter

The **proc_ar_CreateParameter** stored procedure is called to create a Parameter contained by the specified Method. This stored procedure MUST set the **OrdinalNumber** attribute of the created Parameter to 1 plus the current maximum **OrdinalNumber** attribute of all Parameters contained by the specified Method. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateParameter (
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @MethodId int,
    @Direction tinyint,
    @CreatedId int OUTPUT,
    @ErrorCode int OUTPUT
);
```

**@Name:** The name of the Parameter. The value MUST be a Name (section 2.2.1.2).

**@IsCached:** A bit that specifies whether the Parameter is frequently used. The value MUST be an IsCached (section 2.2.1.5).

**@PartitionId:** The metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

**@MethodId:** The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

**@Direction:** The direction of the Parameter. The value MUST be a Direction (section 2.2.1.24).

**@CreatedId:** The identifier for the newly created Parameter. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the newly created Parameter. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

**@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
### Return Values:
An integer that MUST be 0.

### Result Sets:
MUST NOT return any result sets.

#### 3.2.5.23 proc_ar_CreateSystem

The **proc_ar_CreateSystem** stored procedure is called to create a LobSystem. It MUST copy the list of ACEs of the MetadataCatalog associated with the specified metadata partition to the newly created LobSystem. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateSystem (
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @SystemType tinyint,
    @CreatedId int OUTPUT,
    ErrorCode int OUTPUT);
```

- **@Name**: The name of the **LobSystem**. The value MUST be a **Name** (section 2.2.1.2).
- **@IsCached**: A bit that specifies whether the **LobSystem** is frequently used. The value MUST be an **IsCached** (section 2.2.1.5).
- **@PartitionId**: The metadata partition in which to create the MetadataObject. The value MUST be a **PartitionId** (section 2.2.1.4).
- **@SystemType**: Type of the **LobSystem**. The value MUST be a **SystemType** (section 2.2.1.30).
- **@CreatedId**: The identifier for the newly created **LobSystem**. Upon return from this stored procedure with an **ErrorCode** set to 0, this parameter MUST be set to the MetadataObjectId of the newly created **LobSystem**. Upon return from this stored procedure with an **ErrorCode** set to a value other than 0, this parameter is set to a value that MUST be ignored.
@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;45&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-1</td>
<td>The LobSystem with the specified name already exists in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors occurred.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;46&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.
Result Sets: MUST NOT return any result sets.

3.2.5.24 proc_ar_CreateSystemInstance

The proc_ar_CreateSystemInstance stored procedure is called to create a LobSystemInstance in the specified LobSystem. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CreateSystemInstance (  
    @Name nvarchar(255),  
    @IsCached bit,  
    @PartitionId uniqueidentifier,  
    @SystemId int,  
    @CreatedId int OUTPUT,  
    @ErrorCode int OUTPUT  
);  
```

@Name: The name of the LobSystemInstance. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this LobSystemInstance is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

@SystemId: The MetadataObjectId of the LobSystem. The value MUST be an Id (2.2.1.1).

@CreatedId: The identifier for the newly created LobSystemInstance. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the newly created LobSystemInstance. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-3</td>
<td>The LobSystem with the specified MetadataObjectId already contains the implementation-specific maximum allowed number of LobSystemInstances.</td>
</tr>
<tr>
<td>-1</td>
<td>The specified LobSystem already contains a LobSystemInstance with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors occurred.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.25 proc_ar_CreateTypeDescriptor

The proc_ar_CreateTypeDescriptor stored procedure is called to create a TypeDescriptor contained by the specified Parameter. If a TypeDescriptor is also specified, the created TypeDescriptor MUST also be contained by the specified TypeDescriptor. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_CreateTypeDescriptor (  
    @Name nvarchar(255),  
    @IsCached bit,  
    @PartitionId uniqueidentifier,  
    @ParameterId int,  
    @ParentTypeDescriptorId int,  
    @TypeName nvarchar(255),  
    @IdentifierId int,  
    @FilterDescriptorId int,  
    @LobName nvarchar(255),  
    @Flags smallint,  
    @AssociationId int,  
    @IdentifierName nvarchar(255),  
    @IdentifierEntityName nvarchar(255),  
    @IdentifierEntityNamespace nvarchar(255),  
    @AssociationName nvarchar(255),  
    @AssociationEntityName nvarchar(255),  
    @AssociationEntityNamespace nvarchar(255),  
    @CreatedId int OUTPUT,  
    @ErrorCode int OUTPUT);  
```

@Name: The name of the TypeDescriptor. The value MUST be a Name (section 2.2.1.2).
@IsCached: A bit that specifies whether the TypeDescriptor is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).

@ParameterId: The MetadataObjectId of the Parameter. The value MUST be an Id (2.2.1.1).

@ParentTypeDescriptorId: The MetadataObjectId of the TypeDescriptor that MUST contain the created TypeDescriptor. To create the root TypeDescriptor this value MUST be NULL. Otherwise the value MUST be an Id.

@TypeName: The name of the data type that is represented by this TypeDescriptor. The value MUST be a TypeDescriptorTypeName (section 2.2.1.25).

@IdentifierId: The MetadataObjectId of the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an active Entity, the value MUST be an Id. Otherwise, the value MUST be NULL or 0.

@FilterDescriptorId: The MetadataObjectId of the FilterDescriptor associated with the TypeDescriptor. If a FilterDescriptor is associated with the TypeDescriptor, the value MUST be an Id. Otherwise the value MUST be NULL.

@LobName: The name of the data structure that is represented by the TypeDescriptor. The value MUST be a TypeDescriptorLobName (section 2.2.1.26).

@Flags: The flags for the TypeDescriptor. The value MUST be TypeDescriptorFlags (section 2.2.1.28).

@AssociationId: The MetadataObjectId of the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association defined on an active Entity, the value MUST be an Id. Otherwise, the value MUST be NULL or 0.

 @_IdentifierName: The name of the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Name. Otherwise the value MUST be NULL.

 @_IdentifierEntityName: The name of the Entity that contains the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Name. Otherwise the value MUST be NULL.

 @_IdentifierEntityNamespace: The namespace of the Entity that contains the Identifier referenced by the TypeDescriptor. If the TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Namespace. Otherwise the value MUST be NULL.

 @_AssociationName: The name of the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Name. Otherwise the value MUST be NULL.

 @_AssociationEntityName: The name of the Entity that contains the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Name. Otherwise the value MUST be NULL.

 @_AssociationEntityNamespace: The namespace of the Entity that contains the Association referenced by the TypeDescriptor. If the TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Namespace (section 2.2.1.3). Otherwise the value MUST be NULL.
@CreatedId: The identifier for the newly created TypeDescriptor. Upon return from this stored procedure with an @ErrorCode set to 0, this parameter MUST be set to the MetadataObjectId of the newly created TypeDescriptor. Upon return from this stored procedure with an @ErrorCode set to a value other than 0, this parameter is set to a value that MUST be ignored.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-309</td>
<td>The &quot;ReadOnly&quot; flag cannot be set for TypeDescriptor, because the specified Parameter has value &quot;In&quot; for the Direction attribute (section 2.2.1.24).</td>
</tr>
<tr>
<td>-308</td>
<td>A MetadataObjectId is specified for the Association referenced by the TypeDescriptor but the Entity that contains the specified Association is not active.</td>
</tr>
<tr>
<td>-307</td>
<td>A MetadataObjectId is specified for the Identifier referenced by the TypeDescriptor but the Entity that contains the specified Identifier is not active.</td>
</tr>
<tr>
<td>-306</td>
<td>The TypeDescriptor with the specified MetadataObjectId has &quot;IsCollection&quot; flag set and already contains another TypeDescriptor. A TypeDescriptor with &quot;IsCollection&quot; flag set cannot contain more than one TypeDescriptor.</td>
</tr>
<tr>
<td>-305</td>
<td>The TypeDescriptor with the specified MetadataObjectId has &quot;IsCollection&quot; flag set and &quot;IsCollection&quot; flag is also set for the created TypeDescriptor. A TypeDescriptor with &quot;IsCollection&quot; flag set cannot contain another TypeDescriptor that has &quot;IsCollection&quot; flag set.</td>
</tr>
<tr>
<td>-303</td>
<td>The Parameter with the specified MetadataObjectId and the FilterDescriptor with the specified MetadataObjectId do not belong to the same Method.</td>
</tr>
<tr>
<td>-302</td>
<td>The @ParentTypeDescriptorId is equal to NULL and the Parameter with the specified MetadataObjectId already has a root TypeDescriptor.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The Entity containing the Method containing the Parameter with the specified MetadataObjectId is active, but this TypeDescriptor references at least one of either Association or Identifier of an Entity that is not active.</td>
</tr>
<tr>
<td>-3</td>
<td>At least one of the following two statements is true:</td>
</tr>
<tr>
<td></td>
<td>• The TypeDescriptor to be created is not a root TypeDescriptor and the specified TypeDescriptor already has the implementation-specific maximum number of child TypeDescriptors.</td>
</tr>
<tr>
<td></td>
<td>• A FilterDescriptor is associated to the TypeDescriptor and the FilterDescriptor already has the implementation-specific maximum number of associated TypeDescriptors.</td>
</tr>
<tr>
<td>-1</td>
<td>The TypeDescriptor with the specified MetadataObjectId already contains another TypeDescriptor with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
<tr>
<td>-300</td>
<td>The Parameter with the specified MetadataObjectId already has a TypeDescriptor hierarchy deeper than the implementation-specific maximum level allowed.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

#### 3.2.5.26 proc_ar_DeactivateEntity

The proc_ar_DeactivateEntity stored procedure is called to set the active version of an Entity as not active. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeactivateEntity (  
    @Name nvarchar(255),  
    @Namespace nvarchar(255),  
    @PartitionId uniqueidentifier,  
    @MajorVersion int,  
    @MinorVersion int,  
    @BuildVersion int,  
    @RevisionVersion int,  
    @UniqueSessionId uniqueidentifier,  
    @Version int OUTPUT,  
    @ErrorCode int OUTPUT  
);  
```

**@Name:** The name of the **Entity** to deactivate. The value MUST be a Name (section 2.2.1.2).

**@Namespace:** The namespace of the **Entity** to deactivate. The value MUST be Namespace (section 2.2.1.3).

**@PartitionId:** The metadata partition of the **Entity** to deactivate. Value MUST be a PartitionId (section 2.2.1.4).

**@MajorVersion:** The major version of the **Entity** to deactivate. The value MUST be a MajorVersion (section 2.2.1.7).

**@MinorVersion:** The minor version of the **Entity** to deactivate. The value MUST be a MinorVersion (section 2.2.1.8).

**@BuildVersion:** The build version of the **Entity** to deactivate. The value MUST be a BuildVersion (section 2.2.1.9).

**@RevisionVersion:** The revision version of the **Entity** to deactivate. The value MUST be a RevisionVersion (section 2.2.1.10).

**@UniqueSessionId:** The session of the deactivation. The value MUST be a SessionId (section 2.2.1.36).

**@Version:** The object version of the **Entity**. The protocol client MUST set the value to the object version of the **Entity** at the time the **Entity** was last read by the protocol client. The protocol server MUST increment the object version of the **Entity** upon successful execution of this stored procedure.
If the incremented object version of the Entity is equal to 2147483646, the protocol server MUST set the object version of the Entity to 0. The protocol server MUST return the object version of the Entity on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1010</td>
<td>The specified Entity is already not active.</td>
</tr>
<tr>
<td>-1006</td>
<td>Multiple versions of the Entity are marked as active. This happens when there is inconsistency in the metadata store.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Entity has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Entity. For example, this error can be triggered when a thread reads the given Entity, after which another thread updates the same Entity, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified b does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

### 3.2.5.27 proc_ar_DeleteActionById

The proc_ar_DeleteActionById stored procedure is called to delete the specified Action in a given Metadata partition. Action MUST be deleted along with its Properties, localized names, ACEs, and ActionParameters. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteActionById (  
    @Id int,  
    @Version int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT  
);  
```

@Id: The MetadataObjectId of the Action. The value MUST be an Id (2.2.1.1).

@Version: The object version of the Action.
@PartitionId: The Metadata partition of the Action. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>An Action with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Action. For example, this error can be triggered when a thread reads the given Action, after which another thread updates the same Action, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An Action with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.28 proc_ar_DeleteActionParameterById

The proc_ar_DeleteActionParameterById stored procedure is called to delete the specified ActionParameter in the given metadata partition. ActionParameter MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

PROCEDURE proc_ar_DeleteActionParameterById ( 
  @Id int, 
  @Version int, 
  @PartitionId uniqueidentifier, 
  @ErrorCode int OUTPUT 
);

@Id: The MetadataObjectId of the ActionParameter. The value MUST be an Id (2.2.1.1).

@Version: The object version of this ActionParameter.

@PartitionId: The metadata partition of the ActionParameter. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.
### Return Values:
An integer that MUST be 0.

### Result Sets:
MUST NOT return any result sets.

#### 3.2.5.29 `proc_ar_DeleteAdministrationMetadataCatalog`

The `proc_ar_DeleteAdministrationMetadataCatalog` stored procedure is called to delete the MetadataCatalog and all the MetadataObjects from the given metadata partition. MetadataCatalog MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteAdministrationMetadataCatalog (
    @PartitionId uniqueidentifier,
    @ErrorCode int OUTPUT
);
```

- **@PartitionId**: The metadata partition of the MetadataCatalog. The value MUST be a PartitionId (section 2.2.1.4).
- **@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-2</td>
<td>A MetadataCatalog does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>
### Value | Description
---|---
-1100 | The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.

**A positive integer** | A T-SQL error code.

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

#### 3.2.5.30 proc_ar_DeleteAssociationById

The `proc_ar_DeleteAssociationById` stored procedure is called to delete the specified Association. **Association** MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteAssociationById (  
    @Id int,  
    @Version int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT  
);
```

**@Id:** The MetadataObjectId of the **Association**. The value MUST be an **Id** (2.2.1.1).

**@Version:** The object version of this **Association**.

**@PartitionId:** The metadata partition of the **Association**. The value MUST be a **PartitionId** (section 2.2.1.4).

**@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>Cannot delete an <strong>Association</strong> contained by an active Entity.</td>
</tr>
<tr>
<td>-6</td>
<td>The <strong>Association</strong> with the specified <strong>MetadataObjectId</strong> has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>Association</strong>. For example, this error can be triggered when a thread reads the given <strong>Association</strong>, after which another thread updates the same <strong>Association</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>The <strong>Association</strong> with the specified <strong>MetadataObjectId</strong> does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The...</td>
</tr>
</tbody>
</table>
### 3.2.5.31 proc_ar_DeleteAssociationGroupById

The `proc_ar_DeleteAssociationGroupById` stored procedure is called to delete the specified AssociationGroup. The AssociationGroup MUST be deleted along with its Properties, localized names, and all of its AssociationReferences. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteAssociationGroupById (  
    @Id int,  
    @Version int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT
);
```

- **@Id**: The MetadataObjectId of the AssociationGroup. The value MUST be an Id (2.2.1.1).
- **@Version**: The object version of the AssociationGroup.
- **@PartitionId**: The Metadata partition of the AssociationGroup. The value MUST be a PartitionId (section 2.2.1.4).
- **@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>Cannot delete an AssociationGroup contained by an active Entity.</td>
</tr>
<tr>
<td>-6</td>
<td>The AssociationGroup with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the AssociationGroup. For example, this error can be triggered when a thread reads the given AssociationGroup, after which another thread updates the same AssociationGroup, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An AssociationGroup with the specified MetadataObjectId does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.
3.2.5.32 proc_ar_DeleteAssociationReferenceById

The proc_ar_DeleteAssociationReferenceById stored procedure is called to delete the specified AssociationReference. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteAssociationReferenceById (  
   @Id int,  
   @PartitionId uniqueidentifier,  
   @Version int OUTPUT,  
   @ErrorCode int OUTPUT  
);
```

@Id: The implementation-specific identifier of the AssociationReference.

@PartitionId: The Metadata partition of the AssociationGroup that contains the AssociationReference. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the AssociationGroup in which the specified AssociationReference contained. The protocol client MUST set the value to the object version of the AssociationGroup is contained at the time the AssociationGroup was last read by the protocol client. The protocol server MUST increment the object version of the AssociationGroup upon successful execution of this stored procedure. If the incremented object version of the AssociationGroup is equal to 2147483646, the protocol server MUST set the object version of the AssociationGroup to 0. The protocol server MUST return the object version of the AssociationGroup on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>Cannot delete the AssociationReference that is contained by an AssociationGroup contained by an active Entity.</td>
</tr>
<tr>
<td>-6</td>
<td>The AssociationGroup of the AssociationReference with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the AssociationGroup. For example, this error can be triggered when a thread reads the given AssociationGroup, after which another thread updates the same AssociationGroup, and then the original thread tries to update.</td>
</tr>
</tbody>
</table>
### Return Values:
An integer that MUST be 0.

### Result Sets:
MUST NOT return any result sets.

### 3.2.5.33 proc_ar_DeleteDefaultValue

The `proc_ar_DeleteDefaultValue` stored procedure is called to delete the `DefaultValue` (section 2.2.2.17) identified by the specified `TypeDescriptor` and `MethodInstance`. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteDefaultValue (  
    @TypeDescriptorId int,  
    @MethodInstanceId int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT
);
```

- **@TypeDescriptorId**: The `MetadataObjectId` of the `TypeDescriptor` associated with the `DefaultValue`. The value MUST be an `Id` (section 2.2.1.1).

- **@MethodInstanceId**: The `MetadataObjectId` of the `MethodInstance` associated with the `DefaultValue`. The value MUST be an `Id`.

- **@PartitionId**: The metadata partition of the `TypeDescriptor` and the `MethodInstance` associated with the `DefaultValue`. The value MUST be a `PartitionId` (section 2.2.1.4).

- **@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set by the protocol server to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The specified <code>AssociationReference</code> does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

A positive integer is a T-SQL error code.

### Result Sets:
MUST NOT return any result sets.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| -2    | At least one of the following conditions is true:  
  - A `TypeDescriptor` with the specified `MetadataObjectId` does not exist in the specified Metadata partition.  
  - A `MethodInstance` with the specified `MetadataObjectId` does not exist in the specified Metadata partition. |
| 0     | No errors encountered. |
| -1100 | The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again. |
### Return Values:
An integer that MUST be 0.

### Result Sets:
MUST NOT return any result sets.

#### 3.2.5.34 proc_ar_DeleteEntityById

The `proc_ar_DeleteEntityById` stored procedure is called to delete the specified Entity. **Entity** MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteEntityById (  
    @Id int,  
    @Version int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT
);
```

- **@Id:** The MetadataObjectId of the **Entity**. The value MUST be an Id (2.2.1.1).
- **@Version:** The object version of the **Entity**.
- **@PartitionId:** The metadata partition of the **Entity**. The value MUST be a PartitionId (section 2.2.1.4).
- **@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;66&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The <strong>Entity</strong> with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>Entity</strong>. For example, this error can be triggered when a thread reads the given <strong>Entity</strong>, after which another thread updates the same <strong>Entity</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-5</td>
<td>The <strong>Entity</strong> with the specified MetadataObjectId contains at least one of the following child objects:</td>
</tr>
<tr>
<td></td>
<td>- Action</td>
</tr>
<tr>
<td></td>
<td>- Method</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-2</td>
<td>An <strong>Entity</strong> with the specified <strong>MetadataObjectId</strong> does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;68&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.35 proc_ar_DeleteFilterDescriptorById

The **proc_ar_DeleteFilterDescriptorById** stored procedure is called to delete the FilterDescriptor identified by the specified MetadataObjectId. **FilterDescriptor** MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteFilterDescriptorById (  
    @Id int,  
    @Version int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT  
);
```

- **@Id:** The MetadataObjectId of the **FilterDescriptor**. The value MUST be an **Id** (2.2.1.1).
- **@Version:** The object version of this **FilterDescriptor**.
- **@PartitionId:** The metadata partition of the **FilterDescriptor**. The value MUST be a **PartitionId** (section 2.2.1.4).
- **@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>The <strong>FilterDescriptor</strong> to be deleted is of type TimeStampFilter and it is currently used in a MethodInstance of type ChangedIdEnumerator or DeletedIdEnumerator.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;69&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>
| -6    | The **FilterDescriptor** with the specified **MetadataObjectId** has been updated by a context other than the one that it has been currently read by. This happens when the specified...
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>object version is not equal to the current object version of the FilterDescriptor. For example, this error can be triggered when a thread reads the given FilterDescriptor, after which another thread updates the same FilterDescriptor, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>A FilterDescriptor with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.36 proc_ar_DeleteIdentifierById

The `proc_ar_DeleteIdentifierById` stored procedure is called to delete the specified Identifier. Identifier MUST be deleted along with its Properties, localized names, and ACEs. After a successful deletion, the OrdinalNumber attribute of all Identifiers that are contained by the Entity that contained the deleted Identifier MUST be normalized. After normalization, the ordinal number of all these Identifiers MUST be renumbered starting from 0, incrementing by 1 and preserving the original order. During this renumbering, the protocol server MUST increment the object version of all these Identifiers. After incrementing the object versions, the protocol server MUST set the object version of all these Identifiers, whose object version is 2,147,483,646, to 0. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteIdentifierById (  
  @Id int
  ,@Version int
  ,@PartitionId uniqueidentifier
  ,@DeleteActiveReferences bit
  ,@ErrorCode int OUTPUT
);
```

**@Id:** The MetadataObjectId of the Identifier. The value MUST be an Id (2.2.1.1).

**@Version:** The object version of this Identifier.

**@PartitionId:** The metadata partition of the Identifier. The value MUST be a PartitionId (section 2.2.1.4).

**@DeleteActiveReferences:** A bit that specifies whether the Identifiers of active Entities need to be deleted.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Identifier MUST NOT be deleted if the Entity that contains the specified b is active.</td>
</tr>
<tr>
<td>1</td>
<td>The Identifier MUST be deleted regardless of the active status of the Entity that contains the specified Identifier.</td>
</tr>
</tbody>
</table>

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The Entity that contains this Identifier was active and the value of @DeleteActiveReferences parameter was 0.</td>
</tr>
<tr>
<td>-6</td>
<td>The Identifier with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Identifier. For example, this error can be triggered when a thread reads the given Identifier, after which another thread updates the same Identifier, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An Identifier with the specified MetadataObjectId does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.37 proc_ar_DeleteLocalizedNameForMetadataObjectByLCID

The proc_ar_DeleteLocalizedNameForMetadataObjectByLCID stored procedure is called to delete a localized name contained by the specified MetadataObject for a given LCID. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteLocalizedNameForMetadataObjectByLCID ( @MetadataObjectId int ,@LCID int ,@SettingId nvarchar(128) ,@PartitionId uniqueidentifier ,@ErrorCode int OUTPUT );
```
@MetadataObjectId: The MetadataObjectId of the MetadataObject that contains the localized name. The value MUST be an Id (2.2.1.1).

@LCID: The LCID of the localized name.

@SettingId: The Setting from which to delete the localized name. The value MUST be a SettingId (section 2.2.1.6).

@PartitionId: The metadata partition of the MetadataObject. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>A localized name for the given LCID does not exist for the specified MetadataObject in the specified Setting.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>Operation was cancelled because of an implementation-specific integrity violation. Protocol client MAY &lt;73&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
<tr>
<td>-8</td>
<td>Operation was cancelled because of an implementation-specific resource requirement. Protocol client MAY &lt;74&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

### 3.2.5.38 proc_ar_DeleteLocalizedNamesByMetadataObjectId

The proc_ar_DeleteLocalizedNamesByMetadataObjectId stored procedure is called to delete all localized names of the specified MetadataObject for a specified Setting. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_DeleteLocalizedNamesByMetadataObjectId (
  @MetadataObjectId int,
  @SettingId nvarchar(128),
  @ErrorCode int OUTPUT
);
```

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1).

