

[MC-IISA]: Internet Information Services (IIS) Application Host COM Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- This protocol 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 protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- Microsoft does not claim any trade secret rights in this documentation.
- Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. If you are interested in obtaining a patent license, please contact protocol@microsoft.com.
- 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.
- All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does 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.

Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	0.1	Major	Initial Availability
07/03/2007	0.2	Minor	Updated the technical content.
08/10/2007	0.2.1	Editorial	Revised and edited the technical content.
09/28/2007	0.3	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
10/23/2007	0.4	Minor	Updated the technical content.
11/30/2007	0.5	Minor	Updated the technical content.
01/25/2008	0.5.1	Editorial	Revised and edited the technical content.

Table of Contents

1	Introduction	8
1.1	Glossary	8
1.2	References	8
1.2.1	Normative References	8
1.2.2	Informative References.....	8
1.3	Protocol Overview (Synopsis).....	9
1.4	Relationship to Other Protocols.....	9
1.5	Prerequisites/Preconditions	9
1.6	Applicability Statement	9
1.7	Versioning and Capability Negotiation.....	9
1.8	Vendor-Extensible Fields	9
1.9	Standards Assignments.....	9
2	Messages	10
2.1	Transport	10
2.2	Common Data Types	10
3	Protocol Details	11
3.1	IIS Application Host Administration Server Details.....	11
3.1.1	Abstract Data Model	11
3.1.2	Timers	11
3.1.3	Initialization	11
3.1.4	Message Processing Events and Sequencing Rules	11
3.1.4.1	IAppHostAdminManager	11
3.1.4.1.1	GetAdminSection (Opnum 3)	12
3.1.4.1.2	GetMetadata (Opnum 4)	13
3.1.4.1.3	SetMetadata (Opnum 5)	15
3.1.4.1.4	ConfigManager (Opnum 6)	16
3.1.4.2	IAppHostChangeHandler.....	17
3.1.4.2.1	OnSectionChanges (Opnum 3).....	17
3.1.4.3	IAppHostChildElementCollection	18
3.1.4.3.1	Count (Opnum 3).....	18
3.1.4.3.2	Item (Opnum 4)	18
3.1.4.4	IAppHostCollectionSchema	19
3.1.4.4.1	AddElementNames (Opnum 3).....	19
3.1.4.4.2	GetAddElementSchema (Opnum 4)	20
3.1.4.4.3	RemoveElementSchema (Opnum 5)	20
3.1.4.4.4	ClearElementSchema (Opnum 6)	21
3.1.4.4.5	IsMergeAppend (Opnum 7)	21
3.1.4.4.6	GetMetadata (Opnum 8)	21
3.1.4.4.7	DoesAllowDuplicates (Opnum 9)	22
3.1.4.5	IAppHostConfigException.....	22
3.1.4.5.1	LineNumber (Opnum 3)	23
3.1.4.5.2	FileName (Opnum 4).....	23
3.1.4.5.3	ConfigPath (Opnum 5).....	23
3.1.4.5.4	ErrorLine (Opnum 6)	24
3.1.4.5.5	PreErrorLine (Opnum 7).....	24
3.1.4.5.6	PostErrorLine (Opnum 8)	24
3.1.4.5.7	ErrorString (Opnum 9)	25
3.1.4.6	IAppHostConfigFile	25
3.1.4.6.1	ConfigPath (Opnum 3).....	26
3.1.4.6.2	FilePath (Opnum 4).....	26

3.1.4.6.3	Locations (Opnum 5).....	26
3.1.4.6.4	GetAdminSection (Opnum 6)	27
3.1.4.6.5	GetMetadata (Opnum 7)	27
3.1.4.6.6	SetMetadata (Opnum 8)	28
3.1.4.6.7	ClearInvalidSections (Opnum 9).....	28
3.1.4.6.8	RootSectionGroup (Opnum 10)	28
3.1.4.7	IAppHostConfigLocation	29
3.1.4.7.1	Path (Opnum 3).....	29
3.1.4.7.2	Count (Opnum 4).....	30
3.1.4.7.3	Item (Opnum 5)	30
3.1.4.7.4	AddConfigSection (Opnum 6).....	31
3.1.4.7.5	DeleteConfigSection (Opnum 7)	31
3.1.4.8	IAppHostConfigLocationCollection	31
3.1.4.8.1	Count (Opnum 3).....	32
3.1.4.8.2	Item (Opnum 4)	32
3.1.4.8.3	AddLocation (Opnum 5)	33
3.1.4.8.4	DeleteLocation (Opnum 6)	33
3.1.4.9	IAppHostConfigManager	33
3.1.4.9.1	GetConfigFile (Opnum 3)	34
3.1.4.9.2	GetUniqueConfigPath (Opnum 4).....	34
3.1.4.10	IAppHostConstantValue.....	35
3.1.4.10.1	Name (Opnum 3).....	35
3.1.4.10.2	Value (Opnum 4)	35
3.1.4.11	IAppHostConstantValueCollection	36
3.1.4.11.1	Count (Opnum 3).....	36
3.1.4.11.2	Item (Opnum 4)	36
3.1.4.12	IAppHostElement	37
3.1.4.12.1	Name (Opnum 3).....	38
3.1.4.12.2	Collection (Opnum 4)	38
3.1.4.12.3	Properties (Opnum 5).....	38
3.1.4.12.4	ChildElements (Opnum 6)	39
3.1.4.12.5	GetMetadata (Opnum 7)	39
3.1.4.12.6	SetMetadata (Opnum 8)	41
3.1.4.12.7	Schema (Opnum 9).....	42
3.1.4.12.8	GetElementByName (Opnum 10)	43
3.1.4.12.9	GetPropertyByName (Opnum 11)	43
3.1.4.12.10	Clear (Opnum 12).....	43
3.1.4.12.11	Methods (Opnum 13).....	44
3.1.4.13	IAppHostElementCollection	44
3.1.4.13.1	Count (Opnum 3).....	45
3.1.4.13.2	Item (Opnum 4)	45
3.1.4.13.3	AddElement (Opnum 5)	45
3.1.4.13.4	DeleteElement (Opnum 6).....	46
3.1.4.13.5	Clear (Opnum 7).....	46
3.1.4.13.6	CreateNewElement (Opnum 8).....	46
3.1.4.13.7	Schema (Opnum 9).....	47
3.1.4.14	IAppHostElementSchema	47
3.1.4.14.1	Name (Opnum 3).....	48
3.1.4.14.2	DoesAllowUnschematizedProperties (Opnum 4)	48
3.1.4.14.3	GetMetadata (Opnum 5)	49
3.1.4.14.4	CollectionSchema (Opnum 6)	49
3.1.4.14.5	ChildElementSchemas (Opnum 7)	49
3.1.4.14.6	PropertySchemas (Opnum 8).....	50
3.1.4.14.7	IsCollectionDefault (Opnum 9).....	50
3.1.4.15	IAppHostElementSchemaCollection	50

3.1.4.15.1	Count (Opnum 3).....	51
3.1.4.15.2	Item (Opnum 4)	51
3.1.4.16	IAppHostMappingExtension.....	51
3.1.4.16.1	GetSiteNameFromSiteID (Opnum 3)	52
3.1.4.16.2	GetSiteIDFromSiteName (Opnum 4)	52
3.1.4.16.3	GetSiteElementFromSiteID (Opnum 5).....	53
3.1.4.16.4	MapPath (Opnum 6)	53
3.1.4.17	IAppHostMethod	54
3.1.4.17.1	Name (Opnum 3).....	54
3.1.4.17.2	Schema (Opnum 4).....	55
3.1.4.17.3	CreateInstance (Opnum 5)	55
3.1.4.18	IAppHostMethodCollection	55
3.1.4.18.1	Count (Opnum 3).....	56
3.1.4.18.2	Item (Opnum 4)	56
3.1.4.19	IAppHostMethodInstance.....	57
3.1.4.19.1	Input (Opnum 3)	57
3.1.4.19.2	Output (Opnum 4)	57
3.1.4.19.3	Execute (Opnum 5).....	58
3.1.4.19.4	GetMetaData (Opnum 6).....	58
3.1.4.19.5	SetMetadata (Opnum 7)	58
3.1.4.20	IAppHostMethodSchema.....	59
3.1.4.20.1	Name (Opnum 3).....	59
3.1.4.20.2	InputSchema (Opnum 4)	60
3.1.4.20.3	OutputSchema (Opnum 5)	60
3.1.4.20.4	GetMetadata (Opnum 6)	60
3.1.4.21	IAppHostPathMapper	61
3.1.4.21.1	MapPath (Opnum 3).....	61
3.1.4.22	IAppHostProperty	62
3.1.4.22.1	Name (Opnum 3).....	62
3.1.4.22.2	Value (Get) (Opnum 4).....	63
3.1.4.22.3	Value (Set) (Opnum 5)	63
3.1.4.22.4	Clear (Opnum 6).....	63
3.1.4.22.5	StringValue (Opnum 7).....	64
3.1.4.22.6	Exception (Opnum 8)	64
3.1.4.22.7	GetMetadata (Opnum 9)	64
3.1.4.22.8	SetMetadata (Opnum 10).....	65
3.1.4.22.9	Schema (Opnum 11).....	65
3.1.4.23	IAppHostPropertyCollection.....	66
3.1.4.23.1	Count (Opnum 3).....	66
3.1.4.23.2	Item (Opnum 4)	66
3.1.4.24	IAppHostPropertyException.....	67
3.1.4.24.1	InvalidValue (Opnum 10)	68
3.1.4.24.2	ValidationFailureReason (Opnum 11)	68
3.1.4.24.3	ValidationFailureParameters (Opnum 12)	68
3.1.4.25	IAppHostPropertySchema	69
3.1.4.25.1	Name (Opnum 3).....	70
3.1.4.25.2	Type (Opnum 4)	70
3.1.4.25.3	DefaultValue (Opnum 5)	70
3.1.4.25.4	IsRequired (Opnum 6).....	70
3.1.4.25.5	IsUniqueKey (Opnum 7)	71
3.1.4.25.6	IsCombinedKey (Opnum 8)	71
3.1.4.25.7	IsExpanded (Opnum 9).....	71
3.1.4.25.8	ValidationType (Opnum 10).....	72
3.1.4.25.9	ValidationParameter (Opnum 11)	72
3.1.4.25.10	GetMetaData (Opnum 12)	72

3.1.4.25.11	IsCaseSensitive (Opnum 13).....	73
3.1.4.25.12	PossibleValues (Opnum 14)	73
3.1.4.25.13	DoesAllowInfinite (Opnum 15)	73
3.1.4.25.14	IsEncrypted (Opnum 16).....	74
3.1.4.25.15	TimeSpanFormat (Opnum 17).....	74
3.1.4.26	IAppHostPropertySchemaCollection.....	74
3.1.4.26.1	Count (Opnum 3).....	75
3.1.4.26.2	Item (Opnum 4)	75
3.1.4.27	IAppHostSectionDefinition	76
3.1.4.27.1	Name (Opnum 3).....	76
3.1.4.27.2	Type (Get) (Opnum 4).....	77
3.1.4.27.3	Type (Set) (Opnum 5).....	77
3.1.4.27.4	OverrideModeDefault (Get) (Opnum 6)	77
3.1.4.27.5	OverrideModeDefault (Set) (Opnum 7).....	78
3.1.4.27.6	AllowDefinition (Get) (Opnum 8)	78
3.1.4.27.7	AllowDefinition (Set) (Opnum 9)	78
3.1.4.27.8	AllowLocation (Get) (Opnum 10).....	79
3.1.4.27.9	AllowLocation (Set) (Opnum 11)	79
3.1.4.28	IAppHostSectionDefinitionCollection	79
3.1.4.28.1	Count.....	80
3.1.4.28.2	Item	80
3.1.4.28.3	AddSection	80
3.1.4.28.4	DeleteSection	81
3.1.4.29	IAppHostSectionGroup	81
3.1.4.29.1	Count (Opnum 3).....	82
3.1.4.29.2	Item (Opnum 4)	82
3.1.4.29.3	Sections (Opnum 5)	82
3.1.4.29.4	AddSectionGroup (Opnum 6).....	83
3.1.4.29.5	DeleteSectionGroup (Opnum 7)	83
3.1.4.29.6	Name (Opnum 8).....	83
3.1.4.29.7	Type (Get) (Opnum 9).....	84
3.1.4.29.8	Type (Set) (Opnum 10)	84
3.1.4.30	IAppHostWritableAdminManager.....	84
3.1.4.30.1	CommitChanges (Opnum 7)	85
3.1.4.30.2	CommitPath (Get) (Opnum 8).....	85
3.1.4.30.3	CommitPath (Set) (Opnum 9).....	86
3.1.4.31	IAppHostMethodExtension	86
3.1.4.31.1	ProvideMethod	86
3.1.4.32	IAppHostPropertyExtension.....	87
3.1.4.32.1	ProvideGetProperty	87
3.1.4.33	IAppHostElementExtension	88
3.1.4.33.1	ProvideElement	88
3.1.4.34	IAppHostPathMapper2.....	88
3.1.4.34.1	MapPath	89
3.1.5	Timer Events.....	90
3.1.6	Other Local Events.....	90

4 Protocol Examples91

4.1	Create an AppHostAdminManager Locally.....	91
4.2	Get Metadata: Get the overrideMode of a defaultDocument Section	91
4.3	Set Metadata: Set the overrideMode of the defaultDocument Section.....	91
4.4	Create a New Configuration Section Entry in the configSections Section	91
4.5	Get a Section for Read Access: The defaultDocument section.....	92
4.6	Get a Property: Get the Enabled Property of the defaultDocument Section.....	92
4.7	Get a Section: Get the anonymousAuthentication Section.....	92

4.8	List the Entries of a Collection	92
4.9	Remove an Entry of a Collection	93
4.10	Edit the Configuration of APPHOST in a Location Tag	93
4.11	Read Schema Information: Determine If IsMergeAppend Is Set in the defaultDocuments Section	93
4.12	Get a Section for Write: Get the defaultDocument Section and Toggle the Enabled Attribute	94
4.13	Write into a Collection: Clear the Contents of the defaultDocument Section for Site1	94
4.14	Write into a Collection: Add an Entry for the defaultDocument Section for Site1 as a Location Tag.....	94
5	Security	95
5.1	Security Considerations for Implementers	95
5.2	Index of Security Parameters	95
6	Appendix A: Full IDL	96
7	Appendix B: Windows Behavior	133
8	Index.....	134

1 Introduction

This document specifies the Internet Information Services (IIS) Application Host COM Protocol. This protocol is a client-to-server protocol that enables remote read/write access to server data. The server data MAY be used to define administration, configuration, and operational parameters to an application server service, which MAY be a Web server.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

Dynamic Endpoint Interface
Opnum
Remote Procedure Call (RPC)
Universally Unique Identifier (UUID)

The following terms are specific to this document:

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

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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, <http://www.opengroup.org/public/pubs/catalog/c706.htm>

[MS-DCOM] Microsoft Corporation, "[Distributed Component Object Model \(DCOM\) Remote Protocol Specification](#)", March 2007.

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](#)", January 2007.

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)", March 2007.

[MS-OAUT] Microsoft Corporation, "[OLE Automation Protocol Specification](#)", March 2007.

[MS-RPCE] Microsoft Corporation, "[Remote Procedure Call Protocol Extensions](#)", January 2007.

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

1.2.2 Informative References

[MSDN-IIS7AH] Microsoft Corporation, "IIS 7.0: IIS Application Host Administration API Reference", <http://msdn2.microsoft.com/en-us/library/aa965120.aspx>

1.3 Protocol Overview (Synopsis)

This protocol is intended to provide read/write access to administrative configuration data that is located on a remote server computer. The administrative configuration data is implementation-specific for each server.

1.4 Relationship to Other Protocols

This protocol depends on the [DCOM Remote Protocol](#), as specified in [MS-DCOM].

1.5 Prerequisites/Preconditions

This protocol requires that the [DCOM Remote Protocol](#) MUST be implemented on both the client and server computers.

This protocol specification assumes that any security or authentication associations between the client and server MAY be performed by the DCOM layer.

1.6 Applicability Statement

This protocol is applicable to the remote reading and writing of hierarchically organized server administration data.

1.7 Versioning and Capability Negotiation

This protocol does not provide a mechanism for protocol versioning or capability negotiation.

1.8 Vendor-Extensible Fields

This protocol does not include any vendor-extensible fields.