@SettingId: The Setting to delete the localized names from. The value MUST be a SettingId (section 2.2.1.6).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
### proc_ar_DeleteMethodById

The `proc_ar_DeleteMethodById` stored procedure is called to delete the Method identified by the specified `MetadataObjectId`. The Method MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteMethodById (  
@Id int  
, @Version int  
, @PartitionId uniqueidentifier  
, @ErrorCode int OUTPUT  
);
```

- **@Id**: The `MetadataObjectId` of the Method. The value MUST be an `Id` (2.2.1.1).
- **@Version**: The object version of this Method.
- **@PartitionId**: The metadata partition of the Method. The value MUST be a `PartitionId` (section 2.2.1.4).
- **@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Method with the specified <code>MetadataObjectId</code> has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Method. For example, this error can be triggered when a thread reads the given Method, after which another thread updates the same Method, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-5</td>
<td>The specified Method contains at least one child object of type FilterDescriptor, MethodInstance, or Parameter.</td>
</tr>
<tr>
<td>-2</td>
<td>A Method with the specified <code>MetadataObjectId</code> does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.
3.2.5.40 proc_ar_DeleteMethodInstanceById

The proc_ar_DeleteMethodInstanceById stored procedure is called to delete the MethodInstance identified by the specified MetadataObjectId. MethodInstance MUST be deleted along with its Properties, localized names, and ACEs. It MUST also delete any DefaultValues (section 2.2.2.17) associated with the MethodInstance identified by the specified MetadataObjectId. If the MethodInstance to be deleted is a default MethodInstance, and if there is another MethodInstance of the same MethodInstance type for the same DataClass that contains the MethodInstance to be deleted, then it SHOULD<77> be marked as default. This stored procedure is defined as follows.

PROCEDURE proc_ar_DeleteMethodInstanceById (  
  @Id int  
  ,@Version int  
  ,@PartitionId uniqueidentifier  
  ,@ErrorCode int OUTPUT  
);  

@Id: The MetadataObjectId of the MethodInstance. The value MUST be an Id (2.2.1.1).

@Version: The object version of this MethodInstance.

@PartitionId: The Metadata partition of the MethodInstance. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY&lt;76&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The MethodInstance with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the MethodInstance. For example, this error can be triggered when a thread reads the given MethodInstance, after which another thread updates the same MethodInstance, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>A MethodInstance with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>Operation was cancelled because of an implementation-specific integrity violation. Protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.41 `proc_ar_DeleteModelById`

The `proc_ar_DeleteModelById` stored procedure is called to delete the specified Model. It optionally checks if there are any DataClasses that are referenced by the Model to be deleted but are not referenced by any other Model before deleting the Model. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteModelById (  
  @Id int,  
  @Version int,  
  @AllowOrphanedEntities bit,  
  @PartitionId uniqueidentifier,  
  @ErrorCode int OUTPUT
);
```

- **@Id:** The MetadataObjectId of the Model. This value MUST be an Id (2.2.1.1).
- **@Version:** The object version of the Model.
- **@AllowOrphanedEntities:** A bit specifying whether to check the existence of any DataClasses that are referenced by to the Model to be deleted but are not referenced by any other Model. The value MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Model MUST NOT be deleted if there are any DataClasses that are referenced by to the Model to be deleted but are not referenced by any other Models.</td>
</tr>
<tr>
<td>1</td>
<td>The Model MUST be deleted regardless of the DataClasses that are referenced by to the Model. This will cause the DataClasses that are referenced by to the Model to end up not being referenced by any Models upon successful execution of this stored procedure.</td>
</tr>
</tbody>
</table>

- **@PartitionId:** The metadata partition of the Model. Value MUST be a PartitionId (section 2.2.1.4).
- **@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
### Value | Description
---|---
-8 | The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.

-6 | The Model with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current version of the Model. For example, this error can be triggered when a thread reads the given Model, after which another thread updates the same Model, and then the original thread tries to update.

-5 | There exists at least one DataClass that are referenced by the Model to be deleted but are not referenced by any other Model and @AllowOrphanedEntities parameter is set to 0.

-2 | A Model with the specified MetadataObjectId does not exist in the specified Metadata partition.

0 | No errors encountered.

-1100 | The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.

A positive integer | A T-SQL error code.

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

#### 3.2.5.42 proc_ar_DeleteParameterById

The proc_ar_DeleteParameterById stored procedure is called to delete the specified Parameter. Parameter MUST be deleted along with its Properties, localized names, and ACEs. After a successful deletion, the OrdinalNumber attribute of all Parameters MUST be normalized for Parameters that are contained by the same Method that contained the deleted Parameter. After normalization, the OrdinalNumber attribute of all these Parameters MUST be renumbered starting from 0, incrementing by 1 and preserving the original order. During this renumbering, the protocol server MUST increment the object version of all these Parameters. After incrementing the object versions, the protocol server MUST set the object version of all these Parameters, whose object version is 2,147,483,646, to 0. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteParameterById (  
    @Id int,  
    @Version int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT
);
```

@Id: The MetadataObjectId of the Parameter. The value MUST be an Id (2.2.1.1).

@Version: The object version of this Parameter.

@PartitionId: The metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).
@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;82&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Parameter with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Parameter. For example, this error can be triggered when a thread reads the given Parameter, after which another thread updates the same Parameter, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-5</td>
<td>The Parameter contains one or more TypeDescriptors.</td>
</tr>
<tr>
<td>-2</td>
<td>A Parameter with the specified MetadataObjectId does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;83&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.43 proc_ar_DeletePropertiesById

The proc_ar_DeletePropertiesById stored procedure is called to delete all Properties contained by the MetadataObject identified by its given MetadataObjectId for a specified Setting. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeletePropertiesById (  
    @MetadataObjectId int,  
    @SettingId nvarchar(128),  
    @ErrorCode int OUTPUT
);
```

@MetadataObjectId: The MetadataObjectId of the MetadataObject that contains the Properties to be deleted. The value MUST be an Id (2.2.1.1).

@SettingId: The Setting to delete the resource from. The value MUST be a SettingId (section 2.2.1.6).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
### Return Values
An integer that MUST be 0.

### Result Sets
MUST NOT return any result sets.

#### 3.2.5.44 proc_ar_DeletePropertyForMetadataObjectId

The **proc_ar_DeletePropertyForMetadataObjectId** stored procedure is called to delete the specified Property contained by the specified MetadataObject. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeletePropertyForMetadataObjectId (
    @MetadataObjectId int,
    @Name nvarchar(255),
    @SettingId nvarchar(128),
    @PartitionId uniqueidentifier,
    @ErrorCode int OUTPUT
);
```

- **@MetadataObjectId**: The MetadataObjectId of the MetadataObject. The value MUST be an *Id* (section 2.2.1.1).
- **@Name**: The name of the Property.
- **@SettingId**: The Setting to delete the Property from. The value MUST be a *SettingId* (section 2.2.1.6).
- **@PartitionId**: The metadata partition of the MetadataObject. The value MUST be a *PartitionId* (section 2.2.1.4).
- **@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;84&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified MetadataObject does not exist in the specified metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY &lt;85&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>
### Return Values:
An integer that MUST be 0.

### Result Sets:
MUST NOT return any result sets.

**3.2.5.45 proc_ar_DeleteSystemById**

The `proc_ar_DeleteSystemById` stored procedure is called to delete the specified LobSystem. The LobSystem MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteSystemById (  
  @Id int,  
  @Version int,  
  @PartitionId uniqueidentifier,  
  @ErrorCode int OUTPUT  
);
```

- **@Id**: The MetadataObjectId of the LobSystem. The value MUST be an Id (2.2.1.1).
- **@Version**: The object version of this LobSystem.
- **@PartitionId**: The metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).
- **@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The LobSystem with the specified MetadataObjectId has been updated by a context other than the one that has been currently read by. This happens when the specified object version is not equal to the current object version of the LobSystem. For example, this error can be triggered when a thread reads the given LobSystem, after which another thread updates the same LobSystem, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-5</td>
<td>The specified LobSystem contains at least one of the following child objects: DataClass or LobSystemInstance.</td>
</tr>
<tr>
<td>-2</td>
<td>The LobSystem with the specified MetadataObjectId does not exist in the specified metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>
### 3.2.5.46 proc_ar_DeleteSystemInstanceById

The `proc_ar_DeleteSystemInstanceById` stored procedure is called to delete the `LobSystemInstance` identified by the specified MetadataObjectId. The `LobSystemInstance` MUST be deleted along with its Properties, localized names, and ACEs. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteSystemInstanceById (
    @Id int,
    @Version int,
    @PartitionId uniqueidentifier,
    @ErrorCode int OUTPUT
);
```

**@Id**: The MetadataObjectId of the `LobSystemInstance`. The value MUST be a `Id` (2.2.1.1).

**@Version**: The object version of this `LobSystemInstance`.

**@PartitionId**: The Metadata partition of the `LobSystemInstance`. Value MUST be a `PartitionId` (section 2.2.1.4).

**@ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The <code>LobSystemInstance</code> with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <code>LobSystemInstance</code>. For example, this error can be triggered when a thread reads the given <code>LobSystemInstance</code>, after which another thread updates the same <code>LobSystemInstance</code>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>The <code>LobSystemInstance</code> with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>positive</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>
Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

### 3.2.5.47 proc_ar_DeleteTypeDescriptorById

The `proc_ar_DeleteTypeDescriptorById` stored procedure is called to delete the TypeDescriptor identified by the specified MetadataObjectId. The **TypeDescriptor** MUST be deleted along with its Properties, localized names, ACEs. All its child **TypeDescriptors** MUST also be deleted recursively. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_DeleteTypeDescriptorById (
  @Id int,
  @Version int,
  @PartitionId uniqueidentifier,
  @ErrorCode int OUTPUT
);
```

**@Id:** The MetadataObjectId of the **TypeDescriptor**. The value MUST be an **Id** (2.2.1.1).

**@Version:** The object version of this **TypeDescriptor**.

**@PartitionId:** The metadata partition of the **TypeDescriptor**. Value MUST be a **PartitionId** (section 2.2.1.4).

**@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The <strong>TypeDescriptor</strong> with the specified MetadataObjectId belongs to an active Entity.</td>
</tr>
<tr>
<td>-6</td>
<td>The <strong>TypeDescriptor</strong> with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>TypeDescriptor</strong>. For example, this error can be triggered when a thread reads the given <strong>TypeDescriptor</strong>, after which another thread updates the same <strong>TypeDescriptor</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-5</td>
<td>A MethodInstance refers to the <strong>TypeDescriptor</strong> with the specified MetadataObjectId as its ReturnTypeDescriptor.</td>
</tr>
<tr>
<td>-2</td>
<td>The <strong>TypeDescriptor</strong> with the specified MetadataObjectId does not exist in the specified metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.48 proc_ar_GetAccessControlEntriesForMetadataObject

The `proc_ar_GetAccessControlEntriesForMetadataObject` stored procedure is called to retrieve all ACEs for the specified MetadataObject. This stored procedure is defined as follows:

```sql
PROCEDURE proc_ar_GetAccessControlEntriesForMetadataObject (  
  @MetadataObjectId int,  
  @SettingId nvarchar(128),  
  @Fallback bit,  
  @ErrorCode int OUTPUT  
);
```

- **@MetadataObjectId:** The MetadataObjectId of the MetadataObject. The value MUST be an `Id` (2.2.1.1).
- **@SettingId:** The Setting to return the ACEs from. Value MUST be a `SettingId` (section 2.2.1.6).
- **@Fallback:** A bit that specifies whether the default Setting MUST be used when ACEs are found for the specified Setting.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>When no ACEs are found for the specified Setting, the stored procedure MUST return a result set with zero rows.</td>
</tr>
<tr>
<td>1</td>
<td>When no ACEs are found for the specified Setting, the stored procedure MUST return the ACEs for the default Setting. If no ACEs are found for the specified Setting and no ACEs are found for the default Setting, the stored procedure MUST return a result set with zero rows.</td>
</tr>
</tbody>
</table>

- **@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The MetadataObject with the specified MetadataObjectId does not exist. The protocol server SHOULD set the error code to -2 when the MetadataObject with the specified MetadataObjectId exists, but not in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:**
If @ErrorCode is set to -2, this stored procedure MUST NOT return any result sets. Otherwise this stored procedure MUST return an Access Control Entry Result Set.

3.2.5.49 proc_ar_GetActionById

The proc_ar_GetActionById stored procedure is called to retrieve the specified Action. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetActionById (  
  @MetadataObjectId int  
  ,@PartitionId uniqueidentifier  
);

@MetadataObjectId: The MetadataObjectId of the Action. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Action. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return an Action Result Set

3.2.5.50 proc_ar_GetActionParameterById

The proc_ar_GetActionParameterById stored procedure is called to retrieve the specified ActionParameter. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetActionParameterById (  
  @MetadataObjectId int  
  ,@PartitionId uniqueidentifier  
);

@MetadataObjectId: The MetadataObjectId of the ActionParameter. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the ActionParameter. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return an Action Parameter Result Set

3.2.5.51 proc_ar_GetActionParametersForActionWithCount

The proc_ar_GetActionParametersForActionWithCount stored procedure is called to retrieve the ActionParameters contained by the specified Action. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetActionParametersForActionWithCount (  
  @ActionId int  
  ,@PartitionId uniqueidentifier  
);
@ActionId: The MetadataObjectId of the Action. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Action. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return an Action Parameter Result Set

3.2.5.52 proc_ar_GetActionsForEntityWithCount

The proc_ar_GetActionsForEntityWithCount stored procedure is called to retrieve the Actions contained by the specified Entity. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetActionsForEntityWithCount (  
    @EntityId int  
    ,@PartitionId uniqueidentifier  
);  

@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return an Action Result Set

3.2.5.53 proc_ar_GetAdministrationMetadataCatalogById

The proc_ar_GetAdministrationMetadataCatalogById stored procedure is called to retrieve the specified MetadataCatalog. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAdministrationMetadataCatalogById (  
    @MetadataObjectId int  
    ,@PartitionId uniqueidentifier  
);  

@MetadataObjectId: The MetadataObjectId of the MetadataCatalog. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the MetadataCatalog. The value MUST be a PartitionId (section 2.2.1.4).
Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a MetadataCatalog Result Set

3.2.5.54  proc_ar_GetAdministrationMetadataCatalogByPartitionId

The proc_ar_GetAdministrationMetadataCatalogByPartitionId stored procedure is called to retrieve the MetadataCatalog for the specified Metadata partition. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAdministrationMetadataCatalogByPartitionId (
    @PartitionId uniqueidentifier
);

@PartitionId: The Metadata partition to return the MetadataCatalog for. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a MetadataCatalog Result Set

3.2.5.55  proc_ar_GetAllLocalizedNamesForMetadataObjectWithCount

The proc_ar_GetAllLocalizedNamesForMetadataObjectWithCount stored procedure is called to retrieve all localized names of the specified MetadataObject for a specified Setting. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAllLocalizedNamesForMetadataObjectWithCount (
    @MetadataObjectId int,
    @SettingId nvarchar(128),
    @PartitionId uniqueidentifier
);

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1).

@SettingId: The Setting from which to return the localized names. The value MUST be a SettingId (section 2.2.1.6).

@PartitionId: The Metadata partition of the MetadataObject. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return a LocalizedName Result Set
3.2.5.56 proc_ar_GetAllMergedLocalizedNamesForMetadataObjectWithCount

The proc_ar_GetAllMergedLocalizedNamesForMetadataObjectWithCount stored procedure is called to retrieve localized names of specified MetadataObject. The stored procedure MUST retrieve all the localized names of the specified MetadataObject in the specified Setting. This stored procedure MUST also retrieve all the localized names of the specified MetadataObject in the default Setting that correspond to a LCID value that is not in the set of LCID values that correspond to localized names of the specified MetadataObject in the specified Setting. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetAllMergedLocalizedNamesForMetadataObjectWithCount (  
  @MetadataObjectId int  
  ,@SettingId nvarchar(128)  
  ,@PartitionId uniqueidentifier  
);  
```

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1).

@SettingId: The Setting from which to return the localized names. The value MUST be a SettingId (section 2.2.1.6).

@PartitionId: The Metadata partition of the MetadataObject. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Count Result Set
This stored procedure MUST return a LocalizedName Result Set

3.2.5.57 proc_ar_GetAllPartitionIds

The proc_ar_GetAllPartitionIds stored procedure is called to retrieve all the distinct PartitionIds (section 2.2.1.4). This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetAllPartitionIds (  
);  
```

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Partition Result Set

3.2.5.58 proc_ar_GetAllSlicesForMetadataObjectId

The proc_ar_GetAllSlicesForMetadataObjectId stored procedure is called to retrieve all the distinct Settings associated with the specified MetadataObject. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetAllSlicesForMetadataObjectId (  
  @MetadataObjectId int  
);  
```
@MetadataObjectId: The MetadataObjectId of the MetadataObject. This value MUST be an Id (2.2.1.1)

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Setting Result Set

3.2.5.59 proc_ar_GetAssociationById

The proc_ar_GetAssociationById stored procedure is called to retrieve the specified Association. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAssociationById (  
    @MetadataObjectId int,  
    @PartitionId uniqueidentifier  
);

@MetadataObjectId: The MetadataObjectId of the Association. The value must be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Association. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return an Association Result Set

3.2.5.60 proc_ar_GetAssociationGroupById

The proc_ar_GetAssociationGroupById stored procedure is called to retrieve the specified AssociationGroup. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAssociationGroupById (  
    @MetadataObjectId int,  
    @PartitionId uniqueidentifier  
);

@MetadataObjectId: The MetadataObjectId of the AssociationGroup. The value must be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the AssociationGroup. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Association Group Result Set
3.2.5.61 proc_ar_GetAssociationGroupsForEntityWithCount

The proc_ar_GetAssociationGroupsForEntityWithCount stored procedure is called to retrieve the count and details of all AssociationGroups contained by the specified Entity. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAssociationGroupsForEntityWithCount (  
  @EntityId int  
  ,@PartitionId uniqueidentifier  
);

@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return an Association Group Result Set

3.2.5.62 proc_ar_GetAssociationMembersInRoleWithCount

The proc_ar_GetAssociationMembersInRoleWithCount stored procedure is called to retrieve the count and details of Association sources or the destination of the specified Association. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAssociationMembersInRoleWithCount (  
  @AssociationId int  
  ,@EntityRole bit  
  ,@PartitionId uniqueidentifier  
);

@AssociationId: MetadataObjectId of the Association. Value MUST be an Id (2.2.1.1).

@EntityRole: A bit specifies whether to return Association sources or the destination of the Association.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Association sources MUST be returned.</td>
</tr>
<tr>
<td>1</td>
<td>Destination of the Association MUST be returned.</td>
</tr>
</tbody>
</table>

@PartitionId: The Metadata partition of the Association. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return an [Association Member Result Set](#).

### 3.2.5.63 proc_ar_GetAssociationReferencesForAssociationGroupWithCount

The `proc_ar_GetAssociationReferencesForAssociationGroupWithCount` stored procedure is called to retrieve the count and details of AssociationReferences contained by the specified AssociationGroup. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetAssociationReferencesForAssociationGroupWithCount (  
  @AssociationGroupId int  
  , @PartitionId uniqueidentifier
);
```

- **@AssociationGroupId**: MetadataObjectId of the AssociationGroup. Value MUST be an Id ([2.2.1.1](#)).
- **@PartitionId**: The Metadata partition of the AssociationGroup. The value MUST be a PartitionId (section [2.2.1.4](#)).

**Return Values**: An integer that MUST be 0.

**Result Sets**:
- This stored procedure MUST return a [Count Result Set](#).
- This stored procedure MUST return a [AssociationReference Result Set](#).

### 3.2.5.64 proc_ar_GetAssociationsForDataClassWithCount

The `proc_ar_GetAssociationsForDataClassWithCount` stored procedure is called to retrieve the count and details of all Associations contained by the specified Entity. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetAssociationsForDataClassWithCount (  
  @ClassId int  
  , @PartitionId uniqueidentifier
);
```

- **@ClassId**: The MetadataObjectId for the Entity. The value MUST be an Id ([2.2.1.1](#)).
- **@PartitionId**: The Metadata partition of the Entity. The value MUST be a PartitionId (section [2.2.1.4](#)).

**Return Values**: An integer that MUST be 0.

**Result Sets**:
- This stored procedure MUST return a [Count Result Set](#).
- This stored procedure MUST return an [Association Result Set](#).
3.2.5.65 proc_ar_GetAssociationsForEntityAndRoleWithCount

The proc_ar_GetAssociationsForEntityAndRoleWithCount stored procedure is called to retrieve the count and details of Associations which reference the specified Entity as an Association source or destination. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetAssociationsForEntityAndRoleWithCount (
    @EntityId int,
    @EntityRole bit,
    @ActiveOnly bit,
    @PartitionId uniqueidentifier);
```

@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@EntityRole: A bit that specifies whether specified Entity represents an Association source or destination. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Association source</td>
</tr>
<tr>
<td>1</td>
<td>Association destination</td>
</tr>
</tbody>
</table>

@ActiveOnly: A bit that specifies whether to include the Associations that reference Entities that are not active in the result. For the purposes of this stored procedure, an Association is considered to reference an Entity when that Entity is a source or the destination of the Association, or when the Entity contains the Association. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Return all Associations that match the search criteria.</td>
</tr>
<tr>
<td>1</td>
<td>Return Associations that match the search criteria only if they do not reference an Entity that is not active.</td>
</tr>
</tbody>
</table>

@PartitionId: The Metadata partition of the Entity. The Value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

When the value of the @ErrorCode parameter is not 0, this stored procedure MUST NOT return any result sets.

When the value of the @ErrorCode parameter is 0 this stored procedure MUST return a Count Result Set.

When the value of the @ErrorCode parameter is 0 this stored procedure MUST return an Association Result Set.
3.2.5.66 proc_ar_GetAssociationsForMethodWithCount

The proc_ar_GetAssociationsForMethodWithCount stored procedure is called to retrieve the count and details of all Associations contained by the specified Method. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetAssociationsForMethodWithCount (  
    @MethodId int, 
    @PartitionId uniqueidentifier
);

@MethodId: The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Count Result Set

This stored procedure MUST return a Association Result Set

3.2.5.67 proc_ar_GetCacheInvalidationCountersWithCount

The proc_ar_GetCacheInvalidationCountersWithCount stored procedure is called to retrieve the current Cache Version Stamp information (section 2.2.2.22) along with the count of Cache Version Stamps. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetCacheInvalidationCountersWithCount (  
    @LastModified bigint
);

@LastModified: The implementation-specific timestamp to compare with the Timestamp attributes of the Cache Version Stamps. This stored procedure MUST only return Cache Version Stamps which have their Timestamp attribute greater than the specified value.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Count Result Set

This stored procedure MUST return a Cache Version Stamps Result Set

3.2.5.68 proc_ar_GetChildTypeDescriptorsForTypeDescriptorWithCount

The proc_ar_GetChildTypeDescriptorsForTypeDescriptorWithCount stored procedure is called to retrieve the count and details of TypeDescriptors which are contained by the specified TypeDescriptor.