This protocol uses HRESULTs that are vendor-extensible, as specified in [\[MS-ERREF\]](#) section 2.1. Vendors can choose their own values for this field as long as the C bit (0x20000000) is set, indicating it is a customer code.

This protocol uses Win32 error codes. These values are taken from the numbering space of the Windows error codes, as specified in [\[MS-ERREF\]](#) section 2.2. Vendors SHOULD use the values that are shown in section 2.2, retaining their default meaning. Choosing any other meaning for these values risks a future collision.

1.9 Standards Assignments

None.

2 Messages

The following sections specify how Internet Information Services (IIS) Application Host COM Protocol messages are transported and the common data types.

2.1 Transport

This protocol uses **RPC dynamic endpoints** as specified in [\[C706-Ch6RPCCallModel\]](#).

2.2 Common Data Types

This protocol does not define any data types.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other states are required on the client side of this protocol. Calls that are made by any client application are passed directly to the transport, and the results that are returned by the transport are passed directly back to the calling application.

The following sections specify details of the Internet Information Services (IIS) Application Host COM Protocol, including abstract data models, interface method syntax, and message processing rules.

3.1 IIS Application Host Administration Server Details

3.1.1 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.

3.1.2 Timers

This protocol implementation does not require the explicit use of any timers.

3.1.3 Initialization

This protocol uses DCOM initialization.

3.1.4 Message Processing Events and Sequencing Rules

In the **interfaces** that are described in the following sections, no exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [\[MS-RPCE\]](#).

3.1.4.1 IAppHostAdminManager

IAppHostAdminManager represents a read-only interface to an administration system that is implemented on the server.

The administration system consists primarily of a set of administration objects of varying complexity, which are accessed through the **IAppHostElement** interface (section [3.1.4.12](#)) and exist at one or more paths that are exposed by the server. The administration system allows access to individual **IAppHostElement** objects that are available at specific paths and also provides access to merged **IAppHostElement** objects that consist of the merged contents of individual **IAppHostElement** objects.

Secondarily, the administration system provides access to, and the setting of, specific system behaviors that are available for administration objects, which are represented by the term "metadata." Metadata allows a caller to modify and inspect the behavior of the administration system.

IAppHostAdminManager is used by clients to access (for read-only purposes) the contents of the administration system without altering the contents of the system. A tool that seeks to display the administration objects that are contained in the system is an example of a consumer of the **IAppHostAdminManager** interface.

The **IAppHostAdminManager** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 2
GetAdminSection	Opnum: 3
GetMetadata	Opnum: 4
SetMetadata	Opnum: 5
ConfigManager	Opnum: 6

3.1.4.1.1 GetAdminSection (Opnum 3)

The **GetAdminSection** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElement](#) that contains a merging of one or more individual **IAppHostElement** objects, which are set at one or more places in the path hierarchy that is specified in the method call.

```
HRESULT GetAdminSection(  
    [in, string] BSTR bstrSectionName,  
    [in, string] BSTR bstrPath,  
    [out, retval] IAppHostElement** ppAdminSection  
);
```

bstrSectionName: The name of the administration section to fetch. The server contains named **IAppHostElement** objects that are specified at one or more paths. The name of the **IAppHostElement** is called the "section name" of the element. This parameter represents the required section name. The section name syntax is specific to the implementation of the server and "myAuthenticationSection".

bstrPath: The path hierarchy for which to find and merge **IAppHostElement** objects. The server uses this path as an indication of where to look for **IAppHostElement** objects. The server finds all the **IAppHostElement** objects that exist anywhere in this path. How the server parses this path and potentially maps it to the location of **IAppHostElement** objects is specific to each implementation.

ppAdminSection: Returns a merged **IAppHostElement** object that contains the merged contents of one or more **IAppHostElement** objects that are located at points in the hierarchy that is specified in *bstrPath*.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.1.2 GetMetadata (Opnum 4)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. The purpose of the method is to retrieve settings and other behavior-modifying attributes of the administration system. The administration system supports modifications through these settings and attributes. The metadata returns the details of these modifications to the caller. The returned metadata is referenced by a string name, and is thus extensible without changing the main IDL file. As administration systems implementers add more options to their systems, they can expose these options as named metadata through this method.

This method is used to get a metadata property.

```
HRESULT GetMetadata(  
    [in, string] BSTR bstrMetadataType,  
    [out, retval] VARIANT* pValue  
);
```

bstrMetadataType: The name of the metadata property to fetch.

Value	Meaning
"pathMapper"	Returns the IAppHostPathMapper object, which is a user-provided object, that controls how the administration system implementation maps a specified path hierarchy to actual resources. IAppHostPathMapper
"pathMapper2"	Returns the IAppHostPathMapper2 object, which is a user-provided object, that controls how the administration system implementation maps a specified path hierarchy to actual resources. IAppHostPathMapper2
"changeHandler"	Returns the IAppHostChangeHandler object, which is a user-provided object, that receives notification when the administration system detects that a part of the path hierarchy has changed. IAppHostChangeHandler
"ignoreInvalidAttributes"	Returns the flag that controls whether the administration system should ignore certain types of errors when evaluating the IAppHostProperty part of an IAppHostElement . If the flag is true, the administration system ignores these errors. If the flag is false, the administration system treats these IAppHostProperty errors as fatal and invalidates the container IAppHostElement . bool
"ignoreInvalidRanges"	Returns the flag that controls whether the administration system should ignore certain types of errors when evaluating the IAppHostProperty part of an IAppHostElement . If the flag is true, the administration system ignores these errors. If the flag is false, the administration system treats these IAppHostProperty errors as fatal and invalidates the container IAppHostElement . bool
"ignoreInvalidDecryption"	Returns the flag that controls whether the administration system should ignore decryption errors when evaluating the IAppHostProperty part of an IAppHostElement . If the flag is

Value	Meaning
	true, the administration system ignores these errors. If the flag is false, the administration system treats these IAppHostProperty errors as fatal and invalidates the container IAppHostElement . bool
"expandEnvironmentStrings"	Returns the flag that controls whether the administration system should expand environment variables when parsing the configuration system. If the flag is true, the administration system expands the variables. If the flag is false, the administration system does not expand the variables. bool
"disableExtensions"	Returns a flag that determines whether the administration system supports custom server-side extensions that can extend how the system evaluates IAppHostElement objects. If true, the system will not support custom server-side extensions. If false, the system will support custom server-side extensions. bool
"availableSections"	Returns a comma-separated series of strings that contains all the names of the available administration section names that are supported by the administration system. Section names represent the names of IAppHostElement objects. If the administration system supports an IAppHostElement that is named "X", then X must appear in the comma-delimited list that is returned. string
"mappingExtension"	Returns an object that is used to directly access the hierarchy mapping system of the administration system, and is accessed through the IAppHostMappingExtension interface. IAppHostMappingExtension
"hideExceptionPhysicalPath"	Returns the flag that controls whether the administration system should give information about the physical path of the configuration file in exception messages. If the flag is true, the administration system hides the physical path. If the flag is false, the administration system returns the physical path. bool
"lockMetadata"	Returns the flag that controls whether metadata can be modified with the SetMetadata call. If the flag is true, then further calls to SetMetadata will fail with a locking violation. If the flag is false, SetMetadata calls are allowed. bool

pValue: Returns the value of the specified metadata property, and the type depends on the property fetched. Upon success, the property is returned in a [VARIANT](#), and the type of the value depends on the type of metadata that is requested (specified in bstrMetadataType).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.1.3 SetMetadata (Opnum 5)

The **SetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the administration system implementation changes its behavior as specified by the provided metadata. This method is the opposite of the [GetMetadata](#) method, which retrieves a specified behavior property of the administration system. **SetMetadata** sets a specified behavior property of the administration system.

This method is used to set a metadata property.

```
HRESULT SetMetadata(  
    [in, string] BSTR bstrMetadataType,  
    [in] VARIANT value  
);
```

bstrMetadataType: The name of the metadata property to set.

Value	Meaning
"pathMapper"	Sets the IAppHostPathMapper object, which is a user-provided object that controls how the implementation of the administration system maps a specific path hierarchy to actual resources. IAppHostPathMapper
"pathMapper2"	Sets the IAppHostPathMapper2 object, which is a user-provided object that controls how the implementation of the administration system maps a specific path hierarchy to actual resources. IAppHostPathMapper2
"changeHandler"	Sets the IAppHostChangeHandler object, which is a user-provided object that receives notification when the administration system detects that a part of the path hierarchy has changed. IAppHostChangeHandler
"ignoreInvalidAttributes"	Sets the flag that controls whether the administration system should ignore certain types of errors when evaluating the IAppHostProperty part of an IAppHostElement . If the flag is true, the administration system ignores these errors. If the flag is false, the administration system treats these IAppHostProperty errors as fatal and invalidates the IAppHostElement container. bool
"ignoreInvalidRanges"	Sets the flag that controls whether the administration system should ignore certain range validation errors when it evaluates the IAppHostProperty part of an IAppHostElement . If the flag is true, the administration system ignores these errors. If the flag is false, the administration system treats these IAppHostProperty errors as fatal and invalidates the IAppHostElement container. bool
"ignoreInvalidDecryption"	Sets the flag that controls whether the administration system should ignore decryption errors when evaluating the IAppHostProperty part of an IAppHostElement . If the flag is true, the administration system ignores these errors. If the flag is false, the administration system treats these IAppHostProperty errors as fatal and

Value	Meaning
	invalidates the IAppHostElement container. bool
"expandEnvironmentStrings"	Sets the flag that controls whether the administration system should expand environment variables when parsing the configuration system. If the flag is true, the administration system expands the variables. If the flag is false, the administration system does not expand the variables. bool
"disableExtensions"	Sets a flag that determines whether the administration system supports custom server-side extensions that can extend how the system evaluates IAppHostElement objects. If true, the system will not support custom server-side extensions. If false, the system will support custom server-side extensions. bool
"hideExceptionPhysicalPath"	Sets the flag that controls whether the administration system should give information about the physical path of the configuration file in exception messages. If the flag is true, the administration system hides the physical path. If the flag is false, the administration system returns the physical path. bool
"lockMetadata"	Sets the flag that controls whether metadata can be modified with the SetMetadata call. If the flag is true, then further calls to SetMetadata will fail with a locking violation. If the flag is false, SetMetadata calls are allowed. bool

value: The value of the metadata property to set.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.1.4 ConfigManager (Opnum 6)

The **ConfigManager** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostConfigManager](#) that provides direct access to the supported path hierarchy of the administration system and access to individual [IAppHostElement](#) objects that are contained within.

```
[propget] HRESULT ConfigManager(
    [out, retval] IAppHostConfigManager** ppConfigManager
);
```

ppConfigManager: If the server successfully processes the message that is received from the client, *ppConfigManager* contains a pointer to an **IAppHostConfigManager** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. If successful, *ppConfigManager must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.2 IAppHostChangeHandler

The **IAppHostChangeHandler** describes an interface that clients can implement and that is called when the administration system has detected a change in a part of its path hierarchy.

To receive incoming remote calls for this interface, the client must implement this interface (09829352-87c2-418d-8d79-4133969a489d). It must then specify an object that implements this interface to the [IAppHostAdminManager::SetMetadata\(\)](#) method by using a bstrMetadataName of "changeHandler".

The server then calls this object when the administration system detects a change. The administration system is free to determine the supported time period during which the changes will be detected and conveyed through this interface. The time period may be the lifetime of the administration system or a shorter time period.

The **IAppHostChangeHandler** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
OnSectionChanges	Opnum: 3

3.1.4.2.1 OnSectionChanges (Opnum 3)

The **OnSectionChanges** method is called by the server by using an RPC_REQUEST packet. This method is called when a change in the path of the administration system hierarchy is detected. The callee (the client-implemented object) may react to this notification as it determines. It can return any error and the server must ignore it.

```
HRESULT OnSectionChanges(  
    [in, string] BSTR bstrSectionName,  
    [in, string] BSTR bstrConfigPath  
);
```

bstrSectionName: The name of the [IAppHostElement](#) on the server that changed. A server is free to not implement this parameter and always passes NULL.

bstrConfigPath: The path in the hierarchy where the change was detected by the administration system.

Return Values: The return value must be ignored by the server.

3.1.4.3 IAppHostChildElementCollection

The **IAppHostChildElementCollection** interface provides methods that allow access to any fixed child [IAppHostElement](#) objects that are contained by the specific **IAppHostElement** object that provides this interface through [IAppHostElement::ChildElements\(\)](#).

An **IAppHostElement** may contain any number of fixed, uniquely-named child **IAppHostElement** objects. This **IAppHostChildElementCollection** provides access to these child elements.

The **IAppHostChildElementCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4

3.1.4.3.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the number of child [IAppHostElement](#) objects in the specific [IAppHostChildElementCollection](#) object.

```
[propget] HRESULT Count(  
    [out, retval] DWORD* pcCount  
);
```

pcCount: Receives a count of the number of child elements that are contained in the specified **IAppHostChildElementCollection** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.3.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the requested child [IAppHostElement](#) object that is contained in the specific [IAppHostChildElementCollection](#) object.

```
[propget, id(DISPID_VALUE)] HRESULT Item(  
    [in] VARIANT cIndex,  
    [out, retval] IAppHostElement** ppElement
```

);

cIndex: A client-specified [VARIANT](#) that specifies the specific child **IAppHostElement** that should be returned in the method. If the **VARIANT** is of type integer, the value represents a zero-based index that specifies the index of the child element to return, where 0 indicates the first child, 1 the second, and so on. The **VARIANT** can also be of the [BSTR](#) type, and if so, represents a string name that specifies the exact name of the child element to return.

ppElement: Receives a pointer to the child **IAppHostElement** that matches the criteria that is set by the *cIndex* parameter.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppElement must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4 IAppHostCollectionSchema

The **IAppHostCollectionSchema** provides methods that describe the schema and constraints that apply to a specific [IAppHostElementCollection](#) from which this object was retrieved.

The **IAppHostCollectionSchema** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
AddElementNames	Opnum: 3
GetAddElementSchema	Opnum: 4
RemoveElementSchema	Opnum: 5
ClearElementSchema	Opnum: 6
IsMergeAppend	Opnum: 7
GetMetadata	Opnum: 8
DoesAllowDuplicates	Opnum: 9

3.1.4.4.1 AddElementNames (Opnum 3)

The **AddElementNames** method is received by the server in an RPC_REQUEST packet. In response, the server returns a comma-delimited list of names that are supported by the administration system as names to [IAppHostElement](#) objects that are collection items of an [IAppHostElementCollection](#) object. An administration system typically supports only one name

for the **IAppHostElement** objects that are contained in the collection. However, it could support more names in certain conditions; in which case, all the names are returned by using this method.

```
[propget()] HRESULT AddElementNames(  
    [out, string, retval] BSTR* pbstrElementName  
);
```

pbstrElementName: Returns the comma-delimited string that contains the names of all supported names of **IAppHostElement** objects that are contained in the **IAppHostElementCollection** from which the specified [IAppHostCollectionSchema](#) was retrieved.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrElementName must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4.2 GetAddElementSchema (Opnum 4)

The **GetAddElementSchema** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElementSchema](#) that represents the schema and constraints of the [IAppHostElement](#). The **IAppHostElement** can be a collection item of the specified [IAppHostElementCollection](#) from which the specified [IAppHostCollectionSchema](#) was retrieved and whose name matches the specified name in the method call.

```
HRESULT GetAddElementSchema(  
    [in, string] BSTR bstrElementName,  
    [out, retval] IAppHostElementSchema** ppSchema  
);
```

bstrElementName: The name of the **IAppHostElement** that is contained in the specified collection. It is one of the names that is returned in the [AddElementNames](#) method.

ppSchema: Returns the **IAppHostElementSchema** schema object that is associated with the specified element name.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSchema must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4.3 RemoveElementSchema (Opnum 5)

The **RemoveElementSchema** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostElementSchema](#) that represents the schema and constraints of an optionally supported "directive IAppHostElement". This directive element may be used by the administration system to control the behavior of the specific [IAppHostElementCollection](#) from which the specified [IAppHostCollectionSchema](#) was retrieved.

```
[propget] HRESULT RemoveElementSchema(  
    [out, retval] IAppHostElementSchema** ppSchema  
);
```

ppSchema: Returns an **IAppHostElementSchema** object that is associated with the optionally supported directive [IAppHostElement](#).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4.4 ClearElementSchema (Opnum 6)

The **ClearElementSchema** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostElementSchema](#) that represents the schema and constraints of an optionally supported "directive IAppHostElement". This directive element may be used by the administration system to control behavior of the specific [IAppHostElementCollection](#) from which the specified [IAppHostCollectionSchema](#) was retrieved.

```
[propget] HRESULT ClearElementSchema(  
    [out, retval] IAppHostElementSchema** ppSchema  
);
```

ppSchema: Returns an **IAppHostElementSchema** object that is associated with the optionally supported directive [IAppHostElement](#).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4.5 IsMergeAppend (Opnum 7)

The **IsMergeAppend** method is received by the server in an RPC_REQUEST packet. In response, the server returns a Boolean that represents whether the [IAppHostElementCollection](#), from which the specified [IAppHostCollectionSchema](#) was retrieved, will prepend collection [IAppHostElement](#) objects from lower (deeper) in the hierarchy of the administration system with **IAppHostElement** objects from higher (shallower) in the hierarchy of the administration system.

If the value is false, lower **IAppHostElement** objects are prepended; otherwise, they are appended.

```
[propget] HRESULT IsMergeAppend(  
    [out, retval] VARIANT_BOOL* pfIsMergeAppend  
);
```

pfIsMergeAppend: The Boolean value that represents the append or prepend behavior.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4.6 GetMetadata (Opnum 8)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the metadata that is applicable to this [IAppHostCollectionSchema](#).

```
[propget] HRESULT GetMetadata(  

```

```

[in, string] BSTR bstrMetadataName,
[out] VARIANT* pValue
);

```

bstrMetadataName: The name of the metadata that is retrieved.

pValue: Contains the value of the metadata being retrieved.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.4.7 DoesAllowDuplicates (Opnum 9)

The **DoesAllowDuplicates** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether (through a Boolean) the [IAppHostElementCollection](#) from which the specified [IAppHostCollectionSchema](#) was retrieved allows duplicate [IAppHostElement](#) objects to exist in the collection. The concept of "duplicate" is implementation-specific.

```

HRESULT DoesAllowDuplicates(
[out, retval] VARIANT_BOOL* pfAllowDuplicates
);

```

pfAllowDuplicates: The Boolean value that represents whether duplicates are supported.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5 IAppHostConfigException

Methods that receive an error when they access the administration system can retrieve more specific information about why the error occurred by using the **IAppHostConfigException** interface.

The **IAppHostConfigException** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
LineNumber	Opnum: 3
FileName	Opnum: 4

Method	Description
ConfigPath	Opnum: 5
ErrorLine	Opnum: 6
PreErrorLine	Opnum: 7
PostErrorLine	Opnum: 8
ErrorString	Opnum: 9

3.1.4.5.1 LineNumber (Opnum 3)

The **LineNumber** (opnum 3) method is received by the server in an RPC_REQUEST packet. In response, the server returns a specific line number that may provide more detail regarding the location of the error in the hierarchy of the administration system.

```
[propget] HRESULT LineNumber(
    [out, retval] unsigned long* pcLineNumber
);
```

pcLineNumber: Contains the line number.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5.2 FileName (Opnum 4)

The **FileName** method is received by the server in an RPC_REQUEST packet. In response, the server returns a file name that may provide more detail regarding the location of the error in the hierarchy of the administration system.

```
[propget] HRESULT FileName(
    [out, retval] BSTR* pbstrFileName
);
```

pbstrFileName: Contains the file name.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrFileName must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5.3 ConfigPath (Opnum 5)

The **ConfigPath** method is received by the server in an RPC_REQUEST packet. In response, the server returns a path in the supported hierarchy of the administration system that contains the error.

```
[propget] HRESULT ConfigPath(
    [out, retval] BSTR* pbstrConfigPath
);
```

```
);
```

pbstrConfigPath: Contains the hierarchy path.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrConfigPath must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5.4 ErrorLine (Opnum 6)

The **ErrorLine** method is received by the server in an RPC_REQUEST packet. In response, the server returns a textual representation of the specific data in the administration system that is causing the error.

```
[propget] HRESULT ErrorLine(  
    [out, retval] BSTR* pbstrErrorLine  
);
```

pbstrErrorLine: Contains the error data.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrErrorLine must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5.5 PreErrorLine (Opnum 7)

The **PreErrorLine** method is received by the server in an RPC_REQUEST packet. In response, the server returns a textual representation of the data that precedes the specific data in the administration system that is causing the error.

```
[propget] HRESULT PreErrorLine(  
    [out, retval] BSTR* pbstrPreErrorLine  
);
```

pbstrPreErrorLine: Contains the error data.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrPreErrorLine must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5.6 PostErrorLine (Opnum 8)

The **PostErrorLine** method is received by the server in an RPC_REQUEST packet. In response, the server returns a textual representation of the data that follows the specific data in the administration system that is causing the error.

```
[propget] HRESULT PostErrorLine(  
    [out, retval] BSTR* pbstrPostErrorLine  
);
```


pbstrPostErrorLine: Contains the error data.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrPostErrorLine must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.5.7 ErrorString (Opnum 9)

The **ErrorString** method is received by the server in an RPC_REQUEST packet. In response, the server returns a description of the error that occurred in the administration system.

```
[propget] HRESULT ErrorString(  
    [out, retval] BSTR* pbstrErrorString  
);
```

pbstrErrorString: Contains the error description.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrErrorString must not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6 IAppHostConfigFile

The **IAppHostConfigFile** provides methods for direct access to a container of [IAppHostElement](#) objects in the administration system. The administration system implements a path hierarchy that allows for **IAppHostElement** objects to be defined at multiple places. This path hierarchy is conceptually split into two levels. One is the level that is represented by **IAppHostConfigFile**.

Each **IAppHostConfigFile** maps to a specific path (for example, "MACHINE/WEBROOT"). Within each **IAppHostConfigFile**, individual **IAppHostElement** objects exist, which are the base objects of the administration system. Optionally, the **IAppHostConfigFile** may also support subpaths within it.

The **IAppHostConfigFile** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
ConfigPath	Opnum: 3
FilePath	Opnum: 4
Locations	Opnum: 5
GetAdminSection	Opnum: 6

Method	Description
GetMetadata	Opnum: 7
SetMetadata	Opnum: 8
ClearInvalidSections	Opnum: 9
RootSectionGroup	Opnum: 10

3.1.4.6.1 ConfigPath (Opnum 3)

The **ConfigPath** method is received by the server in an RPC_REQUEST packet. In response, the server returns the hierarchy path that is represented by the specific [IAppHostConfigFile](#).

```
[propget] HRESULT ConfigPath(
    [out, string, retval] BSTR* pbstrConfigPath
);
```

pbstrConfigPath: Contains the hierarchy path.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrConfigPath is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.2 FilePath (Opnum 4)

The **FilePath** method is received by the server in an RPC_REQUEST packet. In response, the server returns the server operating system file path that corresponds to the specific [IAppHostConfigFile](#), if that path applies to the administration system implementation.

```
[propget] HRESULT FilePath(
    [out, string, retval] BSTR* pbstrFilePath
);
```

pbstrFilePath: Contains the file path of the **IAppHostConfigFile**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrFilePath is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.3 Locations (Opnum 5)

The **Locations** method is received by the server in an RPC_REQUEST packet. In response, the server returns any hierarchy subpaths that may exist in the specified [IAppHostConfigFile](#). These subpaths are returned in the form of a collection object that implements the [IAppHostConfigLocationCollection](#).

```
[propget] HRESULT Locations(
    [out, retval] IAppHostConfigLocationCollection** ppLocations
);
```

ppLocations: Contains a collection object that contains all the subpaths that are available in the specified **IAppHostConfigFile**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppLocations is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.4 GetAdminSection (Opnum 6)

The **GetAdminSection** method is received by the server in an RPC_REQUEST packet. In response, the server returns a single **IAppHostElement** at the specified hierarchy path in the specific **IAppHostConfigFile**. The section is not a merge and is instead a single **IAppHostElement**.

If no **IAppHostElement** object exists at the specified path, the implementation MAY return either an error or an empty **IAppHostElement** object (the default).

This method is used to get a specific administration section.

```
HRESULT GetAdminSection(  
    [in, string] BSTR bstrSectionName,  
    [in, string] BSTR bstrPath,  
    [out, retval] IAppHostElement** ppAdminSection  
);
```

bstrSectionName: The name of the **IAppHostElement** object to retrieve.

bstrPath: The hierarchy path of the **IAppHostElement** object to retrieve. It must be a path that is supported in the specific **IAppHostConfigFile** for the method to succeed and return an **IAppHostElement** object.

ppAdminSection: Contains an **IAppHostElement** from the specific **IAppHostConfigFile**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppAdminSection is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.5 GetMetadata (Opnum 7)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns metadata that applies to the specific **IAppHostConfigFile**.

No metadata is currently associated with this interface.

```
HRESULT GetMetaData(  
    [in, string] BSTR bstrMetadataType,  
    [out, retval] VARIANT* pValue  
);
```

bstrMetadataType: The name of the metadata property to fetch.

pValue: Returns the value of the specified metadata property. The type depends on the property that is fetched.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.6 SetMetadata (Opnum 8)

The **SetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server sets the metadata for the specific [IAppHostConfigFile](#).

This method is used to set a metadata property.

No metadata is currently associated with this interface.

```
HRESULT SetMetadata(  
    [in, string] BSTR bstrMetadataType,  
    [in] VARIANT value  
);
```

bstrMetadataType: The name of the metadata property to set.

value: The value of the metadata property to set.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.7 ClearInvalidSections (Opnum 9)

The **ClearInvalidSections** method is received by the server in an RPC_REQUEST packet. In response, the server can clear any invalid [IAppHostElement](#) objects that exist in the specific [IAppHostConfigFile](#), if possible.

```
HRESULT ClearInvalidSections();
```

This method has no parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.6.8 RootSectionGroup (Opnum 10)

The **RootSectionGroup** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostSectionGroup](#) object, which represents a declaration of [IAppHostElement](#) objects that apply to the specified [IAppHostConfigFile](#) and that potentially apply to other [IAppHostConfigFile](#) at deeper hierarchy paths than the current file.

A declaration means that an **IAppHostElement** of the name in the declaration MAY exist in the specified **IAppHostConfigFile** or potentially in deeper paths. A declaration is NOT a definition/instance. A declaration only controls whether an actual **IAppHostElement** instance is supported.

This function returns the section group object in the configSections section group that defines the root section group for the current section.

```
[propget] HRESULT RootSectionGroup(
    [out, retval] IAppHostSectionGroup** ppSectionGroups
);
```

ppSectionGroups: Contains an **IAppHostSectionGroup** that contains declarations.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSectionGroups is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.7 IAppHostConfigLocation

The **IAppHostConfigLocation** interface provides methods that access the [IAppHostElement](#) objects that are contained in a specific hierarchy subpath in a specified [IAppHostConfigFile](#). **IAppHostConfigFile** maps to a specific hierarchy path in the administration system, as specified in section 3.1.4.6.

Each **IAppHostConfigFile** can optionally contain subpaths within it. Each subpath is represented by an **IAppHostConfigLocation** object. The object contains a collection of **IAppHostElement** objects, with the guarantee that each **IAppHostConfigLocation** contains at most one **IAppHostElement** object of the same name (in other words, the **IAppHostElement** object name is a key into the collection).

The **IAppHostConfigLocation** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Path	Opnum: 3
Count	Opnum: 4
Item	Opnum: 5
AddConfigSection	Opnum: 6
DeleteConfigSection	Opnum: 7

3.1.4.7.1 Path (Opnum 3)

The **Path** method is received by the server in an RPC_REQUEST packet. In response, the server returns the subpath for the specific [IAppHostConfigLocation](#).

```
[propget] HRESULT Path(
```

```
[out, string, retval] BSTR* pbstrLocationPath
);
```

pbstrLocationPath: Contains the path for the specific **IAppHostConfigLocation**. If the path is "" (an empty string), it represents the same path as the containing [IAppHostConfigFile](#).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrLocationPath is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.7.2 Count (Opnum 4)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the number of [IAppHostElement](#) objects that are defined or exist at the specified [IAppHostConfigLocation](#).

```
[propget()] HRESULT Count(
    [out, retval] DWORD* pcCount
);
```

pcCount: The integer count of the number of **IAppHostElement** objects at the specified **IAppHostConfigLocation**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.7.3 Item (Opnum 5)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostElement](#) object that corresponds to the specific index parameter.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT cIndex,
    [out, retval] IAppHostElement** ppSection
);
```

cIndex: A [VARIANT](#), which is used to specify the **IAppHostElement** object to retrieve. If the **VARIANT** is of type integer, the index is a zero-based index to the collection of **IAppHostElement** objects, where 0 indicates the first **IAppHostElement** object, 1 the second, and so on. If the **VARIANT** is of type string, the index is the name of the **IAppHostElement** object that is retrieved.

ppSection: Contains an **IAppHostElement** object, if found.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSection is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.7.4 AddConfigSection (Opnum 6)

The **AddConfigSection** method is received by the server in an RPC_REQUEST packet. In response, the server attempts to create a new empty [IAppHostElement](#) and add it to the specified [IAppHostConfigLocation](#). The server MAY choose to create the **IAppHostElement** object in memory only and not persist it to permanent storage, such as a disk file, until later.

```
HRESULT AddConfigSection(  
    [in] BSTR bstrSectionName,  
    [out, retval] IAppHostElement** ppAdminElement  
);
```

bstrSectionName: The name of the new **IAppHostElement** section to add.

ppAdminElement: Contains a newly created **IAppHostElement**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppAdminElement is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.7.5 DeleteConfigSection (Opnum 7)

The **DeleteConfigSection** method is received by the server in an RPC_REQUEST packet. In response, the server attempts to delete the [IAppHostElement](#) of the specified index.

```
HRESULT DeleteConfigSection(  
    [in] VARIANT cIndex  
);
```

cIndex: A [VARIANT](#), which is used to specify the **IAppHostElement** object to delete. If the **VARIANT** is of type integer, the index is a zero-based index to the collection of **IAppHostElement** objects, where 0 indicates the first **IAppHostElement** object, 1 the second, and so on. If the **VARIANT** is of type string, the index is the name of the **IAppHostElement** object being retrieved.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.8 IAppHostConfigLocationCollection

The **IAppHostConfigLocationCollection** interface provides methods that access the collection of subpaths that are available in the specified [IAppHostConfigFile](#).

The **IAppHostConfigLocationCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0

Method	Description
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4
AddLocation	Opnum: 5
DeleteLocation	Opnum: 6

3.1.4.8.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of subpaths (each represented by the [IAppHostConfigLocation](#)) in the specified collection.

```
[propget] HRESULT Count(
    [out, retval] DWORD* pcCount
);
```

pcCount: Contains a count of the number of subpaths in the specified collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.8.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the specific [IAppHostConfigLocation](#) that represents the subpath being retrieved.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT varIndex,
    [out, retval] IAppHostConfigLocation** ppLocation
);
```

varIndex: A [VARIANT](#), which is used to specify the **IAppHostConfigLocation** object to retrieve. If the **VARIANT** is of type integer, the index is a zero-based index to the collection of [IAppHostElement](#) objects, where 0 indicates the first **IAppHostElement** object, 1 the second, and so on. If the **VARIANT** is of type string, the index is the name of the subpath being retrieved.

ppLocation: Contains a subpath that represents **IAppHostConfigLocation**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppLocation is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.8.3 AddLocation (Opnum 5)

The **AddLocation** method is received by the server in an RPC_REQUEST packet. In response, the server attempts to create a new subpath container in the [IAppHostConfigFile](#) that provides the specified [IAppHostConfigLocationCollection](#).

```
HRESULT AddLocation(  
    [in, string] BSTR bstrLocationPath,  
    [out, retval] IAppHostConfigLocation** ppNewLocation  
);
```

bstrLocationPath: The new subpath to add.

ppNewLocation: Contains a new subpath container [IAppHostConfigLocation](#) object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppNewLocation is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.8.4 DeleteLocation (Opnum 6)

The **DeleteLocation** method is received by the server in an RPC_REQUEST packet. In response, the server attempts to delete the specific subpath container (the [IAppHostConfigLocation](#) object).

```
HRESULT DeleteLocation(  
    [in] VARIANT cIndex  
);
```

cIndex: A [VARIANT](#), which is used to specify the **IAppHostConfigLocation** object to delete. If the **VARIANT** is of type integer, the index is a zero-based index to the collection of [IAppHostElement](#) objects, where 0 indicates the first **IAppHostElement** object, 1 the second, and so on. If the **VARIANT** is of type string, the index is the name of the subpath being deleted.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.9 IAppHostConfigManager

The **IAppHostConfigManager** provides methods that allow access to available hierarchy paths and the [IAppHostElement](#) objects that are defined within. The **IAppHostConfigManager** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1

Method	Description
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
GetConfigFile	Opnum: 3
GetUniqueConfigPath	Opnum: 4

3.1.4.9.1 GetConfigFile (Opnum 3)

The **GetConfigFile** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostConfigFile](#) for the specific hierarchy path.