PROCEDURE proc_ar_GetChildTypeDescriptorsForTypeDescriptorWithCount (  
    @ParentTypeDescriptorId int, 
    @PartitionId uniqueidentifier
)
@ParentTypeDescriptorId: The MetadataObjectId for the TypeDescriptor. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the TypeDescriptor. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return a TypeDescriptor Result Set

3.2.5.69 proc_ar_GetDataClassById

The proc_ar_GetDataClassById stored procedure is called to retrieve the specified DataClass. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetDataClassById (  
    @MetadataObjectId int  
    ,@PartitionId uniqueidentifier  
);

@MetadataObjectId: The MetadataObjectId of the DataClass. Value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the DataClass. Value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a DataClass Result Set

3.2.5.70 proc_ar_GetDataClassesForSystemWithCount

The proc_ar_GetDataClassesForSystemWithCount stored procedure is called to retrieve the count and details of DataClasses contained by the specified LobSystem. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetDataClassesForSystemWithCount (  
    @SystemId int  
    ,@ActiveOnly bit  
    ,@PartitionId uniqueidentifier  
);

@SystemId: The MetadataObjectId of the LobSystem. The value MUST be an Id (2.2.1.1).

@ActiveOnly: A bit that specifies whether the DataClasses that are not active are to be included in the returned result set or not. The value MUST be listed in the following table.
Value | Description
---|---
0 | All **DataClasses** that are contained by the specified **LobSystem** MUST be returned.
1 | Only the **DataClasses** that are active and contained by the specified **LobSystem** MUST be returned.

@**PartitionId**: The Metadata partition of the **LobSystem**. Value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values**: An integer that MUST be 0.

**Result Sets**:
This stored procedure MUST return a **Count Result Set**
This stored procedure MUST return a **DataClass Result Set**

### 3.2.5.71 proc_ar_GetDefaultValuesForTypeDescriptor

The **proc_ar_GetDefaultValuesForTypeDescriptor** stored procedure is called to retrieve **DefaultValues** (section 2.2.2.17) associated with the specified **TypeDescriptor**. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetDefaultValuesForTypeDescriptor (  @TypeDescriptorId int , @PartitionId uniqueidentifier , @ErrorCode int OUTPUT );
```

@**TypeDescriptorId**: The MetadataObjectId of the **TypeDescriptor** object. The value MUST be an **Id** (2.2.1.1).

@**PartitionId**: The Metadata partition of the **TypeDescriptor**. Value MUST be a **PartitionId** (section 2.2.1.4).

@**ErrorCode**: The error code. Upon return from this stored procedure, this parameter MUST be set by the protocol server to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The specified <strong>TypeDescriptor</strong> does not exist. In this case the result set for this stored procedure MUST contain zero rows.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

**Return Values**: An integer that MUST be 0.

**Result Sets**:
This stored procedure MUST return a **DefaultValues Result Set**
3.2.5.72 proc_ar_GetEntitiesForAssociationAndRoleWithCount

The `proc_ar_GetEntitiesForAssociationAndRoleWithCount` stored procedure is called to retrieve the Entities representing an Association source or destination for the specified Association. This stored procedure is defined as follows.

```c
PROCEDURE proc_ar_GetEntitiesForAssociationAndRoleWithCount (
    @AssociationId int,
    @EntityRole bit,
    @ActiveOnly bit,
    @PartitionId uniqueidentifier
);
```

`@AssociationId`: The MetadataObjectId of the Association. Value MUST be an Id (2.2.1.1).

`@EntityRole`: A bit that specifies whether to return Entities representing an Association source or destination. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Association source</td>
</tr>
<tr>
<td>1</td>
<td>Association dest.</td>
</tr>
</tbody>
</table>

`@ActiveOnly`: A bit that specifies whether the returned Entities are only the active Entities or not. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Return all Entities.</td>
</tr>
<tr>
<td>1</td>
<td>Return only active Entities.</td>
</tr>
</tbody>
</table>

`@PartitionId`: The Metadata partition of the Association. Value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return an Entity Result Set.

3.2.5.73 proc_ar_GetEntitiesForSystemCount

The `proc_ar_GetEntitiesForSystemCount` stored procedure is called to get the number of Entities contained by the specified LobSystem. This stored procedure is defined as follows.

```c
PROCEDURE proc_ar_GetEntitiesForSystemCount (
    @SystemId int,
    @ActiveOnly bit,
    @PartitionId uniqueidentifier
);
```

`@SystemId`: The MetadataObjectId for the LobSystem. The value MUST be an Id (2.2.1.1).
@ActiveOnly: The bit that specifies whether to count Entities that are not active.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This stored procedure MUST return count of all Entities in the LobSystem regardless of the active status of the Entity.</td>
</tr>
<tr>
<td>1</td>
<td>This stored procedure MUST return the count of only active Entities in the LobSystem.</td>
</tr>
</tbody>
</table>

@PartitionId: The Metadata partition of the LobSystem. Value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set

3.2.5.74 proc_ar_GetEntitiesForSystemWithCount

The proc_ar_GetEntitiesForSystemWithCount stored procedure is called to get the Entities contained by the specified LobSystem, along with the count of such Entities. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetEntitiesForSystemWithCount (
    @SystemId int,
    @ActiveOnly bit,
    @PartitionId uniqueidentifier
);
```

@SystemId: The MetadataObjectId of the LobSystem. The value MUST be an Id (2.2.1.1).

@ActiveOnly: A bit that specifies what Entities to be returned. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This stored procedure MUST return Entities regardless of the active status of the Entities.</td>
</tr>
<tr>
<td>1</td>
<td>The stored procedure MUST return only active Entities.</td>
</tr>
</tbody>
</table>

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set

This stored procedure MUST return an Entity Result Set

3.2.5.75 proc_ar_GetEntitiesLikeNameAndNamespace

The proc_ar_GetEntitiesLikeNameAndNamespace stored procedure is called to retrieve Entities whose attributes match the specified patterns. This stored procedure is defined as follows.
PROCEDURE proc_ar_GetEntitiesLikeNameAndNamespace (  
    @WildcardedNamespace nvarchar(255) 
    ,@WildcardedName nvarchar(255) 
    ,@LCID int 
    ,@ActiveOnly bit 
    ,@PartitionId uniqueidentifier 
);  

@WildcardedNamespace: A string that specifies a pattern for the Namespace (section 2.2.1.3) of the Entities. The protocol server MUST match the pattern against the namespaces of the Entities in the metadata store as specified for the LIKE operator in [MSDN-TSQL-Ref] and only return those Entities whose namespaces match. For example, setting the @WildcardedNamespace as "A%" will make this stored procedure return only the Entities with Namespace starting with either "A" or "a".

@WildcardedName: A string that specifies a pattern for the name or the localized name of the Entities. The protocol server MUST match the pattern against the names and localized names of the Entities in the metadata store as specified for the LIKE operator in [MSDN-TSQL-Ref] and only return those Entities whose names or localized names match. If it is only the localized name that matches this parameter, the LCID of the localized name MUST be the specified LCID or 0. For example, setting the @WildcardedName as "A%" will make this stored procedure return only the Entities with names starting with either "A" or "a".

@LCID: The LCID used to restrict which localized names of the Entities to consider.

@ActiveOnly: A bit that specifies whether the Entities to be returned are only active Entities. The value MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This stored procedure MUST return Entities regardless of the active status of the Entities.</td>
</tr>
<tr>
<td>1</td>
<td>This stored procedure MUST return only Entities whose status is active.</td>
</tr>
</tbody>
</table>

@PartitionId: The metadata partition to return the results from. The value MUST be a PartitionId (section 2.2.1.4). This stored procedure MUST only return Entities whose PartitionId is equal to this value.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Count Result Set

This stored procedure MUST return an Entity Result Set

3.2.5.76 proc_ar_GetEntitiesReferencedByModelId

The proc_ar_GetEntitiesReferencedByModelId stored procedure is called to retrieve the Entities that are referenced by the specified Model. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetEntitiesReferencedByModelId (  
    @MetadataObjectId int 
    ,@Mode tinyint 
    ,@ActiveOnly bit 
);
@MetadataObjectId: The MetadataObjectId of the Model. This value MUST be an Id (2.2.1.1).

@Mode: Specifies which Entities to be returned. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Return all Entities referenced by the specified Model.</td>
</tr>
<tr>
<td>1</td>
<td>Return all Entities referenced in the specified Model and not referenced by any other Model.</td>
</tr>
<tr>
<td>2</td>
<td>Return all Entities referenced in the specified Model and referenced by at least one other Model.</td>
</tr>
</tbody>
</table>

@ActiveOnly: A bit that specifies whether the returned Entities are only the active Entities or not. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Return all Entities.</td>
</tr>
<tr>
<td>1</td>
<td>Return only Entities that are active.</td>
</tr>
</tbody>
</table>

@PartitionId: The Metadata partition of the Model. Value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return an Entity Result Set

3.2.577 proc_ar_GetEntityById

The proc_ar_GetEntityById stored procedure is called to retrieve the specified Entity. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetEntityById (
    @MetadataObjectId int,
    ,@PartitionId uniqueidentifier
);
```

@MetadataObjectId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return an Entity Result Set

---

[MS-BDCDPS2] — v20120630

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
3.2.5.78 proc_ar_GetEntityNamesForAssociationAndRole

The **proc_ar_GetEntityNamesForAssociationAndRole** stored procedure is called to retrieve the name and namespace of the Association sources and the destination of the specified Association. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetEntityNamesForAssociationAndRole (  
  @AssociationId int  
  ,@EntityRole bit  
  ,@PartitionId uniqueidentifier  
);

@**AssociationId**: The MetadataObjectId of the Association. Value MUST be an **Id** (2.2.1.1).

@**EntityRole**: A bit that specifies whether to return Entities representing an Association source or destination. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Association source</td>
</tr>
<tr>
<td>1</td>
<td>Association destination</td>
</tr>
</tbody>
</table>

@**PartitionId**: The Metadata partition of the Association. The Value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values**: An integer that MUST be 0.

**Result Sets**:

When the value of **@ErrorCode** parameter is not 0, this stored procedure MUST NOT return any result sets.

Otherwise, this stored procedure MUST return an **Entity Name Result Set**

3.2.5.79 proc_ar_GetEntityWithNameAndNamespace

The **proc_ar_GetEntityWithNameAndNamespace** stored procedure is called to retrieve the active version of the specified Entity. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetEntityWithNameAndNamespace (  
  @Namespace nvarchar(255)  
  ,@Name nvarchar(255)  
  ,@PartitionId uniqueidentifier  
);

@**Namespace**: The namespace of the Entity. The value MUST be a **Namespace** (section 2.2.1.3).

@**Name**: The name of the Entity. The value MUST be a **Name** (section 2.2.1.2).

@**PartitionId**: The metadata partition of the Entity. The value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values**: An integer that MUST be 0.
Result Sets:
This stored procedure MUST return an Entity Result Set

3.2.5.80 proc_ar_GetEntityWithNameAndNamespaceAndVersion

The **proc_ar_GetEntityWithNameAndNamespaceAndVersion** stored procedure is called to retrieve the specified Entity. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetEntityWithNameAndNamespaceAndVersion ( 
  @Namespace nvarchar(255)
  ,@Name nvarchar(255)
  ,@MajorVersion int
  ,@MinorVersion int
  ,@BuildVersion int
  ,@RevisionVersion int
  ,@PartitionId uniqueidentifier
);
```

**@Namespace:** The namespace of the **Entity**. The value MUST be a **Namespace** (section 2.2.1.3).

**@Name:** The name of the **Entity**. The value MUST be a **Name** (section 2.2.1.2).

**@MajorVersion:** The major version of the **Entity**. The value MUST be a **MajorVersion** (section 2.2.1.7).

**@MinorVersion:** The minor version of the **Entity**. The value MUST be a **MinorVersion** (section 2.2.1.8).

**@BuildVersion:** The build version of the **Entity**. The value MUST be a **BuildVersion** (section 2.2.1.9).

**@RevisionVersion:** The revision version of the **Entity**. The value MUST be a **RevisionVersion** (section 2.2.1.10).

**@PartitionId:** The Metadata partition of the **Entity**. The value MUST be a **PartitionId** (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return an Entity Result Set

3.2.5.81 proc_ar_GetFilterDescriptorById

The **proc_ar_GetFilterDescriptorById** stored procedure is called to retrieve the specified FilterDescriptor. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetFilterDescriptorById ( 
  @MetadataObjectId int
  ,@PartitionId uniqueidentifier
);
```

[MS-BDCDPS2] — v20120630

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
@MetadataObjectId: The MetadataObjectId for the FilterDescriptor. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the FilterDescriptor. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a FilterDescriptor Result Set

3.2.5.82 proc_ar_GetFilterDescriptorsForMethodWithCount

The proc_ar_GetFilterDescriptorsForMethodWithCount stored procedure is called to retrieve the FilterDescriptors contained by the specified Method, along with the count of such FilterDescriptors. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetFilterDescriptorsForMethodWithCount (  
    @MethodId int  
    ,@PartitionId uniqueidentifier
);

@MethodId: The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return a FilterDescriptor Result Set

3.2.5.83 proc_ar_GetIdentifierById

The proc_ar_GetIdentifierById stored procedure is called to retrieve the specified Identifier. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetIdentifierById (  
    @MetadataObjectId int  
    ,@PartitionId uniqueidentifier
);

@MetadataObjectId: The MetadataObjectId of the Identifier. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Identifier. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return an Identifier Result Set
3.2.5.84 proc_ar_GetIdentifiersForEntityWithCount

The `proc_ar_GetIdentifiersForEntityWithCount` stored procedure is called to retrieve the Identifiers contained by the specified Entity, along with the count of such Identifiers. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetIdentifiersForEntityWithCount (  
  @EntityId int,
  @PartitionId uniqueidentifier
);
```

@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Count Result Set

This stored procedure MUST return an Identifier Result Set

3.2.5.85 proc_ar_GetMergedPropertiesForMetadataObject

The `proc_ar_GetMergedPropertiesForMetadataObject` stored procedure is called to retrieve Properties for the specified MetadataObject. The stored procedure MUST retrieve all the Properties of the specified MetadataObject in the specified Setting. This stored procedure MUST also retrieve all the Properties of the specified MetadataObject in the default Setting that names that is not in the set of name of the Properties of the specified MetadataObject in the specified Setting. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetMergedPropertiesForMetadataObject (  
  @MetadataObjectId int,
  @SettingId nvarchar(128),
  @PartitionId uniqueidentifier,
  @ErrorCode int OUTPUT
);
```

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1).

@SettingId: The Setting from which to return the Properties. The value MUST be a SettingId (section 2.2.1.6).

@PartitionId: The Metadata partition of the MetadataObject. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, the parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The specified MetadataObject does not exist. In this case the result set for this stored</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>procedure MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a Property Result Set

### 3.2.5.86 proc_ar_GetMethodById

The `proc_ar_GetMethodById` stored procedure is called to retrieve the specified Method. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetMethodById (  
    @MetadataObjectId int,
    @PartitionId uniqueidentifier
);
```

@`MetadataObjectId`: The `MetadataObjectId` of the Method. The value MUST be an Id (2.2.1.1).

@`PartitionId`: The metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a Method Result Set

### 3.2.5.87 proc_ar_GetMethodInstanceById

The `proc_ar_GetMethodInstanceById` stored procedure is called to retrieve the specified MethodInstance. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetMethodInstanceById (  
    @MetadataObjectId int,
    @PartitionId uniqueidentifier
);
```

@`MetadataObjectId`: The `MetadataObjectId` of the MethodInstance. The value MUST be an Id (2.2.1.1).

@`PartitionId`: The Metadata partition of the MethodInstance. The value MUST be a PartitionId (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a MethodInstance Result Set
3.2.5.88 proc_ar_GetMethodInstancesForDataClassWithCount

The **proc_ar_GetMethodInstancesForDataClassWithCount** stored procedure is called to retrieve the MethodInstances that are contained by the specified DataClass, excluding those MethodInstances that are Associations, along with the count of such MethodInstances. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetMethodInstancesForDataClassWithCount (  @ClassId int  ,@PartitionId uniqueidentifier  );
```

@ClassId: The MetadataObjectId of the **DataClass**. The value MUST be an **Id** (2.2.1.1).

@PartitionId: The metadata partition of the **DataClass**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**
- This stored procedure MUST return a **Count Result Set**
- This stored procedure MUST return a **MethodInstance Result Set**

3.2.5.89 proc_ar_GetMethodInstancesForMethodWithCount

The **proc_ar_GetMethodInstancesForMethodWithCount** stored procedure is called to retrieve the count and details of all MethodInstances contained by the specified Method. The MethodInstances that are Associations MUST NOT be returned. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetMethodInstancesForMethodWithCount (  @MethodId int  ,@PartitionId uniqueidentifier  );
```

@MethodId: The MetadataObjectId of the **Method**. The value MUST be an **Id** (2.2.1.1).

@PartitionId: The Metadata partition of the **Method**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**
- This stored procedure MUST return a **Count Result Set**
- This stored procedure MUST return a **MethodInstance Result Set**

3.2.5.90 proc_ar_GetMethodsForDataClassWithCount

The **proc_ar_GetMethodsForDataClassWithCount** stored procedure is called to retrieve the count and details of all Methods contained by the specified DataClass. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetMethodsForDataClassWithCount (  @ClassId int  ,@PartitionId uniqueidentifier  );
```

@ClassId: The MetadataObjectId of the **DataClass**. The value MUST be an **Id** (2.2.1.1).

@PartitionId: The metadata partition of the **DataClass**. The value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**
- This stored procedure MUST return a **Count Result Set**
- This stored procedure MUST return a **MethodInstance Result Set**
PROCEDURE proc_ar_GetMethodsForDataClassWithCount (
    @ClassId int,
    @PartitionId uniqueidentifier
);

@ClassId: The MetadataObjectId of the DataClass. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the DataClass. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return a Method Result Set

3.2.5.91 proc_ar_GetModelById

The proc_ar_GetModelById stored procedure is called to retrieve the specified Model. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetModelById ( 
    @MetadataObjectId int
    ,@PartitionId uniqueidentifier
);

@MetadataObjectId: The MetadataObjectId of the Model. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Model. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Model Result Set

3.2.5.92 proc_ar_GetModelsByEntityId

The proc_ar_GetModelsByEntityId stored procedure is called to retrieve the Models referencing the specified Entity. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetModelsByEntityId ( 
    @MetadataObjectId int
    ,@PartitionId uniqueidentifier
);

@MetadataObjectId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).
**Return Values:** An integer that MUST be 0.

**Result Sets:**
This stored procedure MUST return a Model Result Set

3.2.5.93 proc_ar_GetModelsByName

The proc_ar_GetModelsByName stored procedure is called to retrieve a set of Models. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetModelsByName (  
    @ModelName nvarchar(255)  
    ,@UseWildcard bit  
    ,@LCID int  
    ,@PartitionId uniqueidentifier  
);
```

@ModelName: A string including either the exact name or a wildcard pattern of the Models to be returned. If this parameter is a wildcard pattern, then the @UseWildcard parameter MUST be set to 1. Otherwise, @UseWildcard parameter MUST be set to 0.

@UseWildcard: A bit indicating whether the @ModelName parameter is using wildcards.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The stored procedure MUST return a Model whose name attribute is equal to the @ModelName parameter. The LCID MUST be ignored.</td>
</tr>
<tr>
<td>1</td>
<td>The stored procedure MUST match the pattern specified by @ModelName against the names and localized names of the Models in the metadata store as specified for the LIKE operator in [MSDN-TSQL-Ref] and only return those Models whose names or localized names match. If it is only the localized name that matches this parameter, the LCID of the localized name MUST be the specified LCID.</td>
</tr>
</tbody>
</table>

@LCID: The LCID to use when retrieving the Models when @UseWildcard is set to one. The value MUST be ignored if @UseWildcard is set to zero.

@PartitionId: The Metadata partition to return the results from. The value MUST be a PartitionId (section 2.2.1.4). This stored procedure MUST only return MetadataObjects whose PartitionId match this value.

**Return Values:** An integer that MUST be 0.

**Result Sets:**
This stored procedure MUST return a Model Result Set

3.2.5.94 proc_ar_GetParameterById

The proc_ar_GetParameterById stored procedure is called to retrieve the specified Parameter. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetParameterById (  
    @MetadataObjectId int  
    ,@PartitionId uniqueidentifier  
);
```
@MetadataObjectId: The MetadataObjectId of the Parameter. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Parameter Result Set

3.2.5.95 proc_ar_GetParametersForMethodWithCount

The proc_ar_GetParametersForMethodWithCount stored procedure is called to retrieve Parameter information for the specified Method, along with the count of the retrieved Parameters. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetParametersForMethodWithCount (
    @MethodId int,
    @PartitionId uniqueidentifier
);

@MethodId: The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set
This stored procedure MUST return a Parameter Result Set

3.2.5.96 proc_ar_GetPropertiesForMetadataObject

The proc_ar_GetPropertiesForMetadataObject stored procedure is called to retrieve Properties for the specified MetadataObject for the specified Setting. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetPropertiesForMetadataObject (
    @MetadataObjectId int,
    @SettingId nvarchar(128),
    @PartitionId uniqueidentifier,
    ErrorCode int OUTPUT
);

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1).
@SettingId: The **Setting** to return the **Properties** from. The value MUST be a **SettingId** (section 2.2.1.6).

@PartitionId: The metadata partition of the **MetadataObject**. The value MUST be a **PartitionId** (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The specified <strong>MetadataObject</strong> does not exist. In this case the result set for this stored procedure MUST be ignored by the protocol client.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a **Property Result Set**.

3.2.5.97 **proc_ar_GetRootTypeDescriptorForParameter**

The **proc_ar_GetRootTypeDescriptorForParameter** stored procedure is called to retrieve the root TypeDescriptor of the specified Parameter. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetRootTypeDescriptorForParameter (  
    @MetadataObjectId int,  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT  
);  
```

@MetadataObjectId: The **MetadataObjectId** of the **Parameter**. The value MUST be an **Id** (2.2.1.1).

@PartitionId: The metadata partition of the **Parameter**. The value MUST be a **PartitionId** (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The specified <strong>Parameter</strong> does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:**

When the value of the **@ErrorCode** parameter is 0 this stored procedure MUST return a **TypeDescriptor Result Set**. Otherwise, this stored procedure MUST NOT return any result sets.
3.2.5.98 proc_ar_GetSafetyNetConfigs

The proc_ar_GetSafetyNetConfigs stored procedure is called to retrieve all Throttle Configuration Settings (section 2.2.2.23) available in the metadata store. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetSafetyNetConfigs ( )
```

**Return Values:** An integer that MUST be 0.

**Result Sets:**
This stored procedure MUST return a Throttle Setting Result Set

3.2.5.99 proc_ar_GetSystemById

The proc_ar_GetSystemById stored procedure is called to retrieve the specified LobSystem. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetSystemById ( @MetadataObjectId int ,@PartitionId uniqueidentifier )
```

@MetadataObjectId: The MetadataObjectId of the LobSystem. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**
This stored procedure MUST return a System Result Set

3.2.5.100 proc_ar_GetSystemByName

The proc_ar_GetSystemByName stored procedure is called to retrieve the specified LobSystem. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetSystemByName ( @Name nvarchar(255) ,@PartitionId uniqueidentifier )
```

@Name: The name of the LobSystem. The value MUST be a Name (section 2.2.1.2).