The administration system implementation may choose to fail if the specified hierarchy path does not have an **IAppHostConfigFile** container for it. Or it may choose to succeed and create an empty **IAppHostConfigFile** container instead.

```
HRESULT GetConfigFile(
    [in, string] BSTR bstrConfigPath,
    [out, retval] IAppHostConfigFile** ppConfigFile
);
```

bstrConfigPath: The hierarchy path for the **IAppHostConfigFile** to retrieve.

ppConfigFile: Contains an **IAppHostConfigFile** object for the specified path.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppConfigFile is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.9.2 GetUniqueConfigPath (Opnum 4)

The **GetUniqueConfigPath** method is received by the server in an RPC_REQUEST packet. In response, the server returns the deepest hierarchy path (up to the specified hierarchy path) that contains a unique set of [IAppHostElement](#) objects. For example:

Assume: At hierarchy path A, a set of **IAppHostElement** objects exist.

Assume: At hierarchy path B (deeper than A), the identical set of objects exists.

Given these assumptions, **GetUniqueConfigPath(B)** returns path A. In other words, the method returns the shallowest path that contains the identical set of **IAppHostElement** objects as the specified path.

```
HRESULT GetUniqueConfigPath(
    [in, string] BSTR bstrConfigPath,
    [out, retval, string] BSTR* pbstrUniquePath
);
```

bstrConfigPath: The hierarchy path for which to find the shallowest equivalent path.

pbstrUniquePath: Contains the shallowest equivalent path.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrUniquePath is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.10 IAppHostConstantValue

The **IAppHostConstantValue** interface provides methods that access the string names of a specific constant and its corresponding value.

The **IAppHostConstantValue** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
Value	Opnum: 4

3.1.4.10.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server provides the string name of the specified constant.

```
[propget] HRESULT Name(  
    [out, retval] BSTR* pbstrName  
);
```

pbstrName: Contains the name of the constant value.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.10.2 Value (Opnum 4)

The **Value** method is received by the server in an RPC_REQUEST packet. In response, the server returns the integer value of the constant.

```
[propget] HRESULT Value(  
    [out, retval] DWORD* pdwValue  
);
```

pdwValue: Contains the integer value of the specified constant.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.11 IAppHostConstantValueCollection

The **IAppHostConstantValueCollection** provides methods that access a collection of constant values.

The **IAppHostConstantValueCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4

3.1.4.11.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of constant values in the specified [IAppHostConstantValueCollection](#).

```
[propget] HRESULT Count(  
    [out, retval] DWORD* pcCount  
);
```

pcCount: Contains the count of constant values.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.11.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the specified constant value.

```
[propget, id(DISPID_VALUE)] HRESULT Item(  
    [in] VARIANT cIndex,  
    [out, retval] IAppHostConstantValue** ppConstantValue  
);
```

cIndex: A [VARIANT](#) that represents the constant value to retrieve. If the **VARIANT** is of type integer, the index is a zero-based index to the collection of [IAppHostElement](#) objects, where 0 indicates the first **IAppHostElement** object, 1 the second, and so on. If the **VARIANT** is of type string, it is the name of the constant value.

ppConstantValue: Contains the constant value.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppConstantValue is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12 IAppHostElement

The **IAppHostElement** interface provides methods to the base administration system object of this whole project.

The administration system is a container of **IAppHostElement** objects. **IAppHostElement** objects are primarily an n -way-tree object, with each **IAppHostElement** object primarily containing:

- Zero or more [IAppHostProperty](#) objects. Each has its own unique fixed name.
- Zero or more child **IAppHostElement** objects (hence the n -way-tree description). Each has its own unique fixed name.
- An optional collection of zero or more special child **IAppHostElement** objects that are called collection objects. Each typically has the same fixed name.
- Zero or more [IAppHostMethod](#) objects, which are additional extension methods that may be executed on the specified **IAppHostElement** object.

The IAppHostElement interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
Collection	Opnum: 4
Properties	Opnum: 5
ChildElements	Opnum: 6
GetMetadata	Opnum: 7
SetMetadata	Opnum: 8

Method	Description
Schema	Opnum: 9
GetElementByName	Opnum: 10
GetPropertyByName	Opnum: 11
Clear	Opnum: 12
Methods	Opnum: 13

3.1.4.12.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the [IAppHostElement](#) object.

```
[propget] HRESULT Name(
    [out, retval] BSTR* pbstrName
);
```

pbstrName: Contains the name of the **IAppHostElement**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.2 Collection (Opnum 4)

The **Collection** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElementCollection](#) that represents a collection of "collection IAppHostElement" objects. If the specific [IAppHostElement](#) does not support this type of child object, it indicate this in the return.

```
[propget] HRESULT Collection(
    [out, retval] IAppHostElementCollection** ppCollection
);
```

ppCollection: Contains an **IAppHostElementCollection** that represents the collection child elements. If the specific **IAppHostElement** does not support this type of child element, this parameter is NULL.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.3 Properties (Opnum 5)

The **Properties** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostPropertyCollection](#) that contains the [IAppHostProperty](#) objects that are available for this [IAppHostElement](#).

```
[propget] HRESULT Properties(
```

```
[out, retval] IAppHostPropertyCollection** ppProperties
);
```

ppProperties: Contains the **IAppHostPropertyCollection** that represents the collection of **IAppHostProperty** objects that are available.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppProperties is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.4 ChildElements (Opnum 6)

The **ChildElements** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostChildElementCollection](#) that contains child [IAppHostElement](#) objects if any child **IAppHostElement** objects exist.

```
[propget] HRESULT ChildElements(
    [out, retval] IAppHostChildElementCollection** ppElements
);
```

ppElements: Contains an **IAppHostChildElementCollection** if there are child **IAppHostElement** objects in the specified **IAppHostElement** object. Otherwise, it is NULL.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.5 GetMetadata (Opnum 7)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the specific piece of named metadata.

```
HRESULT GetMetaData(
    [in, string] BSTR bstrMetadataType,
    [out, retval] VARIANT* pValue
);
```

bstrMetadataType: The name of the metadata property to fetch.

Value	Meaning
"overrideMode"	A string that represents the explicit override mode of the specified IAppHostElement object. Override mode defines the locking behavior of an IAppHostElement object as it relates to other IAppHostElement objects at deeper hierarchy paths. The administration system can implement override rules, the selection of which is reflected by this metadata. string
"effectiveOverrideMod"	A string that represents the effective override mode of the specified IAppHostElement object. Override mode defines the locking behavior of an IAppHostElement object as it relates to other

Value	Meaning
	IAppHostElement objects at deeper hierarchy paths. The administration system can implement override rules, the selection of which is reflected by this metadata. string
"allowCommit"	A Boolean that represents whether the administration system supports changing or editing of the specified IAppHostElement object. bool
"deepestPathSet"	A string that represents the deepest hierarchy path from which the specified IAppHostElement object was derived. This is only meaningful if this IAppHostElement object is a merged object from multiple sources. This metadata is the path of the deepest source that was merged. string
"deepestFileNameSet"	A string that represents an operating system file path (if applicable) that corresponds to the deepest hierarchy path metadata "deepestPathSet". string
"deepestFileLineNumberSet"	An integer that represents a location within the corresponding metadata "deepestFileNameSet". uint32
"configSource"	A string that represents an optional redirection path from which the specified IAppHostElement object was retrieved. The path is specific to the administration system and is only applicable if the administration system supports alternate redirection paths. string
"childSource"	A string that represents an optional redirection path from which the specified IAppHostElement object was retrieved. The path is specific to the administration system and only applicable if the administration system supports alternate redirection paths. The difference between "configSource" and "childSource" is an implementation detail of the redirection support of the administration system. string
"lockItem"	A Boolean that represents whether the specified IAppHostElement object can be set at deeper hierarchy paths. If true, the item cannot be set deeper. bool
"lockAllElementsExcept"	A string that represents a comma-delimited list of child element names that are allowed to be set at deeper hierarchy paths. All other unlisted child elements are not allowed to be set at deeper paths. string
"lockElements"	A string that represents a comma-delimited list of child element names that are not allowed to be set at deeper hierarchy paths. string

Value	Meaning
"lockAllAttributesExcept"	A string that represents a comma-delimited list of property names that are allowed to be set at deeper hierarchy paths. All other unlisted properties are not allowed to be set at deeper paths. string
"lockAttributes"	A string that represents a comma-delimited list of property names that are not allowed to be set at deeper hierarchy paths. string
other-unnamed	If SetMetadata() is called with another name, that metadata is stored and can be retrieved with GetMetadata().

pValue: Contains a VARIANT that represents the metadata that is retrieved.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.6 SetMetadata (Opnum 8)

The **SetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the value of the metadata from the server administration system.

This method is used to set a metadata property.

```
HRESULT SetMetadata(
    [in, string] BSTR bstrMetadataType,
    [in] VARIANT value
);
```

bstrMetadataType: The name of the metadata property to set.

Value	Meaning
"overrideMode"	A string that represents the explicit override mode of the specified IAppHostElement object. Override mode defines the locking behavior of an IAppHostElement object as it relates to other IAppHostElement objects at deeper hierarchy paths. The administration system is free to implement override rules, the selection of which is reflected by this metadata. string
"configSource"	A string that represents an optional redirection path from which the specified IAppHostElement object was retrieved. The path is specific to the administration system and only applicable if the administration system supports alternate redirection paths. string
"childSource"	A string that represents an optional redirection path from which the specified IAppHostElement object was retrieved. The path is specific to the administration system and only applicable if the administration system supports alternate redirection paths. The difference between "configSource" and "childSource" is totally an implementation detail of

Value	Meaning
	the redirection support of the administration system. string
"lockItem"	A Boolean that represents whether the specified IAppHostElement object can be set at deeper hierarchy paths. If true, the item cannot be set deeper. bool
"lockAllElementsExcept"	A string that represents a comma-delimited list of child element names that are allowed to be set at deeper hierarchy paths. All other unlisted child elements are not allowed to be set at deeper paths. string
"lockElements"	A string that represents a comma-delimited list of child element names that are not allowed to be set at deeper hierarchy paths. string
"lockAllAttributesExcept"	A string that represents a comma-delimited list of property names that are allowed to be set at deeper hierarchy paths. All other unlisted properties are not allowed to be set at deeper paths. string
"lockAttributes"	A string that represents a comma-delimited list of property names that are not allowed to be set at deeper hierarchy paths. string
other-unnamed	If SetMetadata() is called with another name, that metadata is stored and can be retrieved with GetMetadata().

value: The value of the metadata property to set.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.7 Schema (Opnum 9)

The **Schema** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElementSchema](#) that represents the schema and constraints for the specified [IAppHostElement](#) object.

```
[propget] HRESULT Schema(
    [out, retval] IAppHostElementSchema** ppSchema
);
```

ppSchema: Contains schema of the specified **IAppHostElement** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.8 GetElementByName (Opnum 10)

The **GetElementByName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the child [IAppHostElement](#) object with the specified name.

```
HRESULT GetElementByName(  
    [in, string] BSTR bstrSubName,  
    [out, retval] IAppHostElement** ppElement  
);
```

bstrSubName: The name of the child element to retrieve.

ppElement: Contains the child **IAppHostElement** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppElement is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.9 GetPropertyByName (Opnum 11)

The **GetPropertyByName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostElement](#) of the specified name.

```
HRESULT GetPropertyByName(  
    [in, string] BSTR bstrSubName,  
    [out, retval] IAppHostProperty** ppProperty  
);
```

bstrSubName: The name of the **IAppHostElement** to fetch.

ppProperty: Contains an **IAppHostElement** of the specified name.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppProperty is NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.10 Clear (Opnum 12)

The **Clear** method is received by the server in an RPC_REQUEST packet. In response, the server clears the contents of the specified [IAppHostElement](#) object. The administration system may choose to do the clear in memory or may persist it directly to permanent storage.

```
HRESULT Clear();
```

This method has no parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.12.11 Methods (Opnum 13)

The **Methods** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostMethodCollection](#), which is the collection of methods that are supported for the specific [IAppHostElement](#) object.

```
[propget] HRESULT Methods(  
    [out, retval] IAppHostMethodCollection** ppMethods  
);
```

ppMethods: Contains an **IAppHostMethodCollection**, which is a collection of methods that are supported for the specific **IAppHostElement** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppMethods is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13 IAppHostElementCollection

The **IAppHostElementCollection** provides methods that access a collection of "collection IAppHostElements".

"Collection IAppHostElements" are a special class of child [IAppHostElement](#) objects where all objects typically share the same name but contain different [IAppHostProperty](#) objects. This **IAppHostElementCollection** is a collection of these special objects.

The **IAppHostElementCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Returns the count of elements. Opnum: 3
Item	Returns the value that is associated with the element at the specified index. Opnum: 4
AddElement	Adds an element to the collection. Opnum: 5
DeleteElement	Deletes an element from the collection. Opnum: 6
Clear	Clears an element from the collection.

Method	Description
	Opnum: 7
CreateNewElement	Creates a new element in the collection. Opnum: 8
Schema	Returns the Document Object Model (DOM) description for the collection. Opnum: 9

3.1.4.13.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns a count of the number of [IAppHostElement](#) objects in the collection.

```
[propget] HRESULT Count(
    [out, retval] DWORD* pcElementCount
);
```

pcElementCount: Contains the number of elements that are contained in the element collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElement](#) for the specified index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT cIndex,
    [out, retval] IAppHostElement** ppElement
);
```

cIndex: A [VARIANT](#) that specifies the **IAppHostElement** object to return from the collection. If the **VARIANT** is of type integer, the index is a zero-based index to the collection, where 0 indicates the first **IAppHostElement**, 1 the second, and so on. If the **VARIANT** is of type **IAppHostElement**, the index is a "selector" **IAppHostElement** to the specified collection.

ppElement: Contains the collection **IAppHostElement**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppElement is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13.3 AddElement (Opnum 5)

The **AddElement** method is received by the server in an RPC_REQUEST packet. In response, the server attempts to add a newly created [IAppHostElement](#) object to the specified collection.

This function adds an element to the current collection of elements.

```
HRESULT AddElement (
    [in] IAppHostElement* pElement,
    [in, defaultvalue(-1)] long cPosition
);
```

pElement: The element to add to the collection.

cPosition: The position at which the element was added to the collection. If -1, the new element is appended. Otherwise, the position represents the zero-based index to the collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13.4 DeleteElement (Opnum 6)

The **DeleteElement** method is received by the server in an RPC_REQUEST packet. In response, the server deletes the [IAppHostElement](#) at the specified index.

```
HRESULT DeleteElement (
    [in] VARIANT cIndex
);
```

cIndex: A [VARIANT](#) that specifies the **IAppHostElement** object to return from the collection. If the **VARIANT** is of type integer, the index is a zero-based index to the collection, where 0 indicates the first **IAppHostElement**, 1 the second, and so on. If the **VARIANT** is of type **IAppHostElement**, the index is a "selector" **IAppHostElement** to the specified collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13.5 Clear (Opnum 7)

The **Clear** method is received by the server in an RPC_REQUEST packet. In response, the server clears the specified [IAppHostElementCollection](#) of all the collection [IAppHostElement](#) objects in it.

```
HRESULT Clear();
```

This method has no parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13.6 CreateNewElement (Opnum 8)

The **CreateNewElement** method is received by the server in an RPC_REQUEST packet. In response, the server creates a new unattached [IAppHostElement](#) object that has the specified name. The name must be a supported name as defined by the [IAppHostCollectionSchema](#) of the specified [IAppHostElementCollection](#).

```

HRESULT CreateNewElement(
    [in, string, defaultvalue("")]
    BSTR bstrElementName,
    [out, retval] IAppHostElement** ppElement
);

```

bstrElementName: The name of the **IAppHostElement** to be created.

ppElement: Contains a new **IAppHostElement** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppElement is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.13.7 Schema (Opnum 9)

The **Schema** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostCollectionSchema](#) for the specified [IAppHostElementCollection](#), which represents the schema and constraints of the specified collection.

```

[propget] HRESULT Schema(
    [out, retval] IAppHostCollectionSchema** ppSchema
);

```

ppSchema: Contains the schema object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14 IAppHostElementSchema

The **IAppHostElementSchema** provides methods that access the schema and constraints of a specific [IAppHostElement](#) object.

The **IAppHostElementSchema** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Returns the name of the element entry. Opnum: 3

Method	Description
DoesAllowUnschematizedProperties	Determines if the section allows unrecognized attributes. Opnum: 4
GetMetadata	Used to get a metadata property. (No metadata is currently associated with this interface.) Opnum: 5
CollectionSchema	Opnum: 6
ChildElementSchemas	Opnum: 7
PropertySchemas	Opnum: 8
IsCollectionDefault	Opnum: 9

3.1.4.14.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the corresponding [IAppHostElement](#) from which the [IAppHostElementSchema](#) was retrieved.

```
[propget] HRESULT Name(
    [out, string, retval] BSTR* pbstrName
);
```

pbstrName: Contains the name of the element.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14.2 DoesAllowUnschematizedProperties (Opnum 4)

The **DoesAllowUnschematizedProperties** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the corresponding [IAppHostElement](#) supports [IAppHostProperty](#) objects.

```
[propget] HRESULT DoesAllowUnschematizedProperties(
    [out, retval] VARIANT_BOOL* pfAllowUnschematized
);
```

pfAllowUnschematized: Contains a Boolean that indicates whether the element supports unschematized [IAppHostProperty](#) objects.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14.3 GetMetadata (Opnum 5)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the specific named metadata.

```
HRESULT GetMetadata(  
    [in, string] BSTR bstrMetadataType,  
    [out, retval] VARIANT* pValue  
);
```

bstrMetadataType: The name of the metadata property to fetch.

pValue: Returns the value of the metadata property that is specified, and the type depends on the property that is fetched.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14.4 CollectionSchema (Opnum 6)

The **CollectionSchema** method is received by the server in an RPC_REQUEST packet. If the specified [IAppHostElement](#) object supports child collection elements, the server returns the schema and constraints of the collection that is contained in the corresponding **IAppHostElement** object.

```
[propget] HRESULT CollectionSchema(  
    [out, retval] IAppHostCollectionSchema** ppCollectionSchema  
);
```

ppCollectionSchema: Contains an [IAppHostCollectionSchema](#).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppCollectionSchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14.5 ChildElementSchemas (Opnum 7)

The **ChildElementSchemas** method is received by the server in an RPC_REQUEST packet. In response, the server returns the schema and constraints of any child elements that are contained in the corresponding [IAppHostElement](#) object.

```
[propget] HRESULT ChildElementSchemas(  
    [out, retval] IAppHostElementSchemaCollection** ppChildSchemas  
);
```

ppChildSchemas: Contains the schema collection of the supported child **IAppHostElement** objects.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppChildSchemas is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14.6 PropertySchemas (Opnum 8)

The **PropertySchemas** method is received by the server in an RPC_REQUEST packet. In response, the server returns the schema and constraints for the [IAppHostProperty](#) objects that are contained in the corresponding [IAppHostElement](#).

```
[propget] HRESULT PropertySchemas(  
    [out, retval] IAppHostPropertySchemaCollection** ppPropertySchemas  
);
```

ppPropertySchemas: Contains the collection of **IAppHostProperty** schema.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppPropertySchemas is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.14.7 IsCollectionDefault (Opnum 9)

The **IsCollectionDefault** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the corresponding [IAppHostElement](#) object is also considered to be a supported default for other **IAppHostElement** objects in the administration system.

```
[propget] HRESULT IsCollectionDefault(  
    [out, retval] VARIANT_BOOL* pfIsCollectionDefault  
);
```

pfIsCollectionDefault: Contains a Boolean that indicates whether the corresponding **IAppHostElement** object is a default for other **IAppHostElement** objects.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.15 IAppHostElementSchemaCollection

The **IAppHostElementSchemaCollection** interface provides methods that access a collection of schema and constraints for child [IAppHostElement](#) objects that are supported by the corresponding **IAppHostElement** object.

The **IAppHostElementSchemaCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use.

Method	Description
	Opnum: 2
Count	Opnum: 3
Item	Opnum: 4

3.1.4.15.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of [IAppHostElementSchema](#) objects that are contained in the collection.

```
[propget] HRESULT Count(
    [out, retval] DWORD* pcCount
);
```

pcCount: Contains the count in the collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.15.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostElementSchema](#) that is specified by the specific index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT cIndex,
    [out, retval] IAppHostElementSchema** ppElementSchema
);
```

cIndex: A [VARIANT](#) that specifies the **IAppHostElementSchema** to retrieve from the collection. If the **VARIANT** is of type integer, this is a zero-based index to the collection. If the **VARIANT** is of type string, this is a string index that represents the name of the **IAppHostElementSchema** that is being retrieved.

ppElementSchema: Contains the **IAppHostElementSchema** that is being selected.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppElementSchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.16 IAppHostMappingExtension

The **IAppHostMappingExtension** interface provides methods that access the path hierarchy mapping system of the administration system.

The administration system implements a path hierarchy system that maps paths to potential [IAppHostElement](#) containers. Some details of the path hierarchy are exposed to the user in this **IAppHostMappingExtension**.

The **IAppHostMappingExtension** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
GetSiteNameFromSiteId	Opnum: 3
GetSiteIdFromSiteName	Opnum: 4
GetSiteElementFromSiteId	Opnum: 5
MapPath	Opnum: 6

3.1.4.16.1 GetSiteNameFromSiteID (Opnum 3)

The **GetSiteNameFromSiteId** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string name for the specific integer site ID, which is a concept that is implemented on the administration system (it is an implementation detail).

```
HRESULT GetSiteNameFromSiteId(  
    [in] DWORD dwSiteId,  
    [out, retval, string] BSTR* pbstrSiteName  
);
```

dwSiteId: The unique site ID number that is represented in a double-word format. Note that "0" is not a valid site ID.

pbstrSiteName: Contains the string that represents the unique site name for the specified site ID.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrSiteName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.16.2 GetSiteIdFromSiteName (Opnum 4)

The **GetSiteIdFromSiteName** method is received by the server in an RPC_REQUEST packet. In response, the server returns a unique integer ID for the specific site name. Site name and ID are implementation details of the administration system.

```
HRESULT GetSiteIdFromSiteName(  
    [in, string] BSTR bstrSiteName,  
    [out, retval] DWORD* pdwSiteId
```

```
);
```

bstrSiteName: The unique site name string.

pdwSiteId: Contains the double-word value that represents the unique site ID for the specified name. Note that "0" is NOT a valid site ID.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.16.3 GetSiteElementFromSiteID (Opnum 5)

The **GetSiteElementFromSiteId** method is received by the server in an RPC_REQUEST packet. In response, the server obtains the site section element from a specific site ID in order to access site configuration and properties.

```
HRESULT GetSiteElementFromSiteId(  
    [in] DWORD dwSiteId,  
    [out, retval] IAppHostElement** ppSiteElement  
);
```

dwSiteId: The unique site ID number that is represented in a double-word format.

ppSiteElement: Returns an administration element object that represents the site element of the site section and includes properties, metadata, and methods.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.16.4 MapPath (Opnum 6)

The **MapPath** method is received by the server in an RPC_REQUEST packet. In response, the server returns how the administration system maps the specific site name and virtual path into an optionally returned implementation-specific operating system physical path, and optionally returned the [IAppHostElement](#) objects that contain the definition of the mapping hierarchy that applies.

The "mapping hierarchy that applies" means describing the matching rules that the administration system uses to map a specific hierarchy path. Because there are two returned **IAppHostElement** objects, this allows the server protocol implementation to provide a two-level mapping system where a path is first matched to an application and then mapped to a virtual directory. Regardless of what these two concepts (application and virtual directory) represent, their mapping details are contained in the optionally returned parameters.

```
HRESULT MapPath(  
    [in, string] BSTR bstrSiteName,  
    [in, string] BSTR bstrVirtualPath,  
    [out, string] BSTR* pbstrPhysicalPath,  
    [out] IAppHostElement** ppVirtualDirectoryElement,  
    [out] IAppHostElement** ppApplicationElement  
);
```

bstrSiteName: The unique site name string.

bstrVirtualPath: A hierarchy path to map.

pbstrPhysicalPath: Returns an implementation-specific physical path in the file system to where the virtual path maps. Optional.

ppVirtualDirectoryElement: Returns the second-level mapped virtual directory that is matched in the mapping. Optional.

ppApplicationElement: Returns the first-level mapped application that is matched in the mapping. Optional.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. For each of the output parameters, in this case, they will not be NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.17 IAppHostMethod

The **IAppHostMethod** interface provides methods that access a custom method that is optionally supported on a specific [IAppHostElement](#) object.

IAppHostElement objects provide a means for an administration system to support custom-defined methods that can be executed against a specific **IAppHostElement** object. The methods are executed on the server side and the implementation of these custom methods is not exposed to the client.

The **IAppHostMethod** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
Schema	Opnum: 4
CreateInstance	Opnum: 5

3.1.4.17.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the custom method.

```
[propget] HRESULT Name(  
    [out, string, retval] BSTR* pbstrName
```

```
);
```

pbstrName: Contains the name of the method.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.17.2 Schema (Opnum 4)

The **Schema** method is received by the server in an RPC_REQUEST packet. In response, the server returns the schema and constraints for the specified custom [IAppHostMethod](#).

```
[propget] HRESULT Schema(  
    [out, retval] IAppHostMethodSchema** ppMethodSchema  
);
```

ppMethodSchema: Contains the schema of the specified **IAppHostMethod**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppMethodSchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.17.3 CreateInstance (Opnum 5)

The **CreateInstance** method is received by the server in an RPC_REQUEST packet. In response, the server creates an instance object of the method that can be executed. This behavior is analogous to the stack frame of a native method call.

```
HRESULT CreateInstance(  
    [out, retval] IAppHostMethodInstance** ppMethodInstance  
);
```

ppMethodInstance: Contains the method instance.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppMethodInstance is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.18 IAppHostMethodCollection

The **IAppHostMethodCollection** interface provides methods that access a collection of supported [IAppHostMethod](#) objects.

The **IAppHostMethodCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use.

Method	Description
	Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4

3.1.4.18.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of [IAppHostMethod](#) objects in the collection.

```
[propget] HRESULT Count(
    [out, retval] DWORD* pcCount
);
```

pcCount: Contains the count of IAppHostMethod objects.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.18.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostMethod](#) that is specified by the index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT cIndex,
    [out, retval] IAppHostMethod** ppMethod
);
```

cIndex: A [VARIANT](#) that specifies which **IAppHostMethod** is being selected from the collection. If the **VARIANT** is of type integer, the index is a zero-based index to the collection. If the **VARIANT** is of type string, the index is the name of the method being accessed.

ppMethod: Contains the **IAppHostMethod** that is being retrieved.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppMethod is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.19 IAppHostMethodInstance

The **IAppHostMethodInstance** provides methods that access a specific invocation instance of the corresponding [IAppHostMethod](#). The caller sets parameters and then executes the instance of the method.

The **IAppHostMethodInstance** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Input	Opnum: 3
Output	Opnum: 4
Execute	Opnum: 5
GetMetadata	Opnum: 6
SetMetadata	Opnum: 7

3.1.4.19.1 Input (Opnum 3)

The **Input** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElement](#) in which input parameters may be specified for the specified method instance call. If the method does not support input parameters, no **IAppHostElement** is returned.

```
[propget] HRESULT Input(  
    [out, retval] IAppHostElement** ppInputElement  
);
```

ppInputElement: Contains the **IAppHostElement** that represents the input parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.19.2 Output (Opnum 4)

The **Output** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IAppHostElement](#) in which output parameters may be retrieved after the specified method instance has been executed. If the method does not support output parameters, no **IAppHostElement** is returned.

```
[propget] HRESULT Output(
    [out, retval] IAppHostElement** ppOutputElementg
);
```

ppOutputElementg: Contains the **IAppHostElement** that represents the input parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.19.3 Execute (Opnum 5)

The **Execute** method is received by the server in an RPC_REQUEST packet. In response, the server actually executes the specified custom method.

```
HRESULT Execute();
```

This method has no parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.19.4 GetMetadata (Opnum 6)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the specific piece of named metadata.

Currently no metadata type has been defined.

```
HRESULT GetMetadata(
    [in, string] BSTR bstrMetadataType,
    [out, retval] VARIANT* pValue
);
```

bstrMetadataType: The name of the metadata being retrieved.

pValue: A [VARIANT](#) that contains the metadata value.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.19.5 SetMetadata (Opnum 7)

The **SetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server sets the specific named metadata.

```
HRESULT SetMetadata(
    [in, string] BSTR bstrMetadataType,
    [in] VARIANT value
);
```

);

bstrMetadataType: The name of the metadata being set.

value: A [VARIANT](#) that contains the new value of metadata.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.20 IAppHostMethodSchema

The **IAppHostMethodSchema** interface provides methods that access the schema and constraints of the corresponding [IAppHostMethod](#).

The **IAppHostMethodSchema** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
InputSchema	Opnum: 4
OutputSchema	Opnum: 5
GetMetadata	Opnum: 6

3.1.4.20.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the specified [IAppHostMethod](#).

```
[propget] HRESULT Name(  
    [out, string, retval] BSTR* pbstrName  
);
```

pbstrName: Contains the name of the **IAppHostMethod**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.20.2 InputSchema (Opnum 4)

The **InputSchema** method is received by the server in an RPC_REQUEST packet. In response, the server returns the schema and constraints of the input parameters to the method call. This can be NULL if no input parameters are defined for the method.

```
[propget] HRESULT InputSchema(  
    [out, retval] IAppHostElementSchema** ppInputSchema  
);
```

ppInputSchema: Contains the input parameter schema.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.20.3 OutputSchema (Opnum 5)

The **OutputSchema** method is received by the server in an RPC_REQUEST packet. In response, the server returns the schema and constraints of the output parameters to the method call. This can be NULL if no output parameters are defined for the method.

```
[propget] HRESULT OutputSchema(  
    [out, retval] IAppHostElementSchema** ppOutputSchema  
);
```

ppOutputSchema: Contains the output parameter schema.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.20.4 GetMetadata (Opnum 6)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the named metadata for the specified method schema.

```
HRESULT GetMetadata(  
    [in, string] BSTR bstrMetadataType,  
    [out, retval] VARIANT* pValue  
);
```

bstrMetadataType: The name of the metadata.

pValue: Contains the value of the metadata.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.21 IAppHostPathMapper

The **IAppHostPathMapper** interface provides methods that are called by the server implementation when the server informs the client about hierarchy mapping decisions.

To receive incoming remote calls for this interface, the client must implement a **UUID** (e7927575-5cc3-403b-822e-328a6b904bee). It must then specify an object that implements this interface to the [IAppHostAdminManager::SetMetadata\(\)](#) method with a bstrMetadataName of "pathMapper".

As an administration system maps hierarchy paths to physical paths on the server, it optionally calls this client-supplied object that implements the **IAppHostPathMapper** interface. The implementer of this interface receives details of all mappings and can change the results of each mapping if required.

The **IAppHostPathMapper** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
MapPath	Opnum: 3

3.1.4.21.1 MapPath (Opnum 3)

The **MapPath** method is called by the server in an RPC_REQUEST packet. In response, the client implementation receives the details of the specific mapping decision and optionally, can change the results by using its return.

```
HRESULT MapPath(  
    [in, string] BSTR bstrConfigPath,  
    [in, string] BSTR bstrMappedPhysicalPath,  
    [out, string, retval] BSTR* pbstrNewPhysicalPath  
);
```

bstrConfigPath: The hierarchy path being mapped.

bstrMappedPhysicalPath: The server side physical path that the administration system has determined maps to the specified hierarchy path.

pbstrNewPhysicalPath: Set to the new or updated physical path to use for the mapping. If the mapping stays the same, the client implementer should return the identical physical path that was passed in as *bstrMappedPhysicalPath*.

Return Values: The client MUST return zero if it successfully processes the message that is received from the client. The client MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22 IAppHostProperty

The **IAppHostProperty** interface provides methods that access properties that may be contained under an [IAppHostElement](#) object.

IAppHostElement objects may contain zero or more **IAppHostProperty** objects. These **IAppHostProperty** objects typically represent a single setting or attribute set on an **IAppHostElement** but can be multiple settings or sets.

The **IAppHostProperty** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
Value	"getter" Opnum: 4
Value	"setter" Opnum: 5
Clear	Opnum: 6
StringValue	Opnum: 7
Exception	Opnum: 8
GetMetadata	Opnum: 9
SetMetadata	Opnum: 10
Schema	Opnum: 11

3.1.4.22.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the specific [IAppHostProperty](#).