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.
Result Sets:
This stored procedure MUST return a System Result Set

3.2.5.101  proc_ar_GetSystemDataBySystemId

The proc_ar_GetSystemDataBySystemId stored procedure is called to retrieve SystemData (section 2.2.1.31) associated with the specified LobSystem. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetSystemDataBySystemId (  
  @SystemId int,  
  @PartitionId uniqueidentifier  
);
```

@SystemId: The MetadataObjectId for the LobSystem. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a System Data Result Set

3.2.5.102  proc_ar_GetSystemForParameterId

The proc_ar_GetSystemForParameterId stored procedure is called to retrieve the LobSystem that contains the DataClass containing the Method that contains the specified Parameter. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetSystemForParameterId (  
  @MetadataObjectId nvarchar(255),  
  @PartitionId uniqueidentifier  
);
```

@MetadataObjectId: The MetadataObjectId of the Parameter. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a System Result Set

3.2.5.103  proc_ar_GetSystemForTypeDescriptorId

The proc_ar_GetSystemForTypeDescriptorId stored procedure is called to retrieve the LobSystem that contains the DataClass containing the Method that contains the Parameter that contains the specified TypeDescriptor. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_GetSystemForTypeDescriptorId (  
);
```
@MetadataObjectId nvarchar(255), @PartitionId uniqueidentifier
);

@MetadataObjectId: The MetadataObjectId of the TypeDescriptor. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the TypeDescriptor. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a System Result Set

3.2.5.104  proc_ar_GetSystemInstanceById

The proc_ar_GetSystemInstanceById stored procedure is called to retrieve the specified 
LobSystemInstance. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetSystemInstanceById ( 
    @MetadataObjectId int,
    @PartitionId uniqueidentifier
);

@MetadataObjectId: The MetadataObjectId of the LobSystemInstance. The value MUST be an Id (2.2.1.1).

@PartitionId: The Metadata partition of the LobSystemInstance. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a SystemInstance Result Set

3.2.5.105  proc_ar_GetSystemInstancesForSystemWithCount

The proc_ar_GetSystemInstancesForSystemWithCount stored procedure is called to retrieve 
LobSystemInstances contained by the specified LobSystem, along with the count of the retrieved 
LobSystemInstances. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetSystemInstancesForSystemWithCount ( 
    @SystemId int,
    @PartitionId uniqueidentifier
);

@SystemId: The MetadataObjectId of the LobSystem. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the LobSystem. Value MUST be a PartitionId (section 2.2.1.4).
**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a **Count Result Set**

This stored procedure MUST return a **SystemInstance Result Set**

---

### 3.2.5.106 proc_ar_GetSystemsLikeNameWithCount

The **proc_ar_GetSystemsLikeNameWithCount** stored procedure is called to retrieve a set of LobSystems, along with the count of the retrieved LobSystems. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetSystemsLikeNameWithCount (
    @MetadataObjectName nvarchar(255)
    ,@LCID int
    ,@PartitionId uniqueidentifier
);
```

**@MetadataObjectName:** A string that specifies a pattern for the name or the localized name of the LobSystems. The protocol server MUST match the pattern against the names and localized names of the LobSystems in the metadata store as specified for the LIKE operator in [MSDN-TSQL-Ref] and only return those LobSystems whose names or localized names match. If it is only the localized name that matches this parameter, the LCID of the localized name MUST be the specified LCID.

**@LCID:** The LCID of the localized names of the LobSystems.

**@PartitionId:** The Metadata partition to return the results from. Value MUST be a PartitionId (section 2.2.1.4). This stored procedure MUST only return LobSystems whose PartitionId is equal to this value.

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a **Count Result Set**

This stored procedure MUST return a **System Result Set**

---

### 3.2.5.107 proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId

The **proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId** stored procedure is called to retrieve the LobSystems which contain at least one Entity that is referenced by the specified Model. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId (
    @MetadataObjectId int
    ,@Mode tinyint
    ,@PartitionId uniqueidentifier
);
```

**@MetadataObjectId:** The MetadataObjectId of the Model. The value MUST be a Id (2.2.1.1)
@Mode: Specifies which **LobSystems** to be returned. The value of this parameter MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Return all <strong>LobSystems</strong> containing <strong>Entities</strong> referenced by the specified <strong>Model</strong>.</td>
</tr>
<tr>
<td>1</td>
<td>Return all <strong>LobSystems</strong> containing <strong>Entities</strong> referenced by the specified <strong>Model</strong>, but are not referenced by any other <strong>Model</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>Return all <strong>LobSystems</strong> containing <strong>Entities</strong> referenced in the specified <strong>Model</strong> and also referenced by at least one other <strong>Model</strong>.</td>
</tr>
</tbody>
</table>

@PartitionId: The Metadata partition of the **Model**. The Value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a **System Result Set**

### 3.2.5.108 proc_ar_GetTypeDescriptorById

The **proc_ar_GetTypeDescriptorById** stored procedure is called to retrieve the specified TypeDescriptor. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetTypeDescriptorById ( 
  @MetadataObjectId int 
  ,@PartitionId uniqueidentifier 
);
```

@MetadataObjectId: The MetadataObjectId of the **TypeDescriptor**. The value MUST be **Id** (2.2.1.1).

@PartitionId: The metadata partition of the **TypeDescriptor**. Value MUST be a **PartitionId** (section 2.2.1.4).

**Return Values:** An integer that MUST be 0.

**Result Sets:**

This stored procedure MUST return a **TypeDescriptor Result Set**

### 3.2.5.109 proc_ar_GetTypeDescriptorsByNameAndParameter

The **proc_ar_GetTypeDescriptorsByNameAndParameter** stored procedure is called to retrieve TypeDescriptors which have the specified name and are contained by the specified Parameter. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetTypeDescriptorsByNameAndParameter ( 
  @MetadataObjectId int 
  ,@Name nvarchar(255) 
  ,@PartitionId uniqueidentifier 
);
```
@MetadataObjectId: The MetadataObjectId of an existing Parameter. The value MUST be an Id (2.2.1.1).

@Name: The name of the TypeDescriptor. The value MUST be a Name (section 2.2.1.2).

@PartitionId: The Metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
When the value of the ErrorCode parameter is not 0, this stored procedure MUST NOT return any result sets. Otherwise, this stored procedure MUST return a TypeDescriptor Result Set

3.2.5.110 proc_ar_GetTypeDescriptorsForFilterDescriptorWithCount

The proc_ar_GetTypeDescriptorsForFilterDescriptorWithCount stored procedure is called to retrieve the count and the details of TypeDescriptors that reference the specified FilterDescriptor. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetTypeDescriptorsForFilterDescriptorWithCount (  
@FilterDescriptorId int
,  
@PartitionId uniqueidentifier
)

@FilterDescriptorId: The MetadataObjectId of the FilterDescriptor. The value MUST be an Id (2.2.1.1)

@PartitionId: The Metadata partition of the FilterDescriptor. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return a Count Result Set.
This stored procedure MUST return a TypeDescriptor Result Set.

3.2.5.111 proc_ar_GetViewByMethodInstance

The proc_ar_GetViewByMethodInstance stored procedure is called to retrieve a View of the MethodInstance with the name that is contained in the specified DataClass. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetViewByMethodInstance (  
@EntityId int
,  
@MethodInstanceName nvarchar(255)
,  
@PartitionId uniqueidentifier
,  
@ErrorCode int OUTPUT
)

@EntityId: The MetadataObjectId of the DataClass. The value MUST be an Id (2.2.1.1).
@MethodInstanceName: The name of the MethodInstance. The value MUST be a Name (section 2.2.1.2).

@PartitionId: The metadata partition of the DataClass. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error encountered.</td>
</tr>
<tr>
<td>-200</td>
<td>The specified MethodInstance or the specified DataClass does not exist.</td>
</tr>
<tr>
<td>-201</td>
<td>The specified MethodInstance has a MethodInstanceType (section 2.2.1.23) that does not have a View.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets:
When the value of the @ErrorCode parameter is not 0, this stored procedure MUST NOT return any result sets.
Otherwise, this stored procedure MUST return a TypeDescriptor Result Set.

3.2.5.112 proc_ar_IsMethodInstantiated
The proc_ar_IsMethodInstantiated stored procedure is called to get the MetadataObjectId of any MethodInstance contained by the specified Method, determined with an implementation-specific algorithm. This stored procedure is defined as follows.

PROCEDURE proc_ar_IsMethodInstantiated ( @MetadataObjectId int ,@PartitionId uniqueidentifier );

@MetadataObjectId: The MetadataObjectId of the Method. The value MUST be an Id (2.2.1.1).

@PartitionId: The metadata partition of the Method. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be 0.

Result Sets:
This stored procedure MUST return an Id Result Set.

3.2.5.113 proc_ar_IsParameterReferencedByMethodInstance
The proc_ar_IsParameterReferencedByMethodInstance stored procedure is called to return the MethodInstances which return the specified Parameter. This stored procedure is defined as follows.

PROCEDURE proc_ar_IsParameterReferencedByMethodInstance (}
@MetadataObjectId: The MetadataObjectId of the Parameter. The value MUST be an Id.

@PartitionId: The Metadata partition of the Parameter. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The specified Parameter does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets:

When the value of @ErrorCode parameter is not 0, this stored procedure MUST NOT return any result sets.

Otherwise, this stored procedure MUST return an Id Result Set<94>

3.2.5.114 proc_ar_RemoveEntity

The proc_ar_RemoveEntity stored procedure is called to remove the reference to the specified Entity from the specified Model. This stored procedure is defined as follows.

PROCEDURE proc_ar_RemoveEntity (  
    @ModelId int
    ,@ClassId int
    ,@ErrorCode int OUTPUT
);  

@ModelId: The MetadataObjectId of the Model. The value MUST be an Id (section 2.2.1.1).

@ClassId: The MetadataObjectId of the Entity. The value MUST be an Id.

@ErrorCode: The error code. Upon return from this stored procedure, the parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;95&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-2</td>
<td>Any of the following conditions are true:</td>
</tr>
<tr>
<td></td>
<td>- An Entity with the specified MetadataObjectId does not exist in the specified</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>metadata partition.</td>
</tr>
<tr>
<td></td>
<td>- A Model with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td></td>
<td>- The specified Model does not reference the specified Entity.</td>
</tr>
<tr>
<td>0</td>
<td>No error encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

3.2.5.115 proc_ar_RemoveSafetyNetConfig

The proc_ar_RemoveSafetyNetConfig stored procedure is called to delete the specified Throttle Configuration Setting (section 2.2.2.23) from the metadata store. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_RemoveSafetyNetConfig (
  @ThrottleScope int,
  @ThrottleType int,
  @ProxyId uniqueidentifier
);
```

@ThrottleScope: The scope of the setting to be deleted. The value MUST be a ThrottleScope (section 2.2.1.38).

@ThrottleType: The type of the setting to be deleted. The value MUST be an ThrottleType (section 2.2.1.39).

@ProxyId: The implementation-specific partition associated with the setting to be deleted.

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

3.2.5.116 proc_ar_RetrieveProgress

The proc_ar_RetrieveProgress stored procedure is called to retrieve the progress of an operation represented by the specified identifier, updated by the proc_ar_UpdateProgress (section 3.2.5.133) stored procedure. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_RetrieveProgress (  
  @PartitionId uniqueidentifier
,  @JobKey uniqueidentifier
);  
```
@PartitionId: The metadata partition associated with the operation. The value MUST be a PartitionId (section 2.2.1.4).

@JobKey: The identifier of the operation. The value MUST be a GUID.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Progress Result Set

3.2.5.117 proc_ar_SetAccessControlEntryForMetadataObject

The proc_ar_SetAccessControlEntryForMetadataObject stored procedure is called to add an ACE to the specified MetadataObject for the specified Setting. If an ACE with the specified name of the security principal already exists, it is replaced by the newly created ACE. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_SetAccessControlEntryForMetadataObject (
    @MetadataObjectId int,
    @IdentityName nvarchar(250),
    @DisplayName nvarchar(250),
    @RawSid varbinary(512),
    @Rights bigint,
    @SettingId nvarchar(128));
```

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1).

@IdentityName: The name of the security principal (2).

@DisplayName: The name of the security principal (2) used for display purposes.

@RawSid: The value must be NULL.

@Rights: The permissions available to the security principal (2) for the MetadataObject identified by the MetadataObjectId. The value MUST be MetadataRights (section 2.2.1.32).

@SettingId: The Setting to which to write the ACE. The value MUST be a SettingId (section 2.2.1.6).

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.118 proc_ar_SetDefaultAction

The proc_ar_SetDefaultAction stored procedure is called to set or clear the default Action on the specified Entity. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_SetDefaultAction ( @EntityId int
```
@EntityId: The MetadataObjectId of the Entity. The value MUST be an Id (2.2.1.1).

@ActionName: The name of the Action or NULL. If the value is NULL this stored procedure MUST clear the default Action for the specified Entity. Otherwise the value MUST be a Name (section 2.2.1.2), and this stored procedure MUST set the Action with the specified name contained by the specified Entity as the default Action for the specified Entity.

@PartitionId: The metadata partition of the Entity. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The value of the @ActionName parameter is not NULL, and the specified Entity does not contain an Action with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.119 proc_ar_SetDefaultValuesForTypeDescriptor

The proc_ar_SetDefaultValuesForTypeDescriptor stored procedure is called to set the DefaultValue (section 2.2.2.17) of the specified TypeDescriptor for the specified MethodInstance. This stored procedure is defined as follows.

PROCEDURE proc_ar_SetDefaultValuesForTypeDescriptor (  
  @TypeDescriptorId int  
  ,@MethodInstanceId int  
  ,@PartitionId uniqueidentifier  
  ,@Value sql_variant  
  ,@ErrorCode int OUTPUT  
 );

@TypeDescriptorId: The MetadataObjectId of the TypeDescriptor. The value MUST be an Id (2.2.1.1)
@MethodInstanceId: The MetadataObjectId of the MethodInstance. The value MUST be an Id.

@PartitionId: The Metadata partition of the TypeDescriptor and the MethodInstance. The value MUST be a PartitionId (section 2.2.1.4).

@Value: The implementation-specific representation of the DefaultValue. The value MUST be a DefaultValue.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-600</td>
<td>The Parameter of the specified TypeDescriptor is not contained by the same Method as the Method of the specified MethodInstance.</td>
</tr>
<tr>
<td>-3</td>
<td>The specified TypeDescriptor already has implementation-specific maximum number of DefaultValues.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified TypeDescriptor or the specified MethodInstance does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>99</td>
<td>A positive integer</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.120 proc_ar_SetSafetyNetConfig

The proc_ar_SetSafetyNetConfig stored procedure is called to create a Throttle Configuration Setting (section 2.2.2.23) in the metadata store. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_SetSafetyNetConfig (  
    @ThrottleScope int,  
    @ThrottleType int,  
    @MaxValue int,  
    @DefaultValue int,  
    @Enabled bit,  
    @ProxyId uniqueidentifier);  
```

@ThrottleScope: The scope of the setting. The value MUST be a ThrottleScope (section 2.2.1.38).

@ThrottleType: The type of setting. The value MUST be a ThrottleType (section 2.2.1.39).

@MaxValue: The maximum level to which the setting can be increased.

@DefaultValue: The default level of the setting.
@Enabled: A bit that specifies whether the setting is enabled. The value MUST be a ThrottleConfigEnabled (section 2.2.1.40).

@ProxyId: The implementation-specific value a protocol client uses to specify the partition associated with the setting to be created.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.121 proc_ar_SetSystemDataBySystemId

The proc_ar_SetSystemDataBySystemId stored procedure is called to set the SystemData (section 2.2.1.31) associated with the specified LobSystem. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_SetSystemDataBySystemId (  
  @SystemId int,  
  @AssemblyName nvarchar(255),  
  @Length int,  
  @Data image,  
  @PartitionId uniqueidentifier  
);
```

@SystemId: The MetadataObjectId for the LobSystem. The value MUST be an Id (2.2.1.1).

@AssemblyName: The identifier for the SystemData.

@Length: Size of the SystemData, in bytes.

@Data: The data associated with the LobSystem. The value MUST be a SystemData.

@PartitionId: The Metadata partition of the LobSystem. The value MUST be a PartitionId (section 2.2.1.4).

Return Values: An integer that MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | One of the following conditions is true:  
- The value of at least one of @AssemblyName, @Length, or @Data parameter is NULL.  
- A LobSystem with the specified MetadataObjectId does not exist in the specified Metadata partition. |
| 1     | No errors encountered. |

Result Sets: MUST NOT return any result sets.

3.2.5.122 proc_ar_UpdateActionById

The proc_ar_UpdateActionById stored procedure is called to change the attributes of the Action identified by the specified MetadataObjectId. This stored procedure is defined as follows.
PROCEDURE proc_ar_UpdateActionById (
    @Id int,
    @Name nvarchar(50),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @Version int OUTPUT,OUTPUT
    @Position int,
    @IsDisplayed bit,
    @IsOpenedInNewWindow bit,
    @Icon nvarchar(2080),
    @Url nvarchar(2080),
    @ErrorCode int OUTPUT
);

@Id: The MetadataObjectId of the Action that is to be updated. The value MUST be an Id (2.2.1.1).

@Name: The name of the Action. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this Action is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the Action to update. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the Action. The protocol client MUST set the value to the object version of the Action at the time the Action was last read by the protocol client. The protocol server MUST increment the object version of the Action upon successful execution of this stored procedure. If the incremented object version of the Action is equal to 2147483646, the protocol server MUST set the object version of the Action to 0. The protocol server MUST return the object version of the Action on output.

@Position: The Position attribute of the Action. The value MUST be a Position (section 2.2.1.14).

@IsDisplayed: The IsDisplayed attribute of the Action. The value MUST be an IsDisplayed (section 2.2.1.15).

@IsOpenedInNewWindow: The IsOpenedInNewWindow attribute of the Action. The value MUST be an IsOpenedInNewWindow (section 2.2.1.16).

@Icon: The Icon attribute of the Action. The value MUST be an Icon (section 2.2.1.17).

@Url: The "Url" attribute of the Action. The value MUST be a URL (section 2.2.1.18).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;100&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>
| -6    | The Action with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current version of the Action. For example, this error can be triggered when a thread reads the given Action, after which another thread updates the same
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>An Action, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An Action with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The Entity that contains this Action already contains another Action with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

A positive integer: A T-SQL error code.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

### 3.2.5.123 proc_ar_UpdateActionParameterById

The `proc_ar_UpdateActionParameterById` stored procedure is called to change the attributes of the ActionParameter identified by the specified MetadataObjectId. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateActionParameterById (  
  @Id int  
  ,@IsCached bit  
  ,@PartitionId uniqueidentifier  
  ,@Version int OUTPUT  
  ,@Name nvarchar(50)  
  ,@Index tinyint  
  ,@ErrorCode int OUTPUT  
);
```

@Id: The MetadataObjectId of the ActionParameter that is to be updated. The value MUST be an Id (2.2.1.1).

@IsCached: A bit that specifies whether this ActionParameter is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition of the MetadataObject to update. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the ActionParameter. The protocol client MUST set the value to the object version of the ActionParameter at the time the ActionParameter was last read by the protocol client. The protocol server MUST increment the object version of the ActionParameter upon successful execution of this stored procedure. If the incremented object version of the ActionParameter is equal to 2,147,483,646, the protocol server MUST set the object version of the ActionParameter to 0. The protocol server MUST return the object version of the ActionParameter on output.
@Name: The name of the ActionParameter. The value MUST be an ActionParameterName (section 2.2.1.41).

@index: The Index attribute of the ActionParameter. The value MUST be an Index (section 2.2.1.19).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;102&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The ActionParameter with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the ActionParameter. For example, this error can be triggered when a thread reads the given ActionParameter, after which another thread updates the same ActionParameter, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An ActionParameter with the specified MetadataObjectId does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The Action that contains this ActionParameter already contains another ActionParameter with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled because of an implementation-specific integrity violation in the state of the data maintained by the protocol server. The protocol client MAY &lt;103&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.124 proc_ar_UpdateAssociationById

The proc_ar_UpdateAssociationById stored procedure is called to change the attributes of the Association identified by its given MetadataObjectId. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateAssociationById ( @Id int, @Name nvarchar(255), @IsCached bit, @MethodId int, @ReturnTypeDescriptorId int, @Type tinyint, @PartitionId uniqueidentifier, @Version int OUTPUT, @ErrorCode int OUTPUT );
```
@Id: The MetadataObjectId of the **Association** that is to be updated. The value MUST be an **Id** (2.2.1.1).

@Name: The name of the **Association**. The value MUST be a **Name** (section 2.2.1.2).

@IsCached: A bit that specifies if this **Association** is frequently used. The value MUST be an **IsCached** (section 2.2.1.5).

@MethodId: The **MethodId** of the **Association**. The value MUST be an **Id**.

@ReturnTypeDescriptorId: The MetadataObjectId of the **ReturnTypeDescriptor**. The value MUST be an **Id**. It MUST be equal to the **ReturnTypeDescriptor** specified when the **Association** was created.

@Type: The type of the **Association**. The value MUST be a **MethodInstanceType** (section 2.2.1.23). It MUST be equal to the MethodInstance type specified when the **Association** was created.

@PartitionId: The metadata partition of the **MetadataObject** to update. The value MUST be a **PartitionId** (section 2.2.1.4).

@Version: The object version of the **Association**. The protocol client MUST set the value to the object version of the **Association** at the time the **Association** was last read by the protocol client. The protocol server MUST increment the object version of the **Association** upon successful execution of this stored procedure. If the incremented object version of the **Association** is equal to 2,147,483,646, the protocol server MUST set the object version of the **Association** to 0. The protocol server MUST return the object version of the **Association** on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-500</td>
<td>This happens when the specified <strong>ReturnTypeDescriptorId</strong> does not match the <strong>MetadataObjectId</strong> of the <strong>ReturnTypeDescriptor</strong> of the <strong>Association</strong> or if the value of <strong>@Type</strong> does not match the <strong>MethodInstance</strong> type for the <strong>Association</strong>.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;104&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td><strong>Association</strong> could not be changed on an active <strong>Entity</strong>.</td>
</tr>
<tr>
<td>-6</td>
<td>The <strong>Association</strong> with the specified <strong>MetadataObjectId</strong> has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>Association</strong>. For example, this error can be triggered when a thread reads the given <strong>Association</strong>, after which another thread updates the same <strong>Association</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An <strong>Association</strong> with specified <strong>MetadataObjectId</strong> does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>An <strong>Association</strong> with the specified name already exists within the <strong>Entity</strong> that contains the specified <strong>Association</strong> being updated.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client...</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAY&lt;105&gt;</td>
<td>retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.125 proc_ar_UpdateAssociationGroupById

The proc_ar_UpdateAssociationGroupById stored procedure is called to change the attributes of the AssociationGroup identified by its given MetadataObjectId. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateAssociationGroupById (  
    @Id int,  
    @Name nvarchar(255),  
    @IsCached bit,  
    @EntityId int,  
    @PartitionId uniqueidentifier,  
    @Version int OUTPUT,  
    @ErrorCode int OUTPUT
);
```

@Id: The MetadataObjectId of the AssociationGroup to be updated. The value MUST be an Id (2.2.1.1).

@Name: The name of the AssociationGroup. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies if the AssociationGroup is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@EntityId: The MetadataObjectId of the Entity which contains this AssociationGroup. The value MUST be an Id. The specified Entity SHOULD<106> be in the same Partition as the AssociationGroup to be updated.