```
[propget] HRESULT Name(  
    [out, retval] BSTR* pbstrName
```

```
);
```

pbstrName: Contains the name of the property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.2 Value (Get) (Opnum 4)

The **Value (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns the value of the specified property.

```
[propget] HRESULT Value(  
    [out, retval] VARIANT* pVariant  
);
```

pVariant: Contains the [VARIANT](#) value of the property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.3 Value (Set) (Opnum 5)

The **Value (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server sets the value of the specified property.

```
[propset] HRESULT Value(  
    [in] VARIANT value  
);
```

value: The new [VARIANT](#) value of the property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.4 Clear (Opnum 6)

The **Clear** method is received by the server in an RPC_REQUEST packet. In response, the server clears the value of the specified property.

```
HRESULT Clear();
```

This method has no parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.5 StringValue (Opnum 7)

The **StringValue** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string representation of the value of the specified property.

```
[propget] HRESULT StringValue(  
    [out, retval] BSTR* pbstrValue  
);
```

pbstrValue: Contains the string value that represents the property value.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrValue is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.6 Exception (Opnum 8)

The **Exception** method is received by the server in an RPC_REQUEST packet. In response, the server returns administration system exception information that is related to the processing of the specified property.

```
[propget] HRESULT Exception(  
    [out, retval] IAppHostPropertyException** ppException  
);
```

ppException: Contains the exception information for a specified property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppException is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.7 GetMetadata (Opnum 9)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns the named metadata for the specified property.

```
HRESULT GetMetadata(  
    [in, string] BSTR bstrMetadataType,  
    [out, retval] VARIANT* pValue  
);
```

bstrMetadataType: The name of the metadata property to fetch.

Value	Meaning
"encryptionProvider"	A string that represents server-specific data that defines how the property should be encrypted or decrypted on the server. string
"isPropertyEncrypted"	A Boolean that determines if the property is encrypted. bool

Value	Meaning
"isDefaultValue"	A Boolean that represents whether the specified property was explicitly set in the administration system or is a system default. bool
"isInheritedFromDefault"	A Boolean that represents whether the specified property was explicitly set in the administration system or is a globally defined default. bool
"isLocked"	A Boolean that represents whether this property can be set at deeper hierarchy paths. bool

pValue: Contains the [VARIANT](#) value of the metadata property that is specified and the type depends on the property that is fetched.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.8 SetMetadata (Opnum 10)

The **SetMetadata** method is received by the server in an RPC_REQUEST packet.

This method is used to set a metadata property.

```
HRESULT SetMetadata(
    [in, string] BSTR bstrMetadataType,
    [in] VARIANT value
);
```

bstrMetadataType: The name of the metadata to set on the property.

Value	Meaning
"encryptionProvider"	A string that represents server-specific data that defines how the property should be encrypted or decrypted on the server. string

value: The value of the metadata property to set.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.22.9 Schema (Opnum 11)

The **Schema** method is received by the server in an RPC_REQUEST packet. In response, the server returns the schema and constraints of the specified property, as defined in the [IAppHostPropertySchema](#) object that is returned.

```
[propget] HRESULT Schema(
```

```
[out, retval] IAppHostPropertySchema** ppSchema
);
```

ppSchema: Set with the **IAppHostPropertySchema** for the corresponding [IAppHostProperty](#).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.23 IAppHostPropertyCollection

The **IAppHostPropertyCollection** interface provides methods that access a collection of [IAppHostProperty](#) objects that are supported by a corresponding [IAppHostElement](#).

The **IAppHostPropertyCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4

3.1.4.23.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of IAppHostProperty objects that are contained in the collection.

```
[propget] HRESULT Count(
    [out, retval] DWORD* pcCount
);
```

pcCount: Contains the count of the number of properties in the collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.23.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the property that is specified by the specific index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT cIndex,
    [out, retval] IAppHostProperty** ppProperty
);
```

cIndex: The [VARIANT](#) index of the property to be fetched. If the **VARIANT** is of type integer, the index is a zero-based index to the collection. If the **VARIANT** is of type string, the index is a string that is the name of the property being retrieved.

ppProperty: Contains the retrieved property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppProperty is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.24 IAppHostPropertyException

The **IAppHostPropertyException** provides methods that access the exception information that the administration system encountered when processing the corresponding [IAppHostProperty](#). The administration system may indicate errors as encountered by filling in this exception. This behavior is defined by the [IAppHostAdminManager](#) metadata "ignoreInvalidAttributes".

The **IAppHostPropertyException** interface inherits the [IAppHostConfigException](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Opnum3NotUsedOnWire	Reserved for local use. Opnum: 3
Opnum4NotUsedOnWire	Reserved for local use. Opnum: 4
Opnum5NotUsedOnWire	Reserved for local use. Opnum: 5
Opnum6NotUsedOnWire	Reserved for local use. Opnum: 6
Opnum7NotUsedOnWire	Reserved for local use. Opnum: 7
Opnum8NotUsedOnWire	Reserved for local use. Opnum: 8

Method	Description
Opnum9NotUsedOnWire	Reserved for local use. Opnum: 9
<u>InvalidValue</u>	Opnum: 10
<u>ValidationFailureReason</u>	Opnum: 11
<u>ValidationFailureParameters</u>	Opnum: 12

3.1.4.24.1 InvalidValue (Opnum 10)

The **InvalidValue** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string representation of the invalid value that is encountered by the administration system when processing property.

```
[propget] HRESULT InvalidValue(
    [out, string, retval] BSTR* pbstrValue
);
```

pbstrValue: Contains the invalid value representation.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrValue is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.24.2 ValidationFailureReason (Opnum 11)

The **ValidationFailureReason** method is received by the server in an RPC_REQUEST packet. In response, the server returns a description of the error that is encountered.

```
[propget] HRESULT ValidationFailureReason(
    [out, string, retval] BSTR* pbstrValidationReason
);
```

pbstrValidationReason: Contains a description of the error.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrValidationReason is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.24.3 ValidationFailureParameters (Opnum 12)

The **ValidationFailureParameters** method is received by the server in an RPC_REQUEST packet. In response, the server returns any additional parameters that relate to the reason for the failure that is communicated in the ValidationFailureReason method.

```
[propget] HRESULT ValidationFailureParameters(
    [out, retval] SAFEARRAY(VARIANT)* pParameterArray
);
```

pParameterArray: Contains the [VARIANT](#) array of parameters that are applicable to the error.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25 IAppHostPropertySchema

The **IAppHostPropertySchema** interface provides methods that access the schema and constraints for the corresponding [IAppHostProperty](#) object.

The **IAppHostPropertySchema** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
Type	Opnum: 4
DefaultValue	Opnum: 5
IsRequired	Opnum: 6
IsUniqueKey	Opnum: 7
IsCombinedKey	Opnum: 8
IsExpanded	Opnum: 9
ValidationType	Opnum: 10
ValidationParameter	Opnum: 11
GetMetadata	Opnum: 12
IsCaseSensitive	Opnum: 13
PossibleValues	Opnum: 14
DoesAllowInfinite	Opnum: 15
IsEncrypted	Opnum: 16
TimeSpanFormat	Opnum: 17

3.1.4.25.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the corresponding [IAppHostProperty](#).

```
[propget] HRESULT Name(  
    [out, string, retval] BSTR* pbstrName  
);
```

pbstrName: Contains the name of the corresponding property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.2 Type (Opnum 4)

The **Type** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string that represents the type of the property. Which types are supported is a server implementation detail.

```
[propget] HRESULT Type(  
    [out, string, retval] BSTR* pbstrType  
);
```

pbstrType: Contains the string that represents the type.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrType is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.3 DefaultValue (Opnum 5)

The **DefaultValue** method is received by the server in an RPC_REQUEST packet. In response, the server returns the system-wide default value for the specified property, as defined by the administration system.

```
[propget] HRESULT DefaultValue(  
    [out, retval] VARIANT* pDefaultValue  
);
```

pDefaultValue: A [VARIANT](#) that contains the system default value.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.4 IsRequired (Opnum 6)

The **IsRequired** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the specified property is required to be set on the server when the parent [IAppHostElement](#) exists.

```
[propget] HRESULT IsRequired(
    [out, retval] VARIANT_BOOL* pfIsRequired
);
```

pfIsRequired: A Boolean value that states whether an attribute is required to be present in the parent **IAppHostElement**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.5 IsUniqueKey (Opnum 7)

The **IsUniqueKey** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the specified property must be unique compared to all other properties of the peer collection of **IAppHostElement** objects. In other words, it applies only to properties that are members of the collection of **IAppHostElement** objects.

```
[propget] HRESULT IsUniqueKey(
    [out, retval] VARIANT_BOOL* pfIsUniqueKey
);
```

pfIsUniqueKey: A Boolean value that states whether an attribute is required to be unique in the specified collection of the parent **IAppHostElement**.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.6 IsCombinedKey (Opnum 8)

The **IsCombinedKey** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the specified property is part of a group of properties that combine to be unique compared to all other properties of peer collection **IAppHostElement** objects. In other words, it applies only to properties that are members of collection **IAppHostElement** objects.

```
[propget] HRESULT IsCombinedKey(
    [out, retval] VARIANT_BOOL* pfIsCombinedKey
);
```

pfIsCombinedKey: A Boolean value that states whether an attribute is part of a combined key in the specified collection of the parent **IAppHostElement** object.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.7 IsExpanded (Opnum 9)

The **IsExpanded** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the specified property supports being expanded on the server side to expand any embedded system environment variables.

```
[propget] HRESULT IsExpanded(
    [out, retval] VARIANT_BOOL* pfIsExpanded
);
```

pfIsExpanded: Set to whether the specified property supports environment variable expansion.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.8 ValidationType (Opnum 10)

The **ValidationType** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string representing additional custom validation done when processing the corresponding property. The details of the validation are an implementation detail of the administration system.

```
[propget] HRESULT ValidationType(
    [out, retval, string] BSTR* pbstrValidationType
);
```

pbstrValidationType: Set to name of validation type performed on server.

Return Values: The server MUST return zero if it successfully processes the message received from the client. In this case, *pbstrValidationType is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.9 ValidationParameter (Opnum 11)

The **ValidationParameter** method is received by the server in an RPC_REQUEST packet. In response, the server returns any parameter that applies to the [ValidationType](#) of the specified property. Again, this is implementation-specific.

```
[propget] HRESULT ValidationParameter(
    [out, retval, string] BSTR* pbstrValidationParameter
);
```

pbstrValidationParameter: Set to the parameter of the validation type.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, pbstrValidationParameter is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.10 GetMetadata (Opnum 12)

The **GetMetadata** method is received by the server in an RPC_REQUEST packet. In response, the server returns any named metadata for the property schema.

No metadata is currently associated with this interface.

```
HRESULT GetMetadata(
```



```
[in, string] BSTR bstrMetadataType,
[out, retval] VARIANT* pValue
);
```

bstrMetadataType: The name of the metadata property to fetch.

pValue: Returns the value of the specified metadata property, and the type depends on the property fetched.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.11 IsCaseSensitive (Opnum 13)

The **IsCaseSensitive** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the corresponding property is compared to others in a case-sensitive manner, when determining equality for key (combined/unique) evaluation.

```
[propget] HRESULT IsCaseSensitive(
[out, retval] VARIANT_BOOL* pfIsCaseSensitive
);
```

pfIsCaseSensitive: A Boolean value about whether the property is treated as case-sensitive.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.12 PossibleValues (Opnum 14)

The **PossibleValues** method is received by the server in an RPC_REQUEST packet. In response, the server returns a collection of the possible constant values for the specified property, if applicable. The administration system determines the applicability.

```
[propget] HRESULT PossibleValues(
[out, retval] IAppHostConstantValueCollection** ppValues
);
```

ppValues: Contains the collection of possible values.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppValues is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.13 DoesAllowInfinite (Opnum 15)

The **DoesAllowInfinite** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the property supports having an infinite value set.

```
[propget] HRESULT DoesAllowInfinite(
```

```
[out, retval] VARIANT_BOOL* pfAllowInfinite
);
```

pfAllowInfinite: A pointer to a Boolean value that, if set to TRUE, indicates the property that is represented by this [IAppHostPropertySchema](#) supports infinite values.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.14 IsEncrypted (Opnum 16)

The **IsEncrypted** method is received by the server in an RPC_REQUEST packet. In response, the server returns whether the corresponding [IAppHostProperty](#) should be encrypted when it is persisted in the administration system.

```
[propget] HRESULT IsEncrypted(
    [out, retval] VARIANT_BOOL* pfIsEncrypted
);
```

pfIsEncrypted: A pointer to a Boolean value that, if set to TRUE, indicates the property that is represented by this [IAppHostPropertySchema](#) will be encrypted when persisted.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.25.15 TimeSpanFormat (Opnum 17)

The **TimeSpanFormat** method is received by the server in an RPC_REQUEST packet. In response, the server returns a format string that describes how the corresponding property should be formatted if property represents a time span.

```
[propget] HRESULT TimeSpanFormat(
    [out, retval] BSTR* pbstrTimeSpanFormat
);
```

pbstrTimeSpanFormat: Contains the format string of the time span for the property.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrTimeSpanFormat is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.26 IAppHostPropertySchemaCollection

The **IAppHostPropertySchemaCollection** interface provides methods that access a collection of [IAppHostPropertySchema](#) objects.

The **IAppHostPropertySchemaCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4

3.1.4.26.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of the [IAppHostPropertySchema](#) objects in the specified collection.

```
[propget] HRESULT Count(
    [out, retval] DWORD* pcCount
);
```

pcCount: Contains the count of the **IAppHostPropertySchema** objects in the collection.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.26.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IAppHostPropertySchema](#) that is specified by the index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT cIndex,
    [out, retval] IAppHostPropertySchema** ppPropertySchema
);
```

cIndex: A [VARIANT](#) that specifies the property schema to retrieve. If the **VARIANT** is of type integer, the index is the zero-based index to the collection. If the **VARIANT** is of type string, the index is the string name of the property schema to retrieve.

ppPropertySchema: Contains the **IAppHostPropertySchema** that is selected.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppPropertySchema is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27 IAppHostSectionDefinition

The **IAppHostSectionDefinition** interface provides methods that access a declaration of the [IAppHostElement](#) object that is supported by the administration system. A declaration is distinct from the existence of an **IAppHostElement** in the administration system.

The **IAppHostSectionDefinition** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Name	Opnum: 3
Type	"getter" Opnum: 4
Type	"setter" Opnum: 5
OverrideModeDefault	"getter" Opnum: 6
OverrideModeDefault	"setter" Opnum: 7
AllowDefinition	"getter" Opnum: 8
AllowDefinition	"setter" Opnum: 9
AllowLocation	"getter" Opnum: 10
AllowLocation	"setter" Opnum: 11

3.1.4.27.1 Name (Opnum 3)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the [IAppHostElement](#) being declared.

```
[propget] HRESULT Name(  
    [out, retval] BSTR* pbstrName
```

```
);
```

pbstrName: The name of the **IAppHostElement** being declared.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.2 Type (Get) (Opnum 4)

The **Type (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string that represents an implementation-specific type name for the declaration.

```
[propget] HRESULT Type(  
    [out, retval] BSTR* pbstrType  
);
```

pbstrType: Set to the type of the section.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrType is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.3 Type (Set) (Opnum 5)

The **Type (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server sets the type name of a specified declaration.

```
[propput] HRESULT Type(  
    [in] BSTR bstrType  
);
```

bstrType: The type name to set for the declaration.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.4 OverrideModeDefault (Get) (Opnum 6)

The **OverrideModeDefault (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns an implementation-specific override behavior string for the declaration.

```
[propget] HRESULT OverrideModeDefault(  
    [out, retval] BSTR* pbstrOverrideModeDefault  
);
```

pbstrOverrideModeDefault: Contains the string that represents the override behavior.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, the *pbstrOverrideModeDefault* parameter is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.5 OverrideModeDefault (Set) (Opnum 7)

The **OverrideModeDefault (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server sets an implementation-specific override behavior string for the declaration.

```
[propget] HRESULT OverrideModeDefault(  
    [in] BSTR bstrOverrideModeDefault  
);
```

bstrOverrideModeDefault: The override behavior string.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.6 AllowDefinition (Get) (Opnum 8)

The **AllowDefinition (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns an implementation-specific string that defines where the specified declaration can apply in the administration system.

```
[propget] HRESULT AllowDefinition(  
    [out, retval] BSTR* pbstrAllowDefinition  
);
```

pbstrAllowDefinition: Contains the value that defines where the specified declaration can apply in the administration system.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrAllowDefinition is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.7 AllowDefinition (Set) (Opnum 9)

The **AllowDefinition (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server sets an implementation-specific string that defines where the specified declaration can apply in the administration system.

```
[propset] HRESULT AllowDefinition(  
    [in] BSTR bstrAllowDefinition  
);
```

bstrAllowDefinition: A value that defines where the specified declaration can apply in the administration system.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.8 AllowLocation (Get) (Opnum 10)

The **AllowLocation (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns an implementation-specific string that defines whether the declared [IAppHostElement](#) can exist in subpaths within an [IAppHostConfigFile](#).

```
[propget] HRESULT AllowLocation(  
    [out, retval] BSTR* pbstrAllowLocation  
);
```

pbstrAllowLocation: Contains the subpath behavior.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrAllowLocation is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.27.9 AllowLocation (Set) (Opnum 11)

The **AllowLocation (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server sets an implementation-specific string that defines whether the declared [IAppHostElement](#) can exist in subpaths within an [IAppHostConfigFile](#).

```
[propput] HRESULT AllowLocation(  
    [in] BSTR bstrAllowLocation  
);
```

bstrAllowLocation: The subpath behavior to set.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.28 IAppHostSectionDefinitionCollection

The **IAppHostSectionDefinitionCollection** interface provides methods that access a collection of [IAppHostSectionDefinition](#) objects.

The **IAppHostSectionDefinitionCollection** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3

Method	Description
Item	Opnum: 4
AddSection	Opnum: 5
DeleteSection	Opnum: 6

3.1.4.28.1 Count

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the count of the IAppHostSectionDefinition objects in the collection.

```
[propget] HRESULT Count(
    [out, retval] unsigned long* pcCount
);
```

pcCount: Contains the count of the IAppHostSectionDefinition objects.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.28.2 Item

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns the IAppHostSectionDefinition object that is specified in the index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT varIndex,
    [out, retval] IAppHostSectionDefinition** ppConfigSection
);
```

varIndex: A VARIANT that specifies the section definition to retrieve. If the VARIANT is of type integer, the index is the zero-based index to the collection. If the VARIANT is of type string, the index is the string name of the section definition.

ppConfigSection: Contains the specified section definition.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppConfigSection is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.28.3 AddSection

The **AddSection** method is received by the server in an RPC_REQUEST packet. In response, the server adds a section definition to the administration system.

```
HRESULT AddSection(
    [in, string] BSTR bstrSectionName,
    [out, retval] IAppHostSectionDefinition** ppConfigSection
);
```


bstrSectionName: The name of the new section definition to add.

ppConfigSection: Contains the newly added or created section definition.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppConfigSection is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.28.4 DeleteSection

The **DeleteSection** method is received by the server in an RPC_REQUEST packet. In response, the server deletes the specified section definition.

```
HRESULT DeleteSection(  
    [in] VARIANT varIndex  
);
```

varIndex: A VARIANT index that specifies the section definition to delete. If the VARIANT is of type integer, the index is a zero-based index to the collection. If the VARIANT is of type string, the index is the string name of the section definition.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29 IAppHostSectionGroup

The **IAppHostSectionGroup** provides methods that access a group of section definitions that have a common prefix name.

The **IAppHostSectionGroup** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Count	Opnum: 3
Item	Opnum: 4
Sections	Opnum: 5
AddSectionGroup	Opnum: 6
DeleteSectionGroup	Opnum: 7

Method	Description
Name	Opnum: 8
Type	"getter" Opnum: 9
Type	"setter" Opnum: 10

3.1.4.29.1 Count (Opnum 3)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns a count of child section groups that are contained in the specified section group.

```
[propget] HRESULT Count(
    [out, retval] unsigned long* pcSectionGroup
);
```

pcSectionGroup: Contains the count of section groups in the collection array.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.2 Item (Opnum 4)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns a section group that matches the specified index.

```
[propget, id(DISPID_VALUE)] HRESULT Item(
    [in] VARIANT varIndex,
    [out, retval] IAppHostSectionGroup** ppSectionGroup
);
```

varIndex: A [VARIANT](#) index that specifies the section group. If of type integer, the index is a zero-based index to the collection. If of type string, the index is the name of the section group to retrieve.

ppSectionGroup: Contains the selected [IAppHostSectionGroup](#).

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSectionGroup is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.3 Sections (Opnum 5)

The **Sections** method is received by the server in an RPC_REQUEST packet. In response, the server returns a collection of section definitions in the specified section group.

```
[propget] HRESULT Sections(
    [out, retval] IAppHostSectionDefinitionCollection** ppSections
```

```
);
```

ppSections: Contains the collection of section definitions.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSections is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.4 AddSectionGroup (Opnum 6)

The **AddSectionGroup** method is received by the server in an RPC_REQUEST packet. In response, the server adds a new section group to the specified section group.

```
HRESULT AddSectionGroup(  
    [in, string] BSTR bstrSectionGroupName,  
    [out, retval] IAppHostSectionGroup** ppSectionGroup  
);
```

bstrSectionGroupName: A string that contains the name of the section group to add.

ppSectionGroup: Contains the pointer to the newly created section group.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *ppSectionGroup is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.5 DeleteSectionGroup (Opnum 7)

The **DeleteSectionGroup** method is received by the server in an RPC_REQUEST packet. In response, the server deletes the specified section group.

```
HRESULT DeleteSectionGroup(  
    [in] VARIANT varIndex  
);
```

varIndex: A [VARIANT](#) index that specifies the section group. If of type integer, the index is a zero-based index to the collection. If of type string, the index is the name of the section group to retrieve.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.6 Name (Opnum 8)

The **Name** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name of the specified section group.

```
[propget] HRESULT Name(  
    [out, retval] BSTR* pbstrName  
);
```

pbstrName: Contains the string that contains the name of the section group.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrName is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.7 Type (Get) (Opnum 9)

The **Type (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns an implementation-specific type string for the specified section group.

```
[propget] HRESULT Type(  
    [out, retval] BSTR* pbstrType  
);
```

pbstrType: Contains the string that contains the type string for the section group.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrType is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.29.8 Type (Set) (Opnum 10)

The **Type (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server sets the implementation-specific type string for the specified section group.

```
[propput] HRESULT Type(  
    [in] BSTR bstrType  
);
```

bstrType: The string that contains the type of the section group.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.30 IAppHostWritableAdminManager

The **IAppHostWritableAdminManager** interface provides methods that access a writable version of an administration system. It extends the [IAppHostAdminManager](#), which is a read-only interface. The **IAppHostWritableAdminManager** adds methods to allow writing to the administration system, the most important of which is the CommitChanges method, which instructs the administration system to persist any in-memory changes that it may have accumulated.

The **IAppHostWritableAdminManager** interface inherits the **IAppHostAdminManager** interface, as defined in this protocol specification and the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0

Method	Description
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
Opnum3NotUsedOnWire	Reserved for local use. Opnum: 3
Opnum4NotUsedOnWire	Reserved for local use. Opnum: 4
Opnum5NotUsedOnWire	Reserved for local use. Opnum: 5
Opnum6NotUsedOnWire	Reserved for local use. Opnum: 6
CommitChanges	Opnum: 7
CommitPath	"getter" Opnum: 8
CommitPath	"setter" Opnum: 9

3.1.4.30.1 CommitChanges (Opnum 7)

The **CommitChanges** method is received by the server in an RPC_REQUEST packet. In response, the server commits any in-memory changes that it accumulates to a persisted store. This behavior essentially writes out the changes that are made by the client.

```
HRESULT CommitChanges();
```

This method has no parameters.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.30.2 CommitPath (Get) (Opnum 8)

The **CommitPath (Get)** method is received by the server in an RPC_REQUEST packet. In response, the server returns the hierarchy path that the administration system writes changes to after having its [CommitChanges](#) method called.

```
[propget] HRESULT CommitPath(
    [out, retval, string] BSTR* pbstrCommitPath
);
```

pbstrCommitPath: Contains the hierarchy path where changes are committed.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. In this case, *pbstrCommitPath is not NULL. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.30.3 CommitPath (Set) (Opnum 9)

The **CommitPath (Set)** method is received by the server in an RPC_REQUEST packet. In response, the server changes the hierarchy path where changes are committed.

```
[propput] HRESULT CommitPath(  
    [in, string] BSTR bstrCommitPath  
);
```

bstrCommitPath: The new hierarchy path to commit changes to.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.31 IAppHostMethodExtension

The **IAppHostMethodExtension** interface is used to handle method extensions to the configuration system. When this interface is implemented, the configuration system will call it to handle dynamic configuration.

The administration system allows dynamic methods to be defined, provided that they are backed by a class that implements the **IAppHostMethodExtension** interface.

The **IAppHostMethodExtension** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
ProvideMethod	Opnum: 3

3.1.4.31.1 ProvideMethod

The **ProvideMethod** method is received by the server in an RPC_REQUEST packet. The use of the method is specific to the implementation of the configuration extension. This may include reading input properties and writing output properties on the method instance.

```
HRESULT ProvideMethod(  

```

```

[in] IAppHostMethod* pMethod,
[in] IAppHostMethodInstance* pMethodInstance,
[in] IAppHostElement* pElement
);

```

pMethod: The static [IAppHostMethod](#) object in *pElement*.

pMethodInstance: The current [IAppHostMethodInstance](#) that holds the invocation state.

pElement: The parent [IAppHostElement](#) of *pMethod*.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.32 IAppHostPropertyExtension

The **IAppHostPropertyExtension** interface is used to handle property extensions to the configuration system. When this interface is implemented, the configuration system will call it to handle dynamic configuration.

The administration system allows dynamic properties to be defined, provided that they are backed by a class that implements the **IAppHostPropertyExtension** interface.

The **IAppHostPropertyExtension** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
ProvideGetProperty	Opnum: 3

3.1.4.32.1 ProvideGetProperty

The **ProvideGetProperty** method is received by the server in an RPC_REQUEST packet. If the method succeeds, the value of the property will be filled in.

```

HRESULT ProvideGetProperty(
[in] IAppHostElement* pElement,
[in] IAppHostProperty* pProperty
);

```

pElement: Parent element of the dynamic property.

pProperty: The dynamic property whose value is filled in by the method implementation.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.33 IAppHostElementExtension

The **IAppHostElementExtension** interface is used to handle element extensions to the configuration system. When this interface is implemented, the configuration system will call it to handle dynamic configuration.

The administration system allows dynamic elements to be defined, provided that they are backed by a class that implements the **IAppHostElementExtension** interface.

The **IAppHostElementExtension** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
ProvideElement	Opnum: 3

3.1.4.33.1 ProvideElement

The **ProvideElement** method is received by the server in an RPC_REQUEST packet. In response, the configuration extension can populate the configuration element.

```
HRESULT ProvideElement(  
    [in] IAppHostElement* pNewElement  
);
```

pNewElement: The [IAppHostElement](#) that is to be populated.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.4.34 IAppHostPathMapper2

The **IAppHostPathMapper2** interface provides methods that are called by the server implementation when the server informs the client about hierarchy mapping decisions.

To receive incoming remote calls for this interface, the client must implement a UUID (0f80e901-8f4c-449a-bf90-13d5d082f187). It must then specify an object that implements this interface to the [IAppHostAdminManager::SetMetadata\(\)](#) method with a *bstrMetadataName* of "pathMapper2".

As an administration system maps hierarchy paths to physical paths on the server, it optionally calls this client-supplied object that implements the **IAppHostPathMapper2** interface. The implementer of this interface receives details of all mappings and can change the results of each mapping if required.

The **IAppHostPathMapper2** interface inherits the [IUnknown](#) interface.

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
Opnum2NotUsedOnWire	Reserved for local use. Opnum: 2
MapPath	Opnum: 3

3.1.4.34.1 MapPath

The **MapPath** method is called by the server in an RPC_REQUEST packet. In response, the client implementation receives the details of the specific mapping decision and optionally, can change the results by using its return.

```
HRESULT MapPath(
    [in, string] BSTR bstrConfigPath,
    [in, string] BSTR bstrMappedPhysicalPath,
    [out, string] BSTR* pbstrNewPhysicalPath,
    [out] HANDLE* phImpersonationToken
);
```

bstrConfigPath: The hierarchy path being mapped.

bstrMappedPhysicalPath: The server side physical path that the administration system has determined maps to the specified hierarchy path.

pbstrNewPhysicalPath: Set to the new or updated physical path to use for the mapping. If the mapping stays the same, the client implementer should return the identical physical path that was passed in as *bstrMappedPhysicalPath*.

phImpersonationToken: Set to the impersonation token that was used to access the physical path.

Return Values: The server MUST return zero if it successfully processes the message that is received from the client. The server MUST return an implementation-specific nonzero error code if processing fails.

3.1.5 Timer Events

No timer events are defined by this protocol.

3.1.6 Other Local Events

No other local events are defined by this protocol.

4 Protocol Examples

This section shows examples of client use of this protocol. All examples are shown in JScript development software; however, any client scripting or programming language that can create and manipulate DCOM objects can be used.

For more information and sample usage of the protocol by a client, see [\[MSDN-IIS7AH\]](#).

4.1 Create an AppHostAdminManager Locally

```
var adminManager = WScript.CreateObject("Microsoft.ApplicationHost.AdminManager" );
```

4.2 Get Metadata: Get the overrideMode of a defaultDocument Section

```
var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var overrideMode = configSection.GetMetadata( "overrideMode" );
/* WScript.Echo(overrideMode); */
```

4.3 Set Metadata: Set the overrideMode of the defaultDocument Section

```
var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
configSection.SetMetadata( "overrideMode", "Deny" );
adminManager.CommitChanges();
```

4.4 Create a New Configuration Section Entry in the configSections Section

```
var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";
var configFile = adminManager.ConfigManager.GetConfigFile("MACHINE/WEBROOT/APPHOST");
var sectionGroupTable = configFile.RootSectionGroup;
if ( sectionGroupTable != null )
{
    try
    {
        var sectionGroup = sectionGroupTable.Item("system.webServer");
    }
    catch ( exception )
    {
        var sectionGroup = null;
    }
}
```

```

    }
}
if ( sectionGroup == null )
{
    sectionGroup=sectionGroupTable.AddSectionGroup("system.webServer");
}

var configSection=sectionGroup.Sections.AddSection("NewSectionGroup");
adminManager.CommitChanges();

```

4.5 Get a Section for Read Access: The defaultDocument section

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );

```

4.6 Get a Property: Get the Enabled Property of the defaultDocument Section

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var isEnabled = configSection.Properties.Item( "enabled" ).Value;

```

4.7 Get a Section: Get the anonymousAuthentication Section

Get the anonymousAuthentication section:

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection("system.webServer/security/authentication/anonymousAuthentication", "MACHINE/WEBROOT/APPHOST" );

```

To get a child element:

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var filesElement = configSection.GetElementByName( "files" );

```

4.8 List the Entries of a Collection

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var collection = configSection.GetElementByName( "files" ).Collection;
for ( var i = 0; i < collection.Count; i++ )
{
    WScript.Echo( "file name = " + collection.Item( i ).Properties.Item( "value" ).Value );
}

```

4.9 Remove an Entry of a Collection

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";

var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var collection = configSection.GetElementByName( "files" ).Collection;
if ( collection.Count > 0 )
{
    collection.DeleteElement( 0 );
}
adminManager.CommitChanges();

```

4.10 Edit the Configuration of APPHOST in a Location Tag

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";

var configSection = adminManager.GetAdminSection(
    "system.webServer/urlCompression",
    "MACHINE/WEBROOT/APPHOST/Default Web Site" );
configSection.Properties.Item( "doDynamicCompression" ).Value = true;
adminManager.CommitChanges();

```

4.11 Read Schema Information: Determine If IsMergeAppend Is Set in the defaultDocuments Section

```

var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.AdminManager" );
var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var isMergeAppend = configSection.GetElementByName( "files" ).Collection.Schema.IsMergeAppend;

```

4.12 Get a Section for Write: Get the defaultDocument Section and Toggle the Enabled Attribute

```
var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";

var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
configSection.Properties.Item( "enabled" ).Value = true;
adminManager.CommitChanges();
```

4.13 Write into a Collection: Clear the Contents of the defaultDocument Section for Site1

```
var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";

var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
configSection.GetElementByName( "files" ).Collection.Clear();
adminManager.CommitChanges();
```

4.14 Write into a Collection: Add an Entry for the defaultDocument Section for Site1 as a Location Tag

```
var adminManager = WScript.CreateObject( "Microsoft.ApplicationHost.WritableAdminManager" );
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";

var configSection = adminManager.GetAdminSection( "system.webServer/defaultDocument", "MACHINE/WEBROOT/APPHOST" );
var collection = configSection.GetElementByName( "files" ).Collection;

var newElement = collection.CreateNewElement( "add" );
newElement.Properties.Item( "value" ).Value = "newdefdoc.htm";
collection.AddElement( newElement );
adminManager.CommitChanges();
```

5 Security

The following sections specify security considerations for implementers of the Internet Information Services (IIS) Application Host COM Protocol.

5.1 Security Considerations for Implementers

Implementers MAY enforce security as specified in [\[C706\]](#).

Implementers SHOULD review the security considerations as specified in [\[MS-RPCE\]](#).

5.2 Index of Security Parameters

None.

6 Appendix A: Full IDL

For ease of implementation, the full **IDL** is provided where "ms-dcom.idl" is the IDL specified in [\[MS-DCOM\]](#), Appendix [A](#) and "ms-oaut.idl" is the IDL that is specified in [\[MS-OAUT\]](#) Appendix [A](#).