@PartitionId: The metadata partition of the MetadataObject to update. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the AssociationGroup. The protocol client MUST set the value to the object version of the AssociationGroup at the time the AssociationGroup was last read by the protocol client. The protocol server MUST increment the object version of the AssociationGroup upon successful execution of this stored procedure. If the incremented object version of the AssociationGroup is equal to 2,147,483,646, the protocol server MUST set the object version of the AssociationGroup to 0. The protocol server MUST return the object version of the AssociationGroup on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY[^107] retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>Either the Entity containing the AssociationGroup before update was an active Entity or the specified Entity is an active Entity.</td>
</tr>
<tr>
<td>-6</td>
<td>The AssociationGroup with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the AssociationGroup. For example, this error can be triggered when a thread reads the given AssociationGroup, after which another thread updates the same AssociationGroup, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>An AssociationGroup with the specified MetadataObjectId does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The Entity that contains this AssociationGroup already contains another AssociationGroup with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY[^108] retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.126 proc_ar_UpdateEntityById

The **proc_ar_UpdateEntityById** stored procedure is called to change the attributes of the Entity identified by the specified MetadataObjectId. If the specified name and the namespace is different from the current name and namespace of the Entity, the names and namespaces of all versions of the Entity MUST be updated. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateEntityById (  
@Id int, 
@Name nvarchar(255), 
@Namespace nvarchar(255), 
@IsCached bit, 
@PartitionId uniqueidentifier, 
@MajorVersion int, 
@MinorVersion int, 
@BuildVersion int, 
@RevisionVersion int, 
@Version int OUTPUT, 
@SystemId int, 
@EstimatedInstanceCount int, 
@CacheUsage int, 
@ErrorCode int OUTPUT
)```

[^107]: Reference to a specific section or page number.
[^108]: Reference to a specific section or page number.
@Id: The MetadataObjectId of the Entity to be updated. The value MUST be an Id (2.2.1.1).

@Name: The name of the Entity. The value MUST be a Name (section 2.2.1.2).

@Namespace: Namespace of the Entity to be updated. The value MUST be a Namespace (section 2.2.1.3).

@IsCached: A bit that specifies whether this Entity is frequently used. The value must be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition of the MetadataObject to update. The value MUST be an PartitionId (section 2.2.1.4).

@MajorVersion: Major version of the Entity to update. The value MUST be a MajorVersion (section 2.2.1.7).

@MinorVersion: Minor Version of the Entity to update. The value MUST be a MinorVersion (section 2.2.1.8).

@BuildVersion: Build Version of the Entity to update. The value MUST be a BuildVersion (section 2.2.1.9).

@RevisionVersion: Revision Version of the Entity to update. The value MUST be a RevisionVersion (section 2.2.1.10).

@Version: The object version of the Entity. The protocol client MUST set the value to the object version of the Entity at the time the Entity was last read by the protocol client. The protocol server MUST increment the object version of the Entity upon successful execution of this stored procedure. If the incremented object version of the Entity is equal to 2,147,483,646, the protocol server MUST set the object version of the Entity to 0. The protocol server MUST return the object version of the Entity on output.

@SystemId: The MetadataObjectId of the LobSystem that contains this Entity. The value MUST be an Id.

@EstimatedInstanceCount: Represents the estimated maximum number of EntityInstances for the Entity to be updated, returned from the LobSystemInstance. The value must be an EstimatedInstanceCount (section 2.2.1.11).

@CacheUsage: The Cache usage mode to be used in the Entity. The value must be a CacheUsage (section 2.2.1.13).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1007</td>
<td>The specified name or namespace is currently being referenced from other MetadataObjects.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Entity with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-4</td>
<td>The specified CacheUsage, MajorVersion, MinorVersion, BuildVersion, or RevisionVersion are not valid.</td>
</tr>
<tr>
<td>-3</td>
<td>The LobSystem already contains the implementation-specific maximum allowed number of Entities.</td>
</tr>
<tr>
<td>-2</td>
<td>An Entity with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
</tbody>
</table>
| -1    | Any of the following conditions are true:  
- The LobSystem that contains this Entity already contains another Entity with the specified name and namespace when either the specified name or the specified namespace is different from the existing name or namespace, respectively.  
- The LobSystem that contains this Entity already contains another Entity with the specified name, namespace, major version, minor version, build version and revision version. |
| 0     | No errors encountered. |
| -1100 | The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY<110> retry the operation by calling this stored procedure again. |
| A positive integer | A T-SQL error code. |

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.127 proc_ar_UpdateFilterDescriptorByld

The proc_ar_UpdateFilterDescriptorByld stored procedure is called to change the attributes of the FilterDescriptor identified by the specified MetadataObjectId. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateFilterDescriptorByld (  
@Id int,  
@Name nvarchar(255),  
@IsCached bit,  
@PartitionId uniqueidentifier,  
@Version int OUTPUT,  
@FilterType tinyint,  
@FilterField nvarchar(255),  
@ErrorCode int OUTPUT );
```

@Id: The MetadataObjectId of the FilterDescriptor that is to be updated. The value MUST be an Id (2.2.1.1).
@Name: The name of the FilterDescriptor. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this FilterDescriptor is frequently used. The value MUST be an IsCached (section 2.2.1.9).

@PartitionId: The metadata partition of the MetadataObject to update. The value MUST be a PartitionId.

@Version: The object version of the FilterDescriptor. The protocol client MUST set the value to the object version of the FilterDescriptor at the time the FilterDescriptor was last read by the protocol client. The protocol server MUST increment the object version of the FilterDescriptor upon successful execution of this stored procedure. If the incremented object version of the FilterDescriptor is equal to 2,147,483,646, the protocol server MUST set the object version of the FilterDescriptor to 0. The protocol server MUST return the object version of the FilterDescriptor on output.

@FilterType: The type of the FilterDescriptor. The value MUST be a FilterType (section 2.2.1.20).

@FilterField: The field (4) affected by the FilterDescriptor. The value MUST be a FilterField (section 2.2.1.21).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>The error is thrown in the following cases:</td>
</tr>
<tr>
<td></td>
<td>• The Method associated with this FilterDescriptor already contains another FilterDescriptor of type TimeStampFilter and a new FilterDescriptor of type TimeStampFilter is added.</td>
</tr>
<tr>
<td></td>
<td>• The Method that contains this FilterDescriptor also contains a ChangedIdEnumerator or a DeletedIdEnumerator, and the type of the FilterDescriptor is changed from TimeStampFilter to another type.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The FilterDescriptor with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the FilterDescriptor. For example, this error can be triggered when a thread reads the given FilterDescriptor, after which another thread updates the same FilterDescriptor, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>A FilterDescriptor with specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The Method that contains this FilterDescriptor already contains another FilterDescriptor with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled because of an implementation-specific integrity violation in the state of the data stored by the protocol server. The protocol client MAY retry the</td>
</tr>
</tbody>
</table>
3.2.5.128 proc_ar_UpdateIdentifierById

The proc_ar_UpdateIdentifierById stored procedure is called to change the attributes of the Identifier identified by the specified MetadataObjectId. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_UpdateIdentifierById (
    @Id int,
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @Version int OUTPUT,
    @TypeName nvarchar(255),
    @ErrorCode int OUTPUT);
```

@Id: The MetadataObjectId of the Identifier to be updated. The value MUST be an Id (2.2.1.1).

@Name: The new name to be set for the Identifier. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this Identifier is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition of the Identifier to update. The value MUST be an PartitionId (section 2.2.1.4).

@Version: The object version of the Identifier. The protocol client MUST set the value to the object version of the Identifier at the time the Identifier was last read by the protocol client. The protocol server MUST increment the object version of the Identifier upon successful execution of this stored procedure. If the incremented object version of the Identifier is equal to 2,147,483,646, the protocol server MUST set the object version of the Identifier to 0. The protocol server MUST return the object version of the Identifier on output.

@TypeName: The type name of the Identifier. The value MUST be an IdentifierTypeName (section 2.2.1.22).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7</td>
<td>The Entity with the specified \texttt{MetadataObjectId} was an active \texttt{Entity}.</td>
</tr>
<tr>
<td>-6</td>
<td>An \texttt{Entity} with the specified \texttt{MetadataObjectId} has been updated by a context other than the one that it has been currently read by. This happens when the version specified does not match with the current version of the \texttt{Entity}.</td>
</tr>
<tr>
<td>-2</td>
<td>An \texttt{Identifier} with the specified \texttt{MetadataObjectId} does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The \texttt{Entity} with the specified \texttt{MetadataObjectId} already contains another \texttt{Identifier} with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client \texttt{MAY\textless 114&gt;} retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that \texttt{MUST} be 0.

**Result Sets:** \texttt{MUST NOT} return any result sets.

### 3.2.5.129 proc\_ar\_UpdateMethodById

The \texttt{proc\_ar\_UpdateMethodById} stored procedure is called to change the attributes of the Method identified by the specified \texttt{MetadataObjectId}. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateMethodById (
    @Id int,
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @Version int OUTPUT,
    @IsStatic bit,
    @LobName nvarchar(255),
    @ErrorCode int OUTPUT
);
```

@\texttt{Id}: The \texttt{MetadataObjectId} of the Method to be updated. The value \texttt{MUST} be an \texttt{Id} (2.2.1.1).

@\texttt{Name}: The name of the Method. The value \texttt{MUST} be a \texttt{Name} (section 2.2.1.2).

@\texttt{IsCached}: A bit that specifies whether the Method is frequently used. The value \texttt{MUST} be an \texttt{IsCached} (section 2.2.1.5).

@\texttt{PartitionId}: The metadata partition of the Method to update. Value \texttt{MUST} be a \texttt{PartitionId} (section 2.2.1.4).

@\texttt{Version}: The object version of the Method. The protocol client \texttt{MUST} set the value to the object version of the Method at the time the Method was last read by the protocol client. The protocol server \texttt{MUST} increment the object version of the Method upon successful execution of this stored procedure. If the incremented object version of the Method is equal to 2,147,483,646, the protocol server returns a value of -1100.

[MS-BDCDPS2] — v20120630

\textit{Business Data Connectivity Database Version 2 Protocol Specification}

\textit{Copyright © 2012 Microsoft Corporation.}

\textit{Release: July 16, 2012}
server MUST set the object version of the Method to 0. The protocol server MUST return the object version of the Method on output.

@IsStatic: A bit specifying whether the Method is associated with an EntityInstance. The value MUST be an IsStatic (section 2.2.1.33).

@LobName: The name of the corresponding method on the line-of-business (LOB) system. The value MUST be a MethodLobName (section 2.2.1.34).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Method with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Method. For example, this error can be triggered when a thread reads the given Method, after which another thread updates the same Method, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>A Method with specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The Entity that contains the Method with the specified MetadataObjectId already contains another Method with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.130 proc_ar_UpdateMethodInstanceById

The proc_ar_UpdateMethodInstanceById is called to update the attributes of the MethodInstance with the specified MetadataObjectId. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateMethodInstanceById ( 
    @Id int,
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @Version int OUTPUT,
    @ReturnTypeDescriptorId int,
    @IsDefault bit,
    @Type tinyint
) PREPARE

RETURN
```
@Id: The MetadataObjectId of the MethodInstance to update. The value MUST be an Id (2.2.1.1).

@Name: The name of the MethodInstance. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this MethodInstance is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition of the MethodInstance to update. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the MethodInstance. The protocol client MUST set the value to the object version of the MethodInstance at the time the MethodInstance was last read by the protocol client. The protocol server MUST increment the object version of the MethodInstance upon successful execution of this stored procedure. If the incremented object version of the MethodInstance is equal to 2,147,483,646, the protocol server MUST set the object version of the MethodInstance to 0. The protocol server MUST return the object version of the MethodInstance on output.

@ReturnTypeDescriptorId: The MetadataObjectId of the ReturnTypeDescriptor. If the MethodInstance does not have a return value, the value MUST be NULL. Otherwise, the value MUST be an Id, and the referenced TypeDescriptor MUST exist in the metadata store.

@IsDefault: A bit that specifies if this MethodInstance is default among MethodInstances that has the same value for MethodInstanceType (section 2.2.1.23) attribute within the ancestor DataClass. The value MUST be an IsDefault (section 2.2.1.35). When this value is set to 1, this stored procedure MUST set IsDefault attribute of all other MethodInstances that has the same value for MethodInstanceType attribute (section 2.2.1.23) within the ancestor DataClass to 0. When this value is set to 0, the protocol server MUST set the IsDefault attribute of any MethodInstance with the same value for MethodInstanceType within the ancestor DataClass to 1, determined with an implementation-specific algorithm.

@Type: The type of the MethodInstance. The value MUST be a MethodInstanceType. If the specified type is different from the current type, and if this MethodInstance was a default, this stored procedure MUST set IsDefault attribute of any of the MethodInstance with the MethodInstanceType attribute equal to the previous type within the ancestor DataClass to 1, determined with an implementation-specific algorithm.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-217</td>
<td>The specified type for the MethodInstance requires a Parameter with Direction (section 2.2.1.24) set to &quot;In&quot; or &quot;InOut&quot; to be present on the Method of this MethodInstance.</td>
</tr>
<tr>
<td>-214</td>
<td>The ReturnTypeDescriptor is required not to contain any child TypeDescriptors for the specified type for the MethodInstance, however the specified ReturnTypeDescriptor has child TypeDescriptors.</td>
</tr>
<tr>
<td>-211</td>
<td>The DataClass that contains this MethodInstance already contains another MethodInstance which has the MethodInstanceType attribute set to DeletedIdEnumerator.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>210</td>
<td>The DataClass that contains this MethodInstance already contains another MethodInstance that has the MethodInstanceType attribute set to ChangedIdEnumerator.</td>
</tr>
<tr>
<td>209</td>
<td>The DataClass that contains this MethodInstance already contains another MethodInstance which has the MethodInstanceType attribute set to Deleter.</td>
</tr>
<tr>
<td>208</td>
<td>The ReturnTypeDescriptor is required to have &quot;IsCollection&quot; flag not set for the specified type for the MethodInstance, however the specified ReturnTypeDescriptor has this flag set.</td>
</tr>
<tr>
<td>207</td>
<td>The ReturnTypeDescriptor is required to have &quot;IsCollection&quot; flag set for the specified type for the MethodInstance, however the specified ReturnTypeDescriptor does not have this flag set.</td>
</tr>
<tr>
<td>206</td>
<td>The ReturnTypeDescriptor is required for the specified type for the MethodInstance, however it is passed in as NULL or 0.</td>
</tr>
<tr>
<td>205</td>
<td>The DataClass that contains this MethodInstance already contains another MethodInstance which has the MethodInstanceType attribute set to AccessChecker.</td>
</tr>
<tr>
<td>204</td>
<td>The Parameter of the specified ReturnTypeDescriptor has the Direction attribute set to &quot;In&quot;.</td>
</tr>
<tr>
<td>203</td>
<td>The Parameter of the specified ReturnTypeDescriptor is not in the same Method as this MethodInstance.</td>
</tr>
<tr>
<td>202</td>
<td>The DataClass that contains this MethodInstance already contains another MethodInstance which has the MethodInstanceType attribute set to IdEnumerator.</td>
</tr>
<tr>
<td>8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>6</td>
<td>The MethodInstance with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the MethodInstance. For example, this error can be triggered when a thread reads the given MethodInstance, after which another thread updates the same MethodInstance, and then the original thread tries to update.</td>
</tr>
<tr>
<td>4</td>
<td>The value of @Type parameter is not a valid MethodInstanceType.</td>
</tr>
<tr>
<td>2</td>
<td>A MethodInstance with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>1</td>
<td>The DataClass that contains this MethodInstance already contains another MethodInstance with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>
Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.131 proc_ar_UpdateModelById

The proc_ar_UpdateModelById stored procedure is called to change the attributes of the Model with the specified MetadataObjectId. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_UpdateModelById (
  @Id int,
  @Name nvarchar(255),
  @IsCached bit,
  @PartitionId uniqueidentifier,
  @Version int OUTPUT,
  @ErrorCode int OUTPUT
);
```

@Id: The MetadataObjectId of the Model that needs to be updated. The value MUST be an Id (2.2.1.1)

@Name: The new name of the Model. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit value that specifies whether the Model is frequently used. This value MUST be IsCached (section 2.2.1.5).

@PartitionId: The Metadata partition of the Model to update. Value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the Model. The protocol client MUST set the value to the object version of the Model at the time the Model was last read by the protocol client. The protocol server MUST increment the object version of the Model upon successful execution of this stored procedure. If the incremented object version of the Model is equal to 2,147,483,646, the protocol server MUST set the object version of the Model to 0. The protocol server MUST return the object version of the Model on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>A Model with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version does not match the current object version of the Model. For example, this error can be triggered when a thread reads the given Model, after which another thread updates the same Model, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-2</td>
<td>A Model with the specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>Another Model with the specified name already exists in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integration violation detected in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

**A positive integer**

A T-SQL error code.

### Return Values:
An integer that MUST be 0.

### Result Sets:
MUST NOT return any result sets.

#### 3.2.5.132  proc_ar_UpdateParameterById

The `proc_ar_UpdateParameter` stored procedure is called to update the attributes of the Parameter specified by the given `MetadataObjectId`. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateParameterById (
  @Id int
  ,@Name nvarchar(255)
  ,@IsCached bit
  ,@PartitionId uniqueidentifier
  ,@Version int OUTPUT
  ,@OrdinalNumber tinyint OUTPUT
  ,@Direction tinyint
  ,@ErrorCode int OUTPUT
);
```

**@Id:** The `MetadataObjectId` of the Parameter to update. The value MUST be an Id (2.2.1.1).

**@Name:** The name of the Parameter. The value MUST be a Name (section 2.2.1.2).

**@IsCached:** A bit that specifies whether this Parameter is frequently used. The value MUST be an IsCached (section 2.2.1.5).

**@PartitionId:** The metadata partition of the Parameter to update. The value MUST be a PartitionId (section 2.2.1.4).

**@Version:** The object version of the Parameter. The protocol client MUST set the value to the object version of the Parameter at the time the Parameter was last read by the protocol client. The protocol server MUST increment the object version of the Parameter upon successful execution of this stored procedure. If the incremented object version of the Parameter is equal to 2,147,483,646, the protocol server MUST set the object version of the Parameter to 0. The protocol server MUST return the object version of the Parameter on output.

**@OrdinalNumber:** The position of the Parameter in the signature of the Method containing this Parameter. If the position is the same as another Parameter's position for the same parent Method, the other Parameter's position, along with all Parameters positioned subsequently, are incremented. When the stored procedure returns, all Parameters of the Method containing this Parameter MUST have positions in the range 0 to X, where X plus 1 is the number of Parameters in the Method. Parameters in the Method other than this Parameter MUST NOT have their relative positioning altered.

**@Direction:** The direction of the Parameter. The value MUST be a Direction (section 2.2.1.24).
**@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-103</td>
<td>This Parameter is not allowed to have value &quot;In&quot; for the Direction attribute because one of the TypeDescriptors in this parameter has &quot;Read-Only&quot; flag set for its TypeDescriptorFlags (section 2.2.1.28) attribute.</td>
</tr>
<tr>
<td>-102</td>
<td>This Parameter is not cannot be set to &quot;In&quot; for Direction because this Parameter contains the ReturnTypeDescriptor of a MethodInstance.</td>
</tr>
<tr>
<td>-100</td>
<td>The Method that contains this Parameter already contains another Parameter with Direction set to &quot;Return&quot;.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY&lt;121&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The Parameter with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the Parameter. For example, this error can be triggered when a thread reads the given Parameter, after which another thread updates the same Parameter, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-4</td>
<td>The value of the @Direction parameter is not a valid Direction.</td>
</tr>
<tr>
<td>-2</td>
<td>A Parameter with specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The Method that contains this Parameter already contains another Parameter with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client MAY&lt;122&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.133 proc_ar_UpdateProgress

The proc_ar_UpdateProgress stored procedure is called to update the progress of an application specific operation. The progress can be retrieved by the proc_ar_RetrieveProgress (section 3.2.5.116) stored procedure. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateProgress ( 
@PartitionId uniqueidentifier 
,@JobKey uniqueidentifier 
,@Progress real 
);
```
@PartitionId: The metadata partition associated with the operation. The value MUST be a PartitionId (section 2.2.1.4).

@JobKey: The identifier of the operation. The value MUST be a GUID.

@Progress: The fraction of the operation that is complete. The value MUST be a between 0 and 1.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.134  proc_ar_UpdateSystemById

The proc_ar_UpdateSystemById stored procedure is called to change the attributes of the LobSystem identified by the specified MetadataObjectId. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_UpdateSystemById (  
  @Id int,  
  @Name nvarchar(255),  
  @IsCached bit,  
  @PartitionId uniqueidentifier,  
  @Version int OUTPUT,  
  @SystemType tinyint,  
  @ErrorCode int OUTPUT );
```

@Id: The MetadataObjectId of the LobSystem to be updated. The value MUST be an Id (2.2.1.1).

@Name: The name of the LobSystem. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this LobSystem is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The partition of the MetadataObject to update. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the LobSystem. The protocol client MUST set the value to the object version of the LobSystem at the time the LobSystem was last read by the protocol client. The protocol server MUST increment the object version of the LobSystem upon successful execution of this stored procedure. If the incremented object version of the LobSystem is equal to 2,147,483,646, the protocol server MUST set the object version of the LobSystem to 0. The protocol server MUST return the object version of the LobSystem on output.

@SystemType: Type of the LobSystem. The value MUST be a SystemType (section 2.2.1.30).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The LobSystem with the specified MetadataObjectId has been updated by a context other than the one that it has been currently read by. This happens when the specified object</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-2</td>
<td>A LobSystem with the specified MetadataObjectId does not exist in the given Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The metadata store contains another LobSystem with the specified @Name in the given Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled because of an implementation-specific integrity violation in the state of the data stored by the protocol server. The protocol client MAY retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

A positive integer  
A T-SQL error code.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.135 proc_ar_UpdateSystemInstanceById

The proc_ar_UpdateSystemInstanceById stored procedure is called to change the attributes of LobSystemInstance identified by the specified MetadataObjectId. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateSystemInstanceById (  
    @Id int,
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @Version int OUTPUT,  
    @SystemId int,
    @ErrorCode int OUTPUT);
```

@Id: The MetadataObjectId of the LobSystemInstance to be updated. The value MUST be an Id (2.2.1.1).

@Name: The name of the LobSystemInstance. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this LobSystemInstance is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The partition of the MetadataObject to update. The value MUST be a PartitionId (section 2.2.1.4).

@Version: The object version of the LobSystemInstance. The protocol client MUST set the value to the object version of the LobSystemInstance at the time the LobSystemInstance was last read by the protocol client. The protocol server MUST increment the object version of the LobSystemInstance upon successful execution of this stored procedure. If the incremented object...
version of the **LobSystemInstance** is equal to 2,147,483,646, the protocol server MUST set the object version of the **LobSystemInstance** to 0. The protocol server MUST return the object version of the **LobSystemInstance** on output.

**@SystemId:** The **MetadataObjectId** of the LobSystem that contains this **LobSystemInstance**. The value MUST be an **Id**.

**@ErrorCode:** The error code. Upon return from this stored procedure, this parameter MUST be set to an integer in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;125&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-6</td>
<td>The <strong>LobSystemInstance</strong> with the specified <strong>MetadataObjectId</strong> has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>LobSystemInstance</strong>. For example, this error can be triggered when a thread reads the given <strong>LobSystemInstance</strong>, after which another thread updates the same <strong>LobSystemInstance</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-3</td>
<td>The <strong>LobSystem</strong> with <strong>@SystemId</strong> already contains implementation-specific maximum number of <strong>LobSystemInstances</strong>.</td>
</tr>
<tr>
<td>-2</td>
<td>A <strong>LobSystemInstance</strong> with the specified <strong>MetadataObjectId</strong> does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The specified <strong>LobSystem</strong> contains another <strong>LobSystemInstance</strong> with the specified name in the given Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled because of an implementation-specific integrity violation in the state of the data stored by the protocol server. The protocol client MAY &lt;126&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>A positive integer</td>
<td>A T-SQL error code.</td>
</tr>
</tbody>
</table>

**Return Values:** An integer that MUST be 0.

**Result Sets:** MUST NOT return any result sets.

### 3.2.5.136 proc_ar_UpdateTypeDescriptorByld

The **proc_ar_UpdateTypeDescriptorByld** stored procedure is called to update the attributes of the TypeDescriptor identified by the given **MetadataObjectId**. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_UpdateTypeDescriptorByld (
    @Id int,
    @Name nvarchar(255),
    @IsCached bit,
    @PartitionId uniqueidentifier,
    @ParentTypeDescriptorId int
)```

[MS-BDCDPS2] — v20120630
*Business Data Connectivity Database Version 2 Protocol Specification*

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
@Id: The MetadataObjectId of the TypeDescriptor to update. The value MUST be an Id (2.2.1.1).

@Name: The name of the TypeDescriptor. The value MUST be a Name (section 2.2.1.2).

@IsCached: A bit that specifies whether this TypeDescriptor is frequently used. The value MUST be an IsCached (section 2.2.1.5).

@PartitionId: The metadata partition of the TypeDescriptor to update. The value MUST be a PartitionId (section 2.2.1.4).

@ParentTypeDescriptorId: The MetadataObjectId of the TypeDescriptor which is the parent of the TypeDescriptor that is being updated. If the TypeDescriptor is a root TypeDescriptor, the value MUST be NULL. Otherwise, the value MUST be an Id.

@TypeName: The identifier of the data type that is represented by this TypeDescriptor. The value MUST be a TypeDescriptorTypeName (section 2.2.1.25).

@IdentifierId: The MetadataObjectId of the Identifier referenced by this TypeDescriptor. If this TypeDescriptor references an Identifier of an active Entity, the value MUST be an Id. Otherwise, the value MUST be NULL or 0.

@FilterDescriptorId: The MetadataObjectId of the FilterDescriptor associated with this TypeDescriptor. If a FilterDescriptor is associated with this TypeDescriptor, the value MUST be an Id. Otherwise, the value MUST be NULL.

@LobName: The name of the data structure that is represented by this TypeDescriptor. The value MUST be a TypeDescriptorLobName (section 2.2.1.26).

@Rules: The rules for this TypeDescriptor. The value MUST be a TypeDescriptorInterpretation (section 2.2.1.27).

@Flags: The flags for this TypeDescriptor. The value MUST be a TypeDescriptorFlags (section 2.2.1.28).
@AssociationId: The MetadataObjectId of the Association referenced by this TypeDescriptor. If this TypeDescriptor references an Association defined on an active DataClass, the value MUST be an Id. Otherwise, the value MUST be NULL or 0.

@_IdentifierName: The name of the Identifier referenced by this TypeDescriptor. If this TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Name (section 2.2.1.2). Otherwise, the value MUST be NULL.

@_IdentifierEntityName: The name of the Entity that contains the Identifier referenced by this TypeDescriptor. If this TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Name. Otherwise, the value MUST be NULL.

@_IdentifierEntityNamespace: The namespace of the Entity that contains the Identifier referenced by this TypeDescriptor. If this TypeDescriptor references an Identifier of an Entity that is not active, the value MUST be a Namespace (section 2.2.1.3). Otherwise, the value MUST be NULL.

@_AssociationName: The name of the Association referenced by this TypeDescriptor. If this TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Name. Otherwise, the value MUST be NULL.

@_AssociationEntityName: The name of the Entity that contains the Association referenced by this TypeDescriptor. If this TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Name. Otherwise, the value MUST be NULL.

@_AssociationEntityNamespace: The namespace of the Entity that contains the Association referenced by this TypeDescriptor. If this TypeDescriptor references an Association of an Entity that is not active, the value MUST be a Namespace. Otherwise, the value MUST be NULL.

@Version: The object version of the TypeDescriptor. The protocol client MUST set the value to the object version of the TypeDescriptor at the time the TypeDescriptor was last read by the protocol client. The protocol server MUST increment the object version of the TypeDescriptor upon successful execution of this stored procedure. If the incremented object version of the TypeDescriptor is equal to 2,147,483,646, the protocol server MUST set the object version of the TypeDescriptor to 0. The protocol server MUST return the object version of the TypeDescriptor on output.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-309</td>
<td>The &quot;ReadOnly&quot; flag cannot be set as the Parameter if this TypeDescriptor has Direction (section 2.2.1.24) set to &quot;In&quot;.</td>
</tr>
<tr>
<td>-308</td>
<td>The DataClass of the referenced Association, specified by the MetadataObjectId of the Association is not active.</td>
</tr>
<tr>
<td>-307</td>
<td>The Entity of the referenced Identifier, specified by MetadataObjectId of the Identifier is not active.</td>
</tr>
<tr>
<td>-306</td>
<td>A TypeDescriptor with &quot;IsCollection&quot; flag set can only have one child TypeDescriptor.</td>
</tr>
<tr>
<td>-305</td>
<td>The &quot;IsCollection&quot; flag cannot be set on a TypeDescriptor if its parent TypeDescriptor also has &quot;IsCollection&quot; flag set.</td>
</tr>
<tr>
<td>-304</td>
<td>Parameter of the specified parent TypeDescriptor is different from the Parameter of this TypeDescriptor.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-303</td>
<td>The filter associated with this <strong>TypeDescriptor</strong> is not defined on the <strong>Method</strong> which contains the <strong>Parameter</strong> of this <strong>TypeDescriptor</strong>.</td>
</tr>
<tr>
<td>-302</td>
<td>The specified <strong>Parameter</strong> already has a root <strong>TypeDescriptor</strong>.</td>
</tr>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client <strong>MAY</strong> &lt;127&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-7</td>
<td>The <strong>DataClass</strong> that is the ancestor of this <strong>TypeDescriptor</strong> is active.</td>
</tr>
<tr>
<td>-6</td>
<td>The <strong>Parameter</strong> with the specified <strong>MetadataObjectId</strong> has been updated by a context other than the one that it has been currently read by. This happens when the specified object version is not equal to the current object version of the <strong>Parameter</strong>. For example, this error can be triggered when a thread reads the given <strong>Parameter</strong>, after which another thread updates the same <strong>Parameter</strong>, and then the original thread tries to update.</td>
</tr>
<tr>
<td>-4</td>
<td>The flags set for this <strong>TypeDescriptor</strong> are not valid.</td>
</tr>
<tr>
<td>-3</td>
<td>At least one of the following is true: This <strong>TypeDescriptor</strong> is not a root <strong>TypeDescriptor</strong> and the specified parent <strong>TypeDescriptor</strong> already has the implementation-specific maximum number of child <strong>TypeDescriptors</strong>. A <strong>FilterDescriptor</strong> is associated to this <strong>TypeDescriptor</strong> and the <strong>FilterDescriptor</strong> already has the implementation-specific maximum number of associated <strong>TypeDescriptors</strong>.</td>
</tr>
<tr>
<td>-2</td>
<td>A <strong>TypeDescriptor</strong> with specified <strong>MetadataObjectId</strong> does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>-1</td>
<td>The <strong>TypeDescriptor</strong> with <strong>MetadataObjectId</strong> equal to @<strong>parentTypeDescriptor</strong> that contains this <strong>Parameter</strong> already contains another <strong>Parameter</strong> with the specified name.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
<tr>
<td>-1100</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific integrity violation detected in the state of the data stored by the protocol server. The protocol client <strong>MAY</strong> &lt;128&gt; retry the operation by calling this stored procedure again.</td>
</tr>
</tbody>
</table>

A positive integer A T-SQL error code.

-300 Parameter of this **TypeDescriptor** has a **TypeDescriptor** hierarchy deeper than implementation-specific maximum allowed depth.

@ContainsIdentifier: The stored procedure MUST set this value to 1 if this **TypeDescriptor**, or any of its descendants reference an **Identifier**. Otherwise, this stored procedure MUST set this value to 0.

@ContainsFilterDescriptor: This stored procedure MUST set this value to 1 if this **TypeDescriptor**, or any of its descendants have an associated **FilterDescriptor**. Otherwise, stored procedure MUST set this value to 0.

@ContainsReadOnly: The stored procedure MUST set this value to 1 if this **TypeDescriptor**, or any of its descendants have "ReadOnly" flag set. Otherwise, stored procedure MUST set this value to 0.
@ChildrenContainRules: This stored procedure MUST set this value to 1 if any descendant of this TypeDescriptor have TypeDescriptorInterpretation attribute (section 2.2.1.27) value as not NULL. Otherwise, this stored procedure MUST set this value to 0.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.5.137 proc_ar_GetTypeById

The proc_ar_GetTypeById stored procedure is called to retrieve the type of the specified MetadataObject. This stored procedure is defined as follows.

PROCEDURE proc_ar_GetTypeById (  
    @MetadataObjectId int  
);  

@MetadataObjectId: The MetadataObjectId of the MetadataObject. The value MUST be an Id (2.2.1.1)

Return Values: An integer that MUST be in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The specified MetadataObject does not exist.</td>
</tr>
<tr>
<td>1</td>
<td>The specified MetadataObject is an Action.</td>
</tr>
<tr>
<td>2</td>
<td>The specified MetadataObject is an ActionParameter.</td>
</tr>
<tr>
<td>3</td>
<td>The specified MetadataObject is a MetadataCatalog.</td>
</tr>
<tr>
<td>5</td>
<td>The specified MetadataObject is an AssociationGroup.</td>
</tr>
<tr>
<td>8</td>
<td>The specified MetadataObject is a DataClass or an Entity.</td>
</tr>
<tr>
<td>10</td>
<td>The specified MetadataObject is a FilterDescriptor.</td>
</tr>
<tr>
<td>11</td>
<td>The specified MetadataObject is an Identifier.</td>
</tr>
<tr>
<td>12</td>
<td>The specified MetadataObject is a Method.</td>
</tr>
<tr>
<td>13</td>
<td>The specified MetadataObject is a MethodInstance or an Association.</td>
</tr>
<tr>
<td>14</td>
<td>The specified MetadataObject is a Model.</td>
</tr>
<tr>
<td>15</td>
<td>The specified MetadataObject is a Parameter.</td>
</tr>
<tr>
<td>16</td>
<td>The specified MetadataObject is a LobSystem.</td>
</tr>
<tr>
<td>17</td>
<td>The specified MetadataObject is a LobSystemInstance.</td>
</tr>
<tr>
<td>18</td>
<td>The specified MetadataObject is a TypeDescriptor.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.
3.2.5.138 proc_ar_GetTypeDescriptorForDottedPath

The proc_ar_GetTypeDescriptorForDottedPath stored procedure is called to retrieve a TypeDescriptor with a given path as specified in [MS-BDCMFFS] section 2.1.5.5 relative to the root TypeDescriptor of the specified Parameter if the specified MetadataObjectId belongs to a Parameter, or the specified TypeDescriptor if the specified MetadataObjectId belongs to a TypeDescriptor. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_GetTypeDescriptorForDottedPath (  
    @ParentTypeDescriptorOrParameterId int,  
    @DottedPath nvarchar(4000),  
    @PartitionId uniqueidentifier,  
    @ErrorCode int OUTPUT
);
```

@ParentTypeDescriptorOrParameterId: The MetadataObjectId of the TypeDescriptor or Parameter. The value MUST be an Id (2.2.1.1).

@DottedPath: The path to the TypeDescriptor to be retrieved from the root TypeDescriptor of the specified Parameter or specified TypeDescriptor. The value MUST be path as specified in [MS-BDCMFFS] section 2.1.5.5.

@PartitionId: The metadata partition of the TypeDescriptor or Parameter. The value MUST be a PartitionId (section 2.2.1.4).

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

Integers Less Than -100

The following is the ABNF for the error code structure. ABNF representation is specified in [RFC5234].

```plaintext
errorCode = %x2d errorPosition shortError
errorPosition = 1*DIGIT
shortError = 2*2DIGIT
```

(errorPosition) is an integer that MUST be set to the 1-based index of the character of the path where the error was encountered.

(shortError) is a two digit code that MUST be set to one of the following:

- **01**: The specified path conforms to [MS-BDCMFFS] section 2.1.5.5, but a Field token in the specified path refers to a TypeDescriptor that does not exist.
- **02, 03, 04, 05, or 07**: The specified path does not conform to [MS-BDCMFFS] section 2.1.5.5. <129>
- **08**: The specified path conforms to [MS-BDCMFFS] section 2.1.5.5, but an Indexer token that refers to a TypeDescriptor with the "IsCollection" flag not set.
- **09**: The specified path conforms to [MS-BDCMFFS] section 2.1.5.5, but contains a FieldAccess token that refers to a TypeDescriptor with the "IsCollection" flag.
Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a TypeDescriptor Result Set

3.2.5.139   proc_ar_CopyAccessControlEntriesForMetadataObjectIdAndSetting

The proc_ar_CopyAccessControlEntriesForMetadataObjectIdAndSetting stored procedure is called to copy ACEs of the specified source MetadataObject in the specified Setting to the same Setting on the specified destination MetadataObject in the same Metadata partition. If source MetadataObject and the destination MetadataObject are same, this stored procedure MUST make no changes. Otherwise, this stored procedure MUST first delete all ACEs in the specified Setting which are associated with the specified destination MetadataObject, before copying the ACEs. This stored procedure is defined as follows.

```
PROCEDURE proc_ar_CopyAccessControlEntriesForMetadataObjectIdAndSetting (  
    @SourceMetadataObjectId int  
  ,@DestinationMetadataObjectId int  
  ,@ErrorCode int OUTPUT    
  ,@PartitionId uniqueidentifier  
  ,@SettingId nvarchar(128)  
);  
```

@SourceMetadataObjectId: The MetadataObjectId of the source MetadataObject from which the ACEs will be copied from. The value MUST be an Id (2.2.1.1).

@DestinationMetadataObjectId: The MetadataObjectId of the destination MetadataObject with which ACEs will be copied to. The value MUST be an Id.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>The operation was cancelled by the protocol server because of an implementation-specific resource requirement that could not be fulfilled. The protocol client MAY &lt;130&gt; retry the operation by calling this stored procedure again.</td>
</tr>
<tr>
<td>-2</td>
<td>One or both of the specified MetadataObjects does not exist in the specified metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

@PartitionId: The metadata partition of the MetadataObjects. The value MUST be a PartitionId (section 2.2.1.4).

@SettingId: The Setting to read the ACEs from and write them to. Value MUST be a SettingId (section 2.2.1.6).

Return Values: An integer that MUST be 0.
Result Sets: MUST NOT return any result sets.

3.2.5.140  proc_ar_CheckPathInMethodInstances

The proc_ar_CheckPathInMethodInstances stored procedure is called to retrieve the MetadataObjectId of a MethodInstance in the specified DataClass that contains a specified TypeDescriptor. This stored procedure is defined as follows.

```sql
PROCEDURE proc_ar_CheckPathInMethodInstances (  
    @DottedPath nvarchar(4000)  
    ,@PartitionId uniqueidentifier  
    ,@ClassId int  
    ,@Type tinyint  
    ,@FoundMethodInstanceId int OUTPUT  
    ,@ErrorCode int OUTPUT  
);  
```

@DottedPath: The path to the TypeDescriptor from the TypeDescriptors contained by the ReturnTypeDescriptor of the MethodInstance. The value MUST be a path as specified in [MS-BDCMFFS] section 2.1.5.5.

@PartitionId: The metadata partition of the DataClass that contains the MethodInstance. The value MUST be a PartitionId (section 2.2.1.4).

@ClassId: The MetadataObjectId of DataClass that contains the MethodInstance. The value MUST be an Id (2.2.1.1).

@Type: The type of the MethodInstance to retrieve. The value MUST be a MethodInstanceType (section 2.2.1.23).

@FoundMethodInstanceId: The value MUST be the MetadataObjectId of any of the MethodInstances contained by the specified DataClass that contains a TypeDescriptor corresponding to the specified path. In this case the value MUST be an Id. If the specified DataClass contains more than one MethodInstance that contains a TypeDescriptor corresponding to the specified path, which MethodInstance is returned is determined in an implementation-specific manner. If the specified DataClass does not contain a MethodInstance that contains a TypeDescriptor corresponding to the specified path, the value MUST be 0.

@ErrorCode: The error code. Upon return from this stored procedure, this parameter MUST be set to an integer listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>A DataClass with specified MetadataObjectId does not exist in the specified Metadata partition.</td>
</tr>
<tr>
<td>0</td>
<td>No errors encountered.</td>
</tr>
</tbody>
</table>

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.6  Timer Events

None.
3.2.7 Other Local Events

None.

3.3 Client Details

The protocol client acts as a client when it calls the back-end database server requesting processing of stored procedures and optionally caching some of the data retrieved by the stored procedures.

3.3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The MetadataObjects stored in the metadata store can be maintained as object structures within the protocol client.

The protocol client sends messages to the protocol server to add, retrieve, change, and delete MetadataObjects stored in the protocol server.

3.3.1.1 MetadataObject Caching

The protocol client can cache the MetadataObjects and related structures obtained from the protocol server. Data within these structures may not be a complete representation of all data on the back-end database server, but can be populated as various requests to the back-end database server are fulfilled. Data may be cached at two levels independently:

- The MetadataObjects.
- The relationships between MetadataObjects.

Data maintained in the protocol client can be discarded after individual sequences of requests have finished as part of the cache invalidation mechanism. Cache invalidation can happen independently for objects and relationships. The protocol client MUST invalidate the cache when the cache version stamps obtained by proc_ar_GetCacheInvalidationCountersWithCount (section 3.2.5.67) are different from the corresponding cache invalidation stamps returned in a previous call to the proc_ar_GetCacheInvalidationCountersWithCount. This stored procedure call can be initiated with a timer to detect cache invalidations.

To trigger cache invalidation, the protocol client MUST call proc_ar_BumpCacheInvalidationCounters (section 3.2.5.6) with the type of the cache version stamp to increment.

Note that the cache can be implemented using a variety of techniques. An implementation is at liberty to implement such data in any way it pleases.

3.3.2 Timers

None.

3.3.3 Initialization

None.
3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The protocol client handles each stored procedure with the same basic processing method of calling the stored procedure and waiting for the result code and any result sets that will be returned.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.
4 Protocol Examples

This section provides specific example scenarios for operations on stored MetadataObjects. These examples describe in detail the process of communication between the protocol server and protocol client. In conjunction with the detailed client and server protocol specification in this document, this information is intended to provide a comprehensive view on how the protocol client operates with the protocol server when executing such an operation.

The examples in this section manipulate LobSystem and Entities. However, the principles illustrated apply equally to other MetadataObjects.

4.1 Create an LobSystem

This example illustrates how a user can create an LobSystem in the metadata store.

The following actions are carried out:

1. The user requests the protocol client to create an LobSystem with the name "ExampleCRM".
2. The protocol client calls the proc_ar_CreateSystem stored procedure using [MS-TDS]:

   DECLARE @return_value int,
   @ErrorCode int,
   @CreatedId int

   EXEC @return_value = proc_ar_CreateSystem
   @Name = N'ExampleCRM',
   @IsCached = 1,
   @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B',
   @SystemType = 1,
   @CreatedId = @CreatedId OUTPUT,
   @ErrorCode = @ErrorCode OUTPUT

3. The protocol server creates the LobSystem in the metadata store and it sets @ErrorCode to 0.
4. The protocol server returns a return code that the protocol client ignores.
5. The protocol client returns the @CreatedId and @ErrorCode values to the user.
6. The user inspects the @ErrorCode to see if the creation was successful.
7. The user saves the @CreatedId as the MetadataObjectId of the newly created Entity for subsequent use. Assume the value of @CreatedId is 33.

4.2 Set the Security Information of a MetadataObject

This example illustrates how a user can set security information of an LobSystem.

This example assumes that the preceding example has been successfully executed.

The following actions are carried out:

1. The user requests the protocol client to set ACEs on the LobSystem with the name "ExampleCRM" and SystemId 33.
2. The protocol client calls the `proc_ar_SetAccessControlEntryForMetadataObject` stored procedure using [MS-TDS]:

```sql
DECLARE @return_value int
EXEC @return_value = proc_ar_SetAccessControlEntryForMetadataObject
    @MetadataObjectId = 33,
    @IdentityName = N'Domain\User',
    @DisplayName = N'User',
    @RawSid = NULL,
    @Rights = '1',
    @SettingId = NULL
```

3. The protocol server returns a code that the protocol client ignores.

4.3 Read the Security Information of a MetadataObject

This example illustrates how a user can read the ACEs of an LobSystem.

This example assumes that the preceding examples have been successfully executed.

The following actions are carried out:

1. The user requests the protocol client to read ACEs for the `LobSystem` identified by MetadataObjectId 33.

2. The protocol client calls the `proc_ar_GetAccessControlEntriesForMetadataObject` stored procedure using [MS-TDS]:

```sql
DECLARE @return_value int,
    @ErrorCode int
EXEC @return_value = proc_ar_GetAccessControlEntriesForMetadataObject
    @MetadataObjectId = 33,
    @SettingId = NULL,
    @Fallback = 1,
    @ErrorCode = @ErrorCode OUTPUT
```

3. The protocol server checks whether a MetadataObject with MetadataObjectId 33 exists in the metadata store.

4. The protocol server retrieves the attributes of the ACE associated with the `LobSystem`.

5. The protocol server returns an Access Control Entry result set (section 2.2.5.28) with one row to the protocol client. The columns in the row and the values are as follows:

6. **MetadataObjectId**: 33

7. **IdentityName**: Domain\user

8. **DisplayName**: User

9. **RawSid**: NULL

10. **Rights**: 1
11. The protocol server returns a code that the protocol client ignores.
12. The user uses the ACE information to make an implementation-specific authorization decision.

4.4 Create an Entity

This example illustrates how a user can create an Entity in the metadata store.

The example assumes that the previous examples have been successfully executed.

The following actions are carried out:

1. The user requests the protocol client to create an Entity with the name "Customer", the namespace "example.com", and estimated instance count of 100.
2. The protocol client calls the proc_ar_CreateEntity stored procedure using [MS-TDS]:

   ```sql
   DECLARE @return_value int,
   @CreatedId int,
   @ErrorCode int
   EXEC @return_value = proc_ar_CreateEntity
   @Name = N'Customer',
   @Namespace = N'example.com',
   @IsCached = 1,
   @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B',
   @MajorVersion = 1,
   @MinorVersion = 1,
   @BuildVersion = 1,
   @RevisionVersion = 1,
   @SystemId = 33,
   @EstimatedInstanceCount = 100,
   @CacheUsage = 1,
   @ModelId = NULL,
   @CreatedId = @CreatedId OUTPUT,
   @ErrorCode = @ErrorCode OUTPUT
   ```

3. The protocol server creates the Entity in the metadata store.
4. The protocol server copies the ACE of the LobSystem and associates it with the newly created Entity. Finally it sets @ErrorCode to 0.
5. The protocol server returns a return code that the protocol client ignores.
6. The protocol client returns the @CreatedId and @ErrorCode values to the user.
7. The user inspects the @ErrorCode to see if the creation was successful.
8. The user saves the @CreatedId as the MetadataObjectId of the newly created Entity for subsequent use. Assume the value of @CreatedId is 34.

4.5 Activate an Entity

This example illustrates how a user can set a version of an Entity to be active in the metadata store.

This example assumes that the preceding examples have been successfully executed.
The following actions are carried out:

1. The user requests the protocol client to activate Entity with the name "Customer", the namespace "example.com", the PartitionId "0C37852B-34D0-418E-91C6-2AC25AF4BE5B" and a UniqueSessionId of "1E56484c-34d0-418e-91c6-2ac25af4be5b".

2. The protocol client calls the proc_ar_ActivateEntity stored procedure using [MS-TDS]:

   DECLARE @return_value int,
   @Version int,
   @ErrorCode int
   EXEC @return_value = proc_ar_ActivateEntity
   @Name = N'Customer',
   @Namespace = N'example.com',
   @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B',
   @MajorVersion = 1,
   @MinorVersion = 1,
   @BuildVersion = 1,
   @RevisionVersion = 1,
   @UniqueSessionId = '1E56484c-34d0-418e-91c6-2ac25af4be5b',
   @Version = @Version OUTPUT,
   @ErrorCode = @ErrorCode OUTPUT

3. The protocol server checks whether the Entity exists in the metadata store.

4. If it exists, the protocol server marks the Entity as active. All references to the Entity being activated are bound correctly.

5. The protocol server returns a return code that the protocol client ignores.

6. The protocol client returns the @Version and @ErrorCode values to the user.

7. The user inspects the @ErrorCode to see if the operation was successful.

4.6 Read an Entity

This example shows how a user can read an Entity in the metadata store.

The example assumes that the preceding example has been successfully executed.

The following actions are carried out:

1. The user requests the protocol client to read Entity with MetadataObjectId equal to 34.

2. The protocol client calls the proc_ar_GetEntityByld stored procedure using [MS-TDS]:

   DECLARE @return_value int
   EXEC @return_value = proc_ar_GetEntityByld
   @MetadataObjectId = 34,
   @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'

3. The protocol server checks whether an Entity with MetadataObjectId 34 exists in the metadata store.
4. If it exists, the protocol server retrieves the attributes of the stored Entity.

5. The protocol server returns an Entity result set with one row to the protocol client. The columns in the row and the values are as follows:

6. **Id:** 34

7. **EstimatedInstanceCount:** 100

8. **CacheUsage:** 1

9. **SystemId:** 33

10. **Namespace:** example.com

11. **MajorVersion:** 1

12. **MinorVersion:** 1

13. **BuildVersion:** 1

14. **RevisionVersion:** 1

15. **Active:** 1

16. **Name:** Customer

17. **IsCached:** 1

18. **PartitionId:** 0C37852B-34D0-418E-91C6-2AC25AF4BE5B

19. **Version:** 0

20. The protocol server returns a code that the protocol client ignores.

21. The user retrieves the Entity attributes from the result set.

### 4.7 Create Properties for MetadataObjects

This example shows how a user can create Properties for an Entity in the metadata store. The concepts can be applied to any other MetadataObject.

The example assumes that the preceding examples have been successfully executed.

The following actions are carried out:

1. The user requests the protocol client to create a Property for the Entity with MetadataObjectId equal to 34.

2. The protocol client calls the proc_ar_AddOrInsertPropertyForMetadataObjectId stored procedure using [MS-TDS]:

   ```
   DECLARE @return_value int,
   @ErrorCode int
   EXEC @return_value = proc_ar_AddOrInsertPropertyForMetadataObjectId
   @MetadataObjectId = 34,
   @Name = N'DisplayName',
   ```
3. The protocol server checks whether an Entity with MetadataObjectId 34 exists in the metadata store.

4. If it exists, the protocol server creates a new Property called "DisplayName" for the Entity and sets its value to "Customer Details".

5. The protocol server returns a code that the protocol client ignores.

6. The user inspects the @ErrorCode to see whether the operation was successful.

### 4.8 Add Localized Names for MetadataObjects

This example shows how a user can add a localized name for an Entity in the metadata store. The concepts can be applied to any other MetadataObject.

The example assumes that:

- The preceding examples have been successfully executed.
- The user wants to create the localized name for LCID 2058.

The following actions are carried out:

1. The user requests the protocol client to create the localized name for the Entity with MetadataObjectId equal to 34.

2. The protocol client calls the proc_ar_AddOrInsertLocalizedNameForMetadataObjectId stored procedure using [MS-TDS]:

   ```
   DECLARE @return_value int,
   @ErrorCode int
   EXEC @return_value = proc_ar_AddOrInsertLocalizedNameForMetadataObjectId
   @MetadataObjectId = 34,
   @LocalizedName = N'Cliente',
   @LCID = 2058,
   @SettingId = NULL,
   @PartitionId = '0c37852b-34d0-418e-91c6-2ac25af4be5b',
   @ErrorCode = @ErrorCode OUTPUT
   ```

3. The protocol server checks whether an Entity with MetadataObjectId 34 exists in the metadata store.

4. If it exists, the protocol server creates the localized name for LCID 2058 and sets its value to "Cliente".

5. The protocol server returns a code that the protocol client ignores.

6. The user inspects the @ErrorCode to see whether the operation was successful.
4.9 Update an Entity

This example illustrates how a user can update an Entity in the metadata store.

The example assumes that the preceding example has been successfully executed.

The following actions are carried out:

1. The user requests the protocol client to update Entity with MetadataObjectId equal to 34 and change its name from "Customer" to "Buyer".

2. The protocol client calls the `proc_ar_UpdateEntityById` stored procedure using [MS-TDS]. Attributes other than `Name` are supplied with the values obtained when the Entity was read in the preceding example.

   ```
   DECLARE @return_value int,
           @ErrorCode int
   EXEC @return_value = proc_ar_UpdateEntityById
       @Id = 34,
       @Name = N'Buyer',
       @Namespace = N'example.com',
       @IsCached = 1,
       @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B',
       @MajorVersion = 1,
       @MinorVersion = 1,
       @BuildVersion = 1,
       @RevisionVersion = 1,
       @Version = 0,
       @SystemId = 33,
       @EstimatedInstanceCount = 100,
       @CacheUsage = 1,
       @ErrorCode = @ErrorCode OUTPUT
   ```

3. The protocol server checks whether an Entity with MetadataObjectId 34 exists in the metadata store.

4. If it exists, the protocol server compares the value of `@Version` with the value of the stored version for the Entity with MetadataObjectId 34. Because they are same, the protocol server updates all the attribute of the Entity with the supplied values, increments the version counter from 0 to 1 and sets the `@ErrorCode` to 0.

5. The protocol server returns a code that the protocol client ignores.

6. The protocol client returns the `@ErrorCode` and `@Version` values to the user.

7. The user inspects the `@ErrorCode` to see if the update was successful.

8. The user saves the `@Version` value, whose value is 1, for use in subsequent updates to the Entity.

4.10 Delete an Entity

This example illustrates how a user can delete an Entity in the metadata store.

The example assumes that the preceding example has been successfully executed.
The following actions are carried out:

1. The user requests the protocol client to delete **Entity** with MetadataObjectId equal to 34.

2. The protocol client calls the **proc_ar_DeleteEntityById** stored procedure using [MS-TDS].

   ```sql
   DECLARE @return_value int,
   @ErrorCode int
   EXEC @return_value = proc_ar_DeleteEntityById
   @Id = 34,
   @Version = 1,
   @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B',
   @ErrorCode = @ErrorCode OUTPUT
   ```

3. The protocol server checks whether an **Entity** with **MetadataObjectId** 34 exists in the metadata store.

4. If it exists, the protocol server compares the value of @Version with the value of the stored version for the **Entity** with **MetadataObjectId** 34. Because they are same, the protocol server deletes the **Entity** along with the associated Properties, localized names, and ACEs and sets @ErrorCode to zero.

5. The protocol server returns a code that the protocol client ignores.

6. The protocol client returns the @ErrorCode values to the user.

7. The user inspects the @ErrorCode to see whether the deletion was successful.

**4.11 Cache Invalidation**

This example illustrates how a user can invalidate cached MetadataObjects and all relationships after one or more **MetadataObjects** have been created, updated or deleted.

The example assumes that the preceding example has been successfully executed.

The user wants the Entity named "Customer" that is currently reflected in any in-memory cached metadata representations that may be maintained by a protocol client, but has been deleted from the metadata store, to also be removed from the in-memory representations.

The following actions are carried out:

1. The user requests the protocol client to remove all cached **Entities** from memory.

2. The protocol client calls the **proc_ar_BumpCacheInvalidationCounters** stored procedure using [MS-TDS].

   ```sql
   DECLARE @return_value int
   EXEC @return_value = proc_ar_BumpCacheInvalidationCounters
   @CacheLines = 0x00080000,
   @LastModified = 1,
   @PartitionId = '0C37852B-34D0-418E-91C6-2AC25AF4BE5B'
   ```

3. The protocol server increments the object cache version stamp for the **Entity MetadataObjectType**.
4. The protocol server returns a return code that the protocol client ignores.

In parallel to the preceding process, a cache invalidation timer is polling the cache version stamp values in the metadata store periodically. When the timer is signaled, the following actions are carried out:

1. The protocol client timer event handler calls the `proc_ar_GetCacheInvalidationCountersWithCount` stored procedure using [MS-TDS].

   ```
   DECLARE @return_value int
   EXEC @return_value = proc_ar_GetCacheInvalidationCountersWithCount
   @LastModified = 1
   ```

2. The protocol server retrieves the cache version stamp values for all `MetadataObjectTypes` along with how many types for which there are counters.

3. The protocol server returns a Count result set (section 2.2.5.2) with one row to the protocol client. The columns in the row and the values as follows:

   4. **UnnamedColumn**: 1

5. The protocol server returns a Cache Version Stamps result set (section 2.2.5.11) with as many rows as were indicated in the previous step to the protocol client. The columns in the rows and the values are as follows:

   6. **CacheLine**: 8388608
   7. **Counter**: 1
   8. **PartitionId**: 0C37852B-34D0-418E-91C6-2AC25AF48E5B
   9. **LastModified**: 1

10. The protocol server returns a code that the protocol client ignores.

11. The protocol client compares the returned counter values with the values it read when the timer was previously signaled, and finds that the Cache Version Stamp and the Relationship Cache Version Stamp values are different. In response, the protocol client deletes the cached `Entity` references and the cached `Entity MetadataObjects` from memory.
5 Security

5.1 Security Considerations for Implementers

Interactions with SQL are susceptible to tampering and other forms of security risks. Implementers are advised to sanitize input parameters for stored procedures before invoking the stored procedure.

5.2 Index of Security Parameters

None.
6 Appendix A: Product Behavior

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

- Microsoft® SharePoint® Foundation 2013 Preview

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 SHOULDN'T prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.1.5: SharePoint Foundation 2010 decides this by user input, and assumes the MetadataObject is frequently used, unless specified otherwise.

<2> Section 2.2.1.14: The application that uses the protocol client typically uses this ordering as guidance in an implementation-specific algorithm that represents the Actions in the user interface. Such a use of Position is outside the scope of this protocol.

<3> Section 2.2.1.15: The application that uses the protocol client typically uses this value as a guidance to represent the Action in the user interface. Such a use of IsDisplayed is outside the scope of this protocol.

<4> Section 2.2.1.16: The application that uses the protocol client typically uses this value as guidance on creating new user interface context when the Action is executed. Such a use of IsOpenedInNewWindow is outside the scope of this protocol.

<5> Section 2.2.1.17: The application that uses the protocol client typically uses the resource in the specified location to represent the Action in the user interface possibly along with the localized name of the Action. Such a use of Icon is outside the scope of this protocol.

<6> Section 2.2.1.18: The application that uses the protocol client typically sets the parameter values to corresponding ActionParameters and executes the command. Such a use of URL is outside the scope of this protocol.

<7> Section 2.2.1.30: A Business Logic Module that conforms to the [ECMA-335] specification and is understood by the .NET Framework.

<8> Section 2.2.1.31: A Business Logic Module that conforms to the [ECMA-335] specification and is understood by the .NET Framework.

<9> Section 3.2.5.1: SharePoint Foundation 2010 does not retry operations.

<10> Section 3.2.5.1: SharePoint Foundation 2010 does not retry operations.

<11> Section 3.2.5.2: SharePoint Foundation 2010 does not retry operations.

<12> Section 3.2.5.2: SharePoint Foundation 2010 does not retry operations.

<13> Section 3.2.5.3: SharePoint Foundation 2010 does not retry operations.

<14> Section 3.2.5.3: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.4: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.4: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.5: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.5: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.10: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.10: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.11: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.11: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.12: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.12: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.13: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.13: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.14: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.14: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.15: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.15: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.16: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.16: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.17: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.17: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.18: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.18: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.19: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.19: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.20: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.20: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.21: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.21: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.22: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.22: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.23: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.39: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.40: Under some certain circumstances, SharePoint Foundation 2010 does not mark another MethodInstance as the default MethodInstance upon return from this stored procedure. Protocol client MUST NOT rely on this behavior.

Section 3.2.5.41: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.42: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.43: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.44: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.45: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.46: Windows SharePoint Services currently sets the @ErrorCode to 0 and returns a result set with zero rows in this case.

Section 3.2.5.47: SharePoint Foundation 2010 currently ignores this and returns count of all Entities in the LobSystem.

Section 3.2.5.48: SharePoint Foundation 2010 always returns an empty result set.

Section 3.2.5.49: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.50: SharePoint Foundation 2010 does not retry operations.
Section 3.2.5.126: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.127: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.128: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.129: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.130: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.131: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.132: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.133: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.134: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.135: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.136: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.137: SharePoint Foundation 2010 does not retry operations.

Section 3.2.5.138: SharePoint Foundation 2010 distinguishes between several ways that a string can fail to meet the specification in [MS-BDCMFFS] section 2.1.5.5. It is not necessary for interoperability to distinguish between these error codes. The specific causes of these errors are the following:

- 02: Backslash (\) (%x5C) occurs outside of an EscapedDot, EscapedBracket, or EscapedSlash.
- **03**: An Indexer is followed by a token other than a **FieldAccess**.
- **04**: Index contains a character that was not a **DIGIT**.
- **05**: Period (\%) (\%x2E) is immediately followed by another period.
- **07**: The last character is "[" (\%x5B), "." (\%x2E), or "\" (\%x5C)

<130> **Section 3.2.5.139**: SharePoint Foundation 2010 does not retry operations.
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Abstract data model</td>
<td>180</td>
</tr>
<tr>
<td>client</td>
<td>180</td>
</tr>
<tr>
<td>MetadataObject caching</td>
<td>180</td>
</tr>
<tr>
<td>server</td>
<td>56</td>
</tr>
<tr>
<td>Access Control Entry result set</td>
<td>50</td>
</tr>
<tr>
<td>Access Control Entry simple type</td>
<td>25</td>
</tr>
<tr>
<td>Action Parameter result set</td>
<td>54</td>
</tr>
<tr>
<td>Action result set</td>
<td>34</td>
</tr>
<tr>
<td>Action simple type</td>
<td>31</td>
</tr>
<tr>
<td>ActionParameter simple type</td>
<td>32</td>
</tr>
<tr>
<td>Activating an Entity example</td>
<td>184</td>
</tr>
<tr>
<td>Activation Errors result set</td>
<td>51</td>
</tr>
<tr>
<td>Adding Localized Names for MetadatObjects example</td>
<td>187</td>
</tr>
<tr>
<td>Applicability</td>
<td>12</td>
</tr>
<tr>
<td>Association Group result set</td>
<td>37</td>
</tr>
<tr>
<td>Association Member result set</td>
<td>38</td>
</tr>
<tr>
<td>Association result set</td>
<td>36</td>
</tr>
<tr>
<td>Association simple type</td>
<td>29</td>
</tr>
<tr>
<td>AssociationGroup simple type</td>
<td>31</td>
</tr>
<tr>
<td>AssociationReference result set</td>
<td>38</td>
</tr>
<tr>
<td>AssociationReference simple type</td>
<td>31</td>
</tr>
<tr>
<td>Attribute groups - overview</td>
<td>55</td>
</tr>
<tr>
<td>Attributes - overview</td>
<td>55</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>Binary structures - overview</td>
<td>34</td>
</tr>
<tr>
<td>Bit fields</td>
<td>33</td>
</tr>
<tr>
<td>CacheLine</td>
<td>33</td>
</tr>
<tr>
<td>Bit fields - overview</td>
<td>33</td>
</tr>
<tr>
<td>BuildVersion field</td>
<td>15</td>
</tr>
<tr>
<td>CacheLine bit field</td>
<td>33</td>
</tr>
<tr>
<td>CacheUsage field</td>
<td>15</td>
</tr>
<tr>
<td>Change tracking</td>
<td>198</td>
</tr>
<tr>
<td>Change tracking example</td>
<td>198</td>
</tr>
<tr>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>abstract data model</td>
<td>180</td>
</tr>
<tr>
<td>higher-layer triggered events</td>
<td>181</td>
</tr>
<tr>
<td>initialization</td>
<td>180</td>
</tr>
<tr>
<td>local events</td>
<td>181</td>
</tr>
<tr>
<td>message processing</td>
<td>181</td>
</tr>
<tr>
<td>MetadataObject caching</td>
<td>180</td>
</tr>
<tr>
<td>overview</td>
<td>180</td>
</tr>
<tr>
<td>sequencing rules</td>
<td>181</td>
</tr>
<tr>
<td>timer events</td>
<td>181</td>
</tr>
<tr>
<td>timers</td>
<td>180</td>
</tr>
<tr>
<td>Common data types</td>
<td>14</td>
</tr>
<tr>
<td>Common fields</td>
<td></td>
</tr>
<tr>
<td>BuildVersion</td>
<td>15</td>
</tr>
<tr>
<td>CacheUsage</td>
<td>15</td>
</tr>
<tr>
<td>DefaultValue</td>
<td>21</td>
</tr>
<tr>
<td>Direction</td>
<td>20</td>
</tr>
<tr>
<td>EstimatedInstanceCount</td>
<td>15</td>
</tr>
<tr>
<td>FilterField</td>
<td>17</td>
</tr>
<tr>
<td>FilterType</td>
<td>16</td>
</tr>
<tr>
<td>Icon</td>
<td>16</td>
</tr>
<tr>
<td>Id</td>
<td>14</td>
</tr>
<tr>
<td>IdentifierTypeName</td>
<td>18</td>
</tr>
<tr>
<td>Index</td>
<td>16</td>
</tr>
<tr>
<td>IsActive</td>
<td>15</td>
</tr>
<tr>
<td>IsCached</td>
<td>14</td>
</tr>
<tr>
<td>IsDefault</td>
<td>23</td>
</tr>
<tr>
<td>IsDisplayed</td>
<td>16</td>
</tr>
<tr>
<td>IsOpenedInNewWindow</td>
<td>16</td>
</tr>
<tr>
<td>IsReverse</td>
<td>23</td>
</tr>
<tr>
<td>IsStatic</td>
<td>22</td>
</tr>
<tr>
<td>MajorVersion</td>
<td>15</td>
</tr>
<tr>
<td>MetadataRights</td>
<td>22</td>
</tr>
<tr>
<td>MethodInstanceType</td>
<td>18</td>
</tr>
<tr>
<td>MethodLobName</td>
<td>22</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>15</td>
</tr>
<tr>
<td>Name</td>
<td>14</td>
</tr>
<tr>
<td>Namespace</td>
<td>14</td>
</tr>
<tr>
<td>overview</td>
<td>14</td>
</tr>
<tr>
<td>PartitionId</td>
<td>14</td>
</tr>
<tr>
<td>Position</td>
<td>16</td>
</tr>
<tr>
<td>RevisionVersion</td>
<td>15</td>
</tr>
<tr>
<td>SessionId</td>
<td>23</td>
</tr>
<tr>
<td>SettingId</td>
<td>14</td>
</tr>
<tr>
<td>SystemData</td>
<td>22</td>
</tr>
<tr>
<td>SystemType</td>
<td>21</td>
</tr>
<tr>
<td>ThrottleConfigEnabled</td>
<td>24</td>
</tr>
<tr>
<td>ThrottleScope</td>
<td>23</td>
</tr>
<tr>
<td>ThrottleType</td>
<td>24</td>
</tr>
<tr>
<td>TypeDescriptorFlags</td>
<td>21</td>
</tr>
<tr>
<td>TypeDescriptorInterpretation</td>
<td>20</td>
</tr>
<tr>
<td>TypeDescriptorLobName</td>
<td>20</td>
</tr>
<tr>
<td>TypeDescriptorTypeName</td>
<td>20</td>
</tr>
<tr>
<td>URL</td>
<td>16</td>
</tr>
<tr>
<td>Complex types - overview</td>
<td>55</td>
</tr>
<tr>
<td>Count result set</td>
<td>35</td>
</tr>
<tr>
<td>Creating a LobSystem example</td>
<td>182</td>
</tr>
<tr>
<td>Creating an Entity example</td>
<td>184</td>
</tr>
<tr>
<td>Creating properties for MetadatObjects example</td>
<td>186</td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Data model - abstract</td>
<td></td>
</tr>
<tr>
<td>client</td>
<td>180</td>
</tr>
<tr>
<td>MetadataObject caching</td>
<td>180</td>
</tr>
<tr>
<td>server</td>
<td>56</td>
</tr>
<tr>
<td>Data types</td>
<td></td>
</tr>
<tr>
<td>Access Control Entry simple type</td>
<td>25</td>
</tr>
<tr>
<td>Action simple type</td>
<td>31</td>
</tr>
<tr>
<td>ActionParameter simple type</td>
<td>32</td>
</tr>
<tr>
<td>Association simple type</td>
<td>29</td>
</tr>
<tr>
<td>AssociationGroup simple type</td>
<td>31</td>
</tr>
</tbody>
</table>
AssociationReference simple type 31
Cache Version Stamp simple type 32
common 14
DataClass simple type 27
DefaultValue simple type 31
Entity simple type 27
FilterDescriptor simple type 30
Identifier simple type 28
LobSystem simple type 26
LobSystemInstance simple type 26
Localized Name simple type 25
MetadataObject simple type 24
Method simple type 28
MethodInstance simple type 29
Model simple type 26
Parameter simple type 29
Property simple type 25
Throttle Configuration Setting simple type 32
TypeDescriptor simple type 30

Data types - simple
Access Control Entry 25
Action 31
ActionParameter 32
Association 29
AssociationGroup 31
AssociationReference 31
Cache Version Stamp 32
DataClass 27
DefaultValue 31
Entity 27
FilterDescriptor 30
Identifier 28
LobSystem 26
LobSystemInstance 26
Localized Name 25
MetadataObject 24
Method 28
MethodInstance 29
Model 26
overview 24
Parameter 29
Property 25
Throttle Configuration Setting 32
TypeDescriptor 30

DataClass result set 41
DataClass simple type 27
DefaultValue field 21
DefaultValue simple type 31
DefaultValue result set 42
Deleting an Entity example 188
Direction field 20

E
Elements - overview 55
Entity Name result set 44
Entity result set 43
Entity simple type 27
EstimatedInstanceCount field 15
Events
  local - client 181
  local - server 180
timer - client 181
timer - server 179

Examples
  activating an Entity 184
  adding Localized Names for MetadataObjects 187
  cache invalidation 189
  creating a LobSystem 182
  creating an Entity 184
  creating properties for MetadataObjects 186
  deleting an Entity 188
  overview 182
  reading an Entity 185
  reading the security information of a MetadataObject 183
  setting the security information of a MetadataObject 182
  updating an Entity 188

F
Fields – common
BuildVersion 15
CacheUsage 15
DefaultValue 21
Direction 20
EstimatedInstanceCount 15
FilterField 17
FilterType 16
Icon 16
Id 14
IdentifierTypeName 18
Index 16
IsActive 15
IsCached 14
IsDefault 23
IsDisplayed 16
IsOpenedInNewWindow 16
IsReverse 23
IsStatic 22
MajorVersion 15
MetadataRights 22
MethodInstanceType 18
MethodLobName 22
MinorVersion 15
Name 14
Namespace 14
overview 14
PartitionId 14
Position 16
RevisionVersion 15
SessionId 23
SettingId 14
SystemData 22
SystemType 21
ThrottleConfigEnabled 24
ThrottleScope 23
ThrottleType 24
TypeDescriptorFlags 21
TypeDescriptorInterpretation 20
TypeDescriptorLobName 20
TypeDescriptorTypeName 20
URL 16

[MS-BDCDPS2] — v20120630

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
IsCached 14
IsDefault 23
IsDisplayed 16
IsOpenedInNewWindow 16
IsReverse 23
IsStatic 22
MajorVersion 15
MetadataRights 22
MethodInstanceType 18
MethodLobName 22
MinorVersion 15
Name 14
Namespace 14
overview 14
PartitionId 14
Position 16
RevisionVersion 15
SessionId 23
SettingId 14
SystemData 22
SystemType 21
ThrottleConfigEnabled 24
ThrottleScope 23
TypeDescriptorFlags 21
TypeDescriptorInterpretation 20
TypeDescriptorLobName 20
TypeDescriptorTypeName 20
URL 16
MetadataCatalog result set 35
MetadataObject simple type 24
MetadataRights field 22
Method result set 46
Method simple type 28
MethodInstance result set 46
MethodInstance simple type 29
MethodInstanceType field 18
MethodLobName field 22
Methods
proc_ar_ActivateEntity 62
proc_ar_AddEntity 64
proc_ar_AddOrInsertLocalizedNameForMetadataObjectId 64
proc_ar_AddOrInsertPropertyForMetadataObjectId 65
proc_ar_BulkSwitchActive 66
proc_ar_BumpCacheInvalidationCounters 68
proc_ar_CheckPathInMethodInstances 179
proc_ar_ClearAccessControlEntriesForMetadataObject 69
proc_ar_CopyAccessControlEntriesForMetadataObject 69
proc_ar_CopyAccessControlEntriesForMetadataObjectIdAndSetting 178
proc_ar_CopyAccessControlEntriesForSettings 70
proc_ar_CreateAction 70
proc_ar_CreateActionParameter 71
proc_ar_CreateAdministrationMetadataCatalog 73
proc_ar_CreateAssociation 73
proc_ar_CreateAssociationGroup 75
proc_ar_CreateAssociationReference 76
proc_ar_CreateEntity 78
proc_ar_CreateFilterDescriptor 79
proc_ar_CreateIdentifier 81
proc_ar_CreateMethod 82
proc_ar_CreateMethodInstance 83
proc_ar_CreateModel 85
proc_ar_CreateParameter 86
proc_ar_CreateSystem 87
proc_ar_CreateSystemInstance 88
proc_ar_CreateTypeDescriptor 89
proc_ar_DeactivateEntity 92
proc_ar_DeleteActionById 93
proc_ar_DeleteActionParameterById 94
proc_ar_DeleteAdministrationMetadataCatalog 95
proc_ar_DeleteAssociationByObjectId 96
proc_ar_DeleteAssociationGroupByObjectId 97
proc_ar_DeleteAssociationReferenceByObjectId 98
proc_ar_DeleteDefaultValue 99
proc_ar_DeleteEntityById 100
proc_ar_DeleteFilterDescriptorByObjectId 101
proc_ar_DeleteIdentifierByObjectId 102
proc_ar_DeleteLocalizedNameForMetadataObjectByLCID 103
proc_ar_DeleteLocalizedNamesByMetadataObjectId 104
proc_ar_DeleteMethodById 105
proc_ar_DeleteMethodInstanceById 106
proc_ar_DeleteModelById 107
proc_ar_DeleteParameterById 108
proc_ar_DeletePropertiesById 109
proc_ar_DeletePropertyForMetadataObjectId 110
proc_ar_DeleteSystemById 111
proc_ar_DeleteSystemInstanceById 112
proc_ar_DeleteTypeDescriptorById 113
proc_ar_GetAccessControlEntriesForMetadataObject 114
proc_ar_GetActionById 115
proc_ar_GetActionParameterById 115
proc_ar_GetActionParametersForActionWithCount 115
proc_ar_GetActionsForEntityWithCount 116
proc_ar_GetAdministrationMetadataCatalogByObject 116
proc_ar_GetAdministrationMetadataCatalogByPartitionId 117
proc_ar_GetAllLocalizedNamesForMetadataObjectWithCount 117
proc_ar_GetAllMergedLocalizedNamesForMetadataObjectWithCount 118
proc_ar_GetAllPartitionIds 118
proc_ar_GetAllSlicesForMetadataObjectId 118
proc_ar_GetAssociationByObjectId 119
proc_ar_GetAssociationGroupByObjectId 119
proc_ar_GetAssociationGroupsForEntityWithCount 120
proc_ar_GetAssociationMembersInRoleWithCount 120
proc_ar_GetAssociationReferencesForAssociationGroupWithCount 121
proc_ar_GetAssociationsForDataClassWithCount 121
proc_ar_GetAssociationsForEntityAndRoleWithCount 122
proc_ar_GetAssociationsForMethodWithCount 123
proc_ar_GetCacheInvalidationCountersWithCount 123
proc_ar_GetChildTypeDescriptorsForTypeDescriptorWithCount 125
proc_ar_GetDataClassById 124
proc_ar.GetDataClassesForSystemWithCount 124
proc_ar_GetDefaultValuesForTypeDescriptor 125
proc_ar_GetEntitiesForAssociationAndRoleWithCount 126
proc_ar_GetEntitiesForSystemCount 126
proc_ar_GetEntitiesForSystemWithCount 127
proc_ar_GetEntitiesLikeNameAndNamespace 127
proc_ar_GetEntitiesReferencedByModelId 128
proc_ar_GetEntityById 129
proc_ar_GetEntityNamesForAssociationAndRole 130
proc_ar_GetEntityWithNameAndNamespace 130
proc_ar_GetEntityWithNameAndNamespaceAndVersion 131
proc_ar_GetFilterDescriptorById 132
proc_ar_GetFilterDescriptorsForMethodWithCount 132
proc_ar_GetIdentifierById 133
proc_ar_GetIdentifiersForEntityWithCount 133
proc_ar_GetMergedPropertiesForMetadataObject 133
proc_ar_GetMethodById 134
proc_ar_GetMethodInstanceById 134
proc_ar_GetMethodInstancesForDataClassWithCount 135
proc_ar_GetMethodInstancesForMethodWithCount 135
proc_ar_GetMethodsForDataClassWithCount 135
proc_ar_GetModelsByEntityId 136
proc_ar_GetModelsByName 136
proc_ar_GetParametersForMethodWithCount 137
proc_ar_GetPropertiesForMetadataObject 137
proc_ar_GetRootTypeDescriptorForParameter 138
proc_ar_GetSafetyNetConfigs 139
proc_ar_GetSystemById 140
proc_ar_GetSystemDataBySystemId 140
proc_ar_GetSystemForParameterId 141
proc_ar_GetSystemForTypeDescriptorId 141
proc_ar_GetSystemInstanceById 142
proc_ar_GetSystemInstancesForSystemWithCount 142
proc_ar_GetSystemsLikeNameWithCount 143
proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId 143
proc_ar_GetTypeById 144
proc_ar_GetTypeDescriptorById 144
proc_ar_GetTypeDescriptorForDottedPath 144
proc_ar_GetTypeDescriptorsByNameAndParameter 144
proc_ar_GetTypeDescriptorsForFilterDescriptorWithCount 145
proc_ar_GetViewByMethodInstance 145
proc_ar_IsMethodInstantiated 146
proc_ar_IsParameterReferencedByMethodInstance 146
proc_ar_RemoveEntity 147
proc_ar_RemoveSafetyNetConfig 148
proc_ar_RetrieveProgress 148
proc_ar_SetAccessControlEntryForMetadataObject 149
proc_ar_SetDefaultAction 149
proc_ar_SetDefaultValuesForTypeDescriptor 150
proc_ar_SetSafetyNetConfig 151
proc_ar_SetSystemDataBySystemId 152
proc_ar_UpdateActionById 152
proc_ar_UpdateActionParameterById 153
proc_ar_UpdateAssociationById 153
proc_ar_UpdateAssociationGroupById 154
proc_ar_UpdateEntityType 155
proc_ar_UpdateEntityById 155
proc_ar_UpdateEntityWithId 156
proc_ar_UpdateFilterDescriptorById 156
proc_ar_UpdateFilterDescriptorsForMethodWithCount 156
proc_ar_UpdateIdentifierById 157
proc_ar_UpdateMethodById 158
proc_ar_UpdateMethodInstanceById 158
proc_ar_UpdateMethodInstancesForMethodWithCount 158
proc_ar_UpdateModelById 159
proc_ar_UpdateParameterById 159
proc_ar_UpdateProgress 159
proc_ar_UpdateSystemById 160
proc_ar_UpdateSystemInstanceById 160
proc_ar_UpdateSystemInstancesForSystemWithCount 160
proc_ar_UpdateTypeDescriptorById 161
MinorVersion field 15
Model result set 47
Model simple type 26

N
Name field 14
Namespace field 14
Namespaces 55
Normative references 11

O
Overview (synopsis) 11

P
Parameter result set 47
Parameter simple type 29
Parameters - security index 191
Partition result set 36
PartitionId field 14
Position field 16
Preconditions 12
Prerequisites 12
proc_ar_ActivateEntity method 62
proc_ar_AddEntity method 64
proc_ar_AddEntity method 64
proc_ar_AddEntity method 64
proc_ar_AddOrInsertLocalizedNameForMetadataObjectId method 64
proc_ar_AddOrInsertPropertyForMetadataObjectId method 65
proc_ar_BulkSwitchActive method 66
proc_ar_GetMethodInstancesForDataClassWithCount method 135
proc_ar_GetMethodInstancesForMethodWithCount method 135
proc_ar_GetMethodsForDataClassWithCount method 135
proc_ar_GetModelById method 136
proc_ar_GetModelsByEntityId method 136
proc_ar_GetModelByParameterId method 137
proc_ar_GetParametersForMethodWithCount method 138
proc_ar_GetPropertiesForMetadataObject method 138
proc_ar_GetRootTypeDescriptorForParameter method 139
proc_ar_GetSafetyNetConfigs method 140
proc_ar_GetSystemById method 140
proc_ar_GetSystemByName method 140
proc_ar_GetSystemDataBySystemId method 141
proc_ar_GetSystemForParameterId method 141
proc_ar_GetSystemForTypeDescriptorId method 141
proc_ar_GetSystemInstanceById method 142
proc_ar_GetSystemInstancesForSystemWithCount method 142
proc_ar_GetSystemsLikeNameWithCount method 143
proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId method 143
proc_arGetTypeById method 144
proc_arGetTypeDescriptorById method 144
proc_arGetTypeDescriptorForDottedPath method 144
proc_arGetTypeDescriptorsByNameAndParameter method 144
proc_arGetTypeDescriptorsForFilterDescriptorWithCount method 145
proc_arGetTypeDescriptorByMethodInstance method 145
proc_ar_TypeDescriptorInstantiated method 146
proc_ar_GetSafetyNetConfigReferencedByMethodInstance method 146
proc_ar_RemoveEntity method 147
proc_ar_RemoveSafetyNetConfig method 148
proc_ar_RetrieveProgress method 148
proc_ar_SetAccessControlEntryForMetadataObject method 149
proc_ar_SetDefaultAction method 149
proc_ar_SetDefaultValueForTypeDescriptor method 150
proc_ar_setSafetyNetConfig method 151
proc_ar_setSystemDataBySystemId method 152
proc_ar_UpdateActionByMethod method 152
proc_ar_UpdateActionParameterByMethod method 154
proc_ar_UpdateAssociationByMethod method 155
proc_ar_UpdateAssociationGroupByMethod method 157
proc_ar_UpdateEntityByMethod method 158
proc_ar_UpdateFilterDescriptorByMethod method 160
proc_ar_UpdateIdentifierByMethod method 162
proc_ar_UpdateMethodById method 163
proc_ar_UpdateMethodInstancesForById method 164
proc_ar_UpdateModelById method 167
proc_ar_UpdateParameterById method 168
proc_ar_UpdateProgress method 169
proc_ar_UpdateSystemById method 170
proc_ar_UpdateSystemInstanceByMethod method 171
proc_ar_UpdateTypeDescriptorByMethod method 172
Product behavior 192
Progress result set 50
Property result set 45
Property simple type 25

R

Reading an Entity example 185
Reading the security information of a MetadataObject example 183
References 11
Informative 11
normative 11
Relationship to other protocols 12
Result sets - messages
Access Control Entry 50
Action 34
Action Parameter 54
Activation Errors 51
Association 36
Association Group 37
Association Member 38
AssociationReference 38
Cache Version Stamps 39
Count 35
DataClass 41
DefaultValues 42
Entity 43
Entity Name 44
FilterDescriptor 44
Id 50
Identifier 45
LocalizedName 36
MetadataCatalog 35
Method 46
MethodInstance 46
Model 47
Parameter 47
Partition 36
Progress 50
Property 45
Setting 36
System 49
System Data 49
SystemInstance 49
Throttle Setting 48
TypeDescriptor 39
Result sets - overview 34
RevisionVersion field 15

S

Security
implementer considerations 191
parameter index 191
Sequencing rules
client 181
server 62
Server
abstract data model 56
higher-layer triggered events 62
initialization 62
local events 180
message processing 62
overview 56
proc_ar_ActivateEntity method 62
proc_ar_AddEntity method 64
proc_ar_AddOrInsertLocalizedNameForMetadataObjectId method 64
proc_ar_AddOrInsertPropertyForMetadataObjectId method 65
proc_ar_BulkSwitchActive method 66
proc_ar_BumpCacheInvalidationCounters method 68
proc_ar_CheckPathInMethodInstances method 79
proc_ar_ClearAccessControlEntriesForMetadataObject method 69
proc_ar_CopyAccessControlEntriesForMetadataObjectId method 69
proc_ar_CopyAccessControlEntriesForMetadataObjectIdAndSetting method 178
proc_ar_CopyAccessControlEntriesForSettings method 70
proc_ar_CreateAction method 70
proc_ar_CreateActionParameter method 71
proc_ar_CreateAdministrationMetadataCatalog method 73
proc_ar_CreateAssociation method 73
proc_ar_CreateAssociationGroup method 75
proc_ar_CreateAssociationReference method 76
proc_ar_CreateEntity method 78
proc_ar_CreateFilterDescriptor method 79
proc_ar_CreateIdentifier method 81
proc_ar_CreateModel method 85
proc_ar_CreateParameter method 86
proc_ar_CreateSystem method 87
proc_ar_CreateSystemInstance method 88
proc_ar_CreateTypeDescriptor method 89
proc_ar_DeactivateEntity method 92
proc_ar_DeleteActionById method 93
proc_ar_DeleteActionParameterById method 94
proc_ar_DeleteAdministrationMetadataCatalog method 95
proc_ar_DeleteAssociationById method 96
proc_ar_DeleteAssociationGroupById method 97
proc_ar_DeleteAssociationReferenceById method 98
proc_ar_DeleteDefaultValue method 99
proc_ar_DeleteEntityById method 100
proc_ar_DeleteFilterDescriptorById method 101
proc_ar_DeleteIdentifierById method 102
proc_ar_DeleteLocalizedNamesByMetadataObjectId method 103
proc_ar_DeleteMethodByMetadataObjectId method 104
proc_ar_DeleteMethodByMetadataObjectId method 105
proc_ar_DeleteMethodInstanceByMetadataObjectId method 106
proc_ar_DeleteModelByMetadataObjectId method 107
proc_ar_DeleteParameterByMetadataObjectId method 108
proc_ar_DeletePropertiesByMetadataObjectId method 109
proc_ar_DeletePropertyForMetadataObjectId method 110
proc_ar_DeleteSystemByMetadataObjectId method 111
proc_ar_DeleteSystemInstanceByMetadataObjectId method 112
proc_ar_GetAccessControlEntriesForMetadataObjectId method 113
proc_ar_GetActionById method 115
proc_ar_GetActionParameterById method 115
proc_ar_GetActionParametersForActionWithCount method 115
proc_ar_GetActionsForEntityWithCount method 116
proc_ar_GetAdministrationMetadataCatalogByMetadataObjectId method 116
proc_ar_GetAdministrationMetadataCatalogByPartitionId method 117
proc_ar_GetAllLocalizedNameByMetadataObjectIdWithCount method 117
proc_ar_GetAllMergedLocalizedNameByMetadataObjectIdWithCount method 118
proc_ar_GetAllSlicesForMetadataObjectId method 118
proc_ar_GetAssociationById method 119
proc_ar_GetAssociationGroupById method 119
proc_ar_GetAssociationGroupsForEntityWithCount method 120
proc_ar_GetAssociationMembersInRoleWithCount method 120
proc_ar_GetAssociationReferencesForAssociationGroupWithCount method 121
proc_ar_GetAssociationsForDataClassWithCount method 121
proc_ar_GetAssociationsForEntityAndRoleWithCount method 122
proc_ar_GetAssociationsForEntityAndRoleWithCount method 122
proc_ar_GetAssociationsForEntityWithCount method 123
proc_ar_GetAssociationsForSystemWithCount method 123
proc_ar_GetCacheInvalidationCountersWithCount method 123
proc_ar_GetChildTypeDescriptorsForTypeDescriptorWithCount method 123
proc_ar_GetDataClassById method 124
proc_ar_GetDataClassesForSystemWithCount method 124
proc_ar_GetDefaultValuesForTypeDescriptor method 125
proc_ar_GetEntitiesForAssociationAndRoleWithCount method 126
proc_ar_GetEntitiesForSystemWithCount method 126
proc_ar_GetEntitiesForSystemWithCount method 127
proc_ar_GetEntitiesLikeNameAndNamespace method 127
proc_ar_GetEntitiesReferencedByModelId method 128
proc_ar_GetEntityById method 129
proc_ar_GetEntityNamesForAssociationAndRole method 130
proc_ar_GetEntityWithNamespaceAndVersion method 131
proc_ar_GetFilterDescriptorById method 131
proc_ar_GetFilterDescriptorsForMethodWithCount method 132
proc_ar_GetIdentifierById method 132
proc_ar_GetIdentifiersForEntityWithCount method 133
proc_ar_GetMergedPropertiesForMetadataObject method 133
proc_ar_GetMethodById method 134
proc_ar_GetMethodInstanceById method 134
proc_ar_GetMethodInstancesForDataClassWithCount method 135
proc_ar_GetMethodInstancesForMethodWithCount method 135
proc_ar_GetMethodsForDataClassWithCount method 135
proc_ar_GetModelById method 136
proc_ar_GetModelsByEntityId method 136
proc_ar_GetModelsByName method 137
proc_ar_GetParameterById method 137
proc_ar_GetParametersForMethodWithCount method 138
proc_ar_GetPropertiesForMetadataObject method 138
proc_ar_GetRootTypeDescriptorForParameter method 139
proc_ar_GetSafetyNetConfigs method 140
proc_ar_GetSystemById method 140
proc_ar_GetSystemByNamespace method 140
proc_ar_GetSystemDataBySystemId method 141
proc_ar_GetSystemForParameterId method 141
proc_ar_GetSystemForTypeDescriptorId method 141
proc_ar_GetSystemInstanceById method 142
proc_ar_GetSystemInstancesForSystemWithCount method 142
proc_ar_GetSystemsLikeNameWithCount method 143
proc_ar_GetSystemsReferencedByEntitiesAssociatedWithModelId method 143
proc_ar_GetTypeById method 176
proc_arGetTypeDescriptorById method 144
proc_arGetTypeDescriptorForDottedPath method 177
proc_arGetTypeDescriptorsByNameAndParameter method 144
proc_arGetTypeDescriptorsForFilterDescriptorWithCount method 145
proc_ar_GetViewByMethodInstance method 145
proc_ar_IsMethodInstantiated method 146
proc_ar_IsParameterReferencedByMethodInstance method 146
proc_ar_RemoveEntity method 147
proc_ar_RemoveSafetyNetConfig method 148
proc_ar_RetrieveProgress method 148
proc_ar_SetAccessControlEntryForMetadataObject method 149
proc_ar_SetDefaultAction method 149
proc_ar_SetDefaultValuesForTypeDescriptor method 150
proc_ar_SetSafetyNetConfig method 151
proc_ar_SetSystemDataBySystemId method 152
proc_ar_UpdateActionByMethodId method 152
proc_ar_UpdateActionByParameterId method 154
proc_ar_UpdateAssociationByMethodId method 155
proc_ar_UpdateAssociationGroupByMethodId method 157
proc_ar_UpdateEntityByMethodId method 158
proc_ar_UpdateFilterDescriptorByMethodId method 160
proc_ar_UpdateIdentifierByMethodId method 160
proc_ar_UpdateMethodByMethodId method 163
proc_ar_UpdateMethodInstanceByMethodId method 164
proc_ar_UpdateModelByMethodId method 167
proc_ar_UpdateParameterByMethodId method 168
proc_ar_UpdateProgress method 169
proc_ar_UpdateSystemByMethodId method 170
proc_ar_UpdateSystemStatementByMethodId method 171
proc_ar_UpdateTypeDescriptorByMethodId method 172
settingId field 23
Setting the result set 36
Setting the security information of a MetadataObject example 182
SessionId field 14
Simple data types
Access Control Entry 25
Action 31
ActionParameter 32
Association 29
AssociationGroup 31
AssociationReference 31
Cache Version Stamp 32
DataClass 27
DefaultValue 31
Entity 27
FilterDescriptor 30
Identifier 28
LobSystem 26
LobSystemInstance 26
Localized Name 25
MetadataObject 24
Method 28
MethodInstance 29
Model 26
overview 24
Parameter 29
Property 25
Throttle Configuration Setting 32
TypeDescriptor 30
Simple types - overview 55
Standards assignments 13
Structures

- binary 34
- table and view 55
- XML 55

**System Data result set** 49
**System result set** 49
**SystemData field** 22
**SystemInstance result set** 49
**SystemType field** 21

T

**Table structures - overview** 55
**Throttle Configuration Setting simple type** 32
**Throttle Setting result set** 48
**ThrottleConfigEnabled field** 24
**ThrottleScope field** 23
**ThrottleType field** 24

Timer events

- client 181
- server 179

Timers

- client 180
- server 62

**Tracking changes** 198

Transport 14

Triggered events - higher-layer

- client 181
- server 62

**TypeDescriptor result set** 39
**TypeDescriptor simple type** 30
**TypeDescriptorFlags field** 21
**TypeDescriptorInterpretation field** 20
**TypeDescriptorLobName field** 20
**TypeDescriptorTypeName field** 20

Types

- complex 55
- simple 55

U

**Updating an Entity example** 188
**URL field** 16

V

**Vendor-extensible fields** 13
**Versioning** 13
**View structures - overview** 55

X

**XML structures** 55