```
import "ms-dcom.idl";
import "ms-oaut.idl";

#define string

interface IAppHostElementCollection;

interface IAppHostElement;

interface IAppHostConfigManager;

interface IAppHostAdminManager;

interface IAppHostPathMapper;

interface IAppHostChangeHandler;

interface IAppHostConfigFile;

interface IAppHostConfigLocation;

interface IAppHostProperty;

interface IAppHostPropertyCollection;

interface IAppHostElementSchema;

interface IAppHostPropertySchema;

interface IAppHostCollectionSchema;

interface IAppHostPropertySchemaCollection;

interface IAppHostElementSchemaCollection;

interface IAppHostSectionGroup;

interface IAppHostSectionDefinitionCollection;

interface IAppHostSectionDefinition;

interface IAppHostMethodCollection;

interface IAppHostMethodInstance;

interface IAppHostMethod;

interface IAppHostMethodExtension;

interface IAppHostConstantValue;
```



```

interface IAppHostConstantValueCollection;

#define SAFEARRAY(x) SAFEARRAY

[
    object,
    uuid( 70184ac9-7673-4770-96b1-445ce035cf70 ),
    pointer_default( unique ),
    helpstring( "IAppHostMethodExtension" )
]

interface IAppHostMethodExtension : IUnknown
{
    HRESULT
    ProvideMethod(
        [in] IAppHostMethod * pMethod,
        [in] IAppHostMethodInstance * pMethodInstance,
        [in] IAppHostElement * pElement
    );
};

[
    object,
    uuid( 88e220f3-33e8-4534-afac-b4a98eccf9ae ),
    pointer_default( unique ),
    helpstring( "IAppHostPropertyExtension" )
]

interface IAppHostPropertyExtension : IUnknown
{
    HRESULT
    ProvideGetProperty(
        [in] IAppHostElement * pElement,
        [in] IAppHostProperty * pProperty
    );
};

```

```

);

};

[
object,
uuid( 518eb37d-1ff4-42dd-86c3-3140bc35b823 ),
pointer_default( unique ),
helpstring( "IAppHostElementExtension" )
]

interface IAppHostElementExtension : IUnknown
{
HRESULT
ProvideElement(
[in] IAppHostElement * pNewElement
);
};

[
object,
uuid( 31a83ea0-c0e4-4a2c-8a01-353cc2a4c60a ),
pointer_default( unique ),
helpstring( "IAppHostMappingExtension" )
]

interface IAppHostMappingExtension : IUnknown
{
HRESULT
GetSiteNameFromSiteId(
[in] DWORD dwSiteId,
[out, retval, string] BSTR * pbstrSiteName
);
HRESULT

```

```

GetSiteIdFromSiteName(
    [in, string] BSTR bstrSiteName,
    [out, retval] DWORD * pdwSiteId
);

HRESULT
GetSiteElementFromSiteId(
    [in] DWORD dwSiteId,
    [out, retval] IAppHostElement ** ppSiteElement
);

HRESULT
MapPath(
    [in, string] BSTR bstrSiteName,
    [in, string] BSTR bstrVirtualPath,
    [out, string] BSTR * pbstrPhysicalPath,
    [out] IAppHostElement ** ppVirtualDirectoryElement,
    [out] IAppHostElement ** ppApplicationElement
);
};

[
    object,
    uuid( 08a90f5f-0702-48d6-b45f-02a9885a9768 ),
    pointer_default( unique ),
    helpstring( "IAppHostChildElementCollection" )
]

interface IAppHostChildElementCollection : IUnknown
{
    [propget]
    HRESULT
    Count(

```

```

[out, retval] DWORD * pcCount
);

[propget, id(DISPID_VALUE)]
HRESULT
Item(
[in] VARIANT cIndex,
[out, retval] IAppHostElement ** ppElement
);
}

[
object,
uuid( 0191775e-bcff-445a-b4f4-3bdda54e2816 ),
pointer_default( unique ),
helpstring( "IAppHostPropertyCollection" )
]

interface IAppHostPropertyCollection : IUnknown
{
[propget]
HRESULT
Count(
[out, retval] DWORD * pcCount
);
[propget, id(DISPID_VALUE)]
HRESULT
Item(
[in] VARIANT cIndex,
[out, retval] IAppHostProperty ** ppProperty
);
}

```

```

[
    object,
    uuid( 832a32f7-b3ea-4b8c-b260-9a2923001184 ),
    pointer_default( unique ),
    helpstring( "IAppHostConfigLocationCollection" )
]

interface IAppHostConfigLocationCollection : IUnknown
{
    [propget]
    HRESULT
    Count(
        [out, retval] DWORD * pcCount
    );
    [propget, id(DISPID_VALUE)]
    HRESULT
    Item(
        [in] VARIANT varIndex,
        [out, retval] IAppHostConfigLocation ** ppLocation
    );
    HRESULT
    AddLocation(
        [in, string] BSTR bstrLocationPath,
        [out, retval] IAppHostConfigLocation ** ppNewLocation
    );
    HRESULT
    DeleteLocation(
        [in] VARIANT cIndex
    );
}

```

```

[
    object,
    uuid( d6c7cd8f-bb8d-4f96-b591-d3a5f1320269 ),
    pointer_default( unique ),
    helpstring( "IAppHostMethodCollection" )
]

interface IAppHostMethodCollection : IUnknown
{
    [propget]
    HRESULT
    Count(
        [out, retval] DWORD * pcCount
    );
    [propget, id(DISPID_VALUE)]
    HRESULT
    Item(
        [in] VARIANT cIndex,
        [out, retval] IAppHostMethod ** ppMethod
    );
}

[
    object,
    uuid( 0344cdda-151e-4cbf-82da-66ae61e97754 ),
    pointer_default( unique ),
    helpstring( "IAppHostElementSchemaCollection" )
]

interface IAppHostElementSchemaCollection : IUnknown
{
    [propget]

```

```

HRESULT

Count(

[out, retval] DWORD * pcCount

);

[propget, id(DISPID_VALUE)]

HRESULT

Item(

[in] VARIANT cIndex,

[out, retval] IAppHostElementSchema ** ppElementSchema

);

}

[

object,

uuid( 8bed2c68-a5fb-4b28-8581-a0dc5267419f ),

pointer_default( unique ),

helpstring( "IAppHostPropertySchemaCollection" )

]

interface IAppHostPropertySchemaCollection : IUnknown

{

[propget]

HRESULT

Count(

[out, retval] DWORD * pcCount

);

[propget, id(DISPID_VALUE)]

HRESULT

Item(

[in] VARIANT cIndex,

[out, retval] IAppHostPropertySchema ** ppPropertySchema

```

```

);

}

[
object,

uuid( 5b5a68e6-8b9f-45e1-8199-a95ffccdffff ),

pointer_default( unique ),

helpstring( "IAppHostConstantValueCollection" )

]

interface IAppHostConstantValueCollection : IUnknown
{
[propget]
HRESULT
Count(
[out, retval] DWORD * pcCount
);

[propget, id(DISPID_VALUE)]
HRESULT
Item(
[in] VARIANT cIndex,
[out, retval] IAppHostConstantValue ** ppConstantValue
);
}

[
object,

uuid( 0716caf8-7d05-4a46-8099-77594be91394 ),

pointer_default( unique ),

helpstring( "IAppHostConstantValue" )

]

interface IAppHostConstantValue : IUnknown

```



```

{
    [propget]
    HRESULT
    Name(
        [out, retval] BSTR * pbstrName
    );
    [propget]
    HRESULT
    Value(
        [out, retval] DWORD * pdwValue
    );
}

[
    object,
    uuid( 450386db-7409-4667-935e-384dbbee2a9e ),
    pointer_default( unique ),
    helpstring( "IAppHostPropertySchema" )
]

interface IAppHostPropertySchema : IUnknown
{
    [propget]
    HRESULT
    Name(
        [out, string, retval] BSTR * pbstrName
    );
    [propget]
    HRESULT
    Type(
        [out, string, retval] BSTR * pbstrType

```

```

);

[propget]
HRESULT
DefaultValue(
[out, retval] VARIANT * pDefaultValue
);

[propget]
HRESULT
IsRequired(
[out, retval] VARIANT_BOOL * pfIsRequired
);

[propget]
HRESULT
IsUniqueKey(
[out, retval] VARIANT_BOOL * pfIsUniqueKey
);

[propget]
HRESULT
IsCombinedKey(
[out, retval] VARIANT_BOOL * pfIsCombinedKey
);

[propget]
HRESULT
IsExpanded(
[out, retval] VARIANT_BOOL * pfIsExpanded
);

[propget]
HRESULT
ValidationType(

```

```

[out, retval, string] BSTR * pbstrValidationType
);

[propget]
HRESULT
ValidationParameter(
[out, retval, string] BSTR * pbstrValidationParameter
);

HRESULT
GetMetadata(
[in, string] BSTR bstrMetadataType,
[out, retval] VARIANT * pValue
);

[propget]
HRESULT
IsCaseSensitive(
[out, retval] VARIANT_BOOL * pfIsCaseSensitive
);

[propget]
HRESULT
PossibleValues(
[out, retval] IAppHostConstantValueCollection ** ppValues
);

[propget]
HRESULT
DoesAllowInfinite(
[out, retval] VARIANT_BOOL * pfAllowInfinite
);

[propget]
HRESULT

```

```

IsEncrypted(
    [out, retval] VARIANT_BOOL * pfIsEncrypted
);

[propget]
HRESULT
TimeSpanFormat(
    [out, retval] BSTR * pbstrTimeSpanFormat
);
}

[
    object,
    uuid( de095db1-5368-4d11-81f6-efef619b7bcf ),
    pointer_default( unique ),
    helpstring( "IAppHostCollectionSchema" )
]

interface IAppHostCollectionSchema : IUnknown
{
    [propget]
    HRESULT
    AddElementNames(
        [out, string, retval] BSTR * pbstrElementName
    );
    HRESULT
    GetAddElementSchema(
        [in, string] BSTR bstrElementName,
        [out, retval] IAppHostElementSchema ** ppSchema
    );
    [propget]
    HRESULT

```

```

RemoveElementSchema(

[out, retval] IAppHostElementSchema ** ppSchema

);

[propget]

HRESULT

ClearElementSchema(

[out, retval] IAppHostElementSchema ** ppSchema

);

[propget]

HRESULT

IsMergeAppend(

[out, retval] VARIANT_BOOL * pfIsMergeAppend

);

HRESULT

GetMetadata(

[in, string] BSTR bstrMetadataType,

[out, retval] VARIANT * pValue

);

[propget]

HRESULT

DoesAllowDuplicates(

[out, retval] VARIANT_BOOL * pfAllowDuplicates

);

}

[

object,

uuid( ef13d885-642c-4709-99ec-b89561c6bc69 ),

pointer_default( unique ),

helpstring( "IAppHostElementSchema" )

```

```

]

interface IAppHostElementSchema : IUnknown
{
    [propget]
    HRESULT
    Name(
        [out, string, retval] BSTR * pbstrName
    );

    [propget]
    HRESULT
    DoesAllowUnschemasProperties(
        [out, retval] VARIANT_BOOL * pfAllowUnschemas
    );

    HRESULT
    GetMetadata(
        [in, string] BSTR bstrMetadataType,
        [out, retval] VARIANT * pValue
    );

    [propget]
    HRESULT
    CollectionSchema(
        [out, retval] IAppHostCollectionSchema ** ppCollectionSchema
    );

    [propget]
    HRESULT
    ChildElementSchemas(
        [out, retval] IAppHostElementSchemaCollection ** ppChildSchemas
    );

    [propget]

```

```

HRESULT

PropertySchemas(

[out, retval] IAppHostPropertySchemaCollection ** ppPropertySchemas

);

[propget]

HRESULT

IsCollectionDefault(

[out, retval] VARIANT_BOOL * pfIsCollectionDefault

);

}

[

object,

uuid( 2d9915fb-9d42-4328-b782-1b46819fab9e ),

pointer_default( unique ),

helpstring( "IAppHostMethodSchema" )

]

interface IAppHostMethodSchema : IUnknown

{

[propget]

HRESULT

Name(

[out, string, retval] BSTR * pbstrName

);

[propget]

HRESULT

InputSchema(

[out, retval] IAppHostElementSchema ** ppInputSchema

);

[propget]

```

```

HRESULT

OutputSchema(

[out, retval] IAppHostElementSchema ** ppOutputSchema

);

HRESULT

GetMetadata(

[in, string] BSTR bstrMetadataType,

[out, retval] VARIANT * pValue

);

}

[

object,

uuid( b80f3c42-60e0-4ae0-9007-f52852d3dbed ),

pointer_default( unique ),

helpstring( "IAppHostMethodInstance" )

]

interface IAppHostMethodInstance : IUnknown

{

[propget]

HRESULT

Input(

[out, retval] IAppHostElement ** ppInputElement

);

[propget]

HRESULT

Output(

[out, retval] IAppHostElement ** ppOutputElement

);

HRESULT

```



```

Execute(

);

HRESULT

GetMetadata(

[in, string] BSTR bstrMetadataType,

[out, retval] VARIANT * pValue

);

HRESULT

SetMetadata(

[in, string] BSTR bstrMetadataType,

[in] VARIANT value

);

}

[

object,

uuid( 7883ca1c-1112-4447-84c3-52fbeb38069d ),

pointer_default( unique ),

helpstring( "IAppHostMethod" )

]

interface IAppHostMethod : IUnknown

{

[propget]

HRESULT

Name(

[out, string, retval] BSTR * pbstrName

);

[propget]

HRESULT

Schema(

```

```

[out, retval] IAppHostMethodSchema ** ppMethodSchema
);

HRESULT
CreateInstance(
[out, retval] IAppHostMethodInstance ** ppMethodInstance
);
}

[
object,
uuid( 4dfa1df3-8900-4bc7-bbb5-d1a458c52410 ),
pointer_default( unique ),
helpstring( "IAppHostConfigException" )
]

interface IAppHostConfigException : IUnknown
{
[propget]
HRESULT
LineNumber(
[out, retval] unsigned long * pcLineNumber
);
[propget]
HRESULT
FileName(
[out, retval] BSTR * pbstrFileName
);
[propget]
HRESULT
ConfigPath(
[out, retval] BSTR * pbstrConfigPath

```

```

);

[propget]
HRESULT
ErrorLine(
[out, retval] BSTR * pbstrErrorLine
);

[propget]
HRESULT
PreErrorLine(
[out, retval] BSTR * pbstrPreErrorLine
);

[propget]
HRESULT
PostErrorLine(
[out, retval] BSTR * pbstrPostErrorLine
);

[propget]
HRESULT
ErrorString(
[out, retval] BSTR * pbstrErrorString
);
}

[
object,
uuid( eafe4895-a929-41ea-b14d-613e23f62b71 ),
pointer_default( unique ),
helpstring( "IAppHostPropertyException" )
]

interface IAppHostPropertyException : IAppHostConfigException

```

```

{

[propget]

HRESULT

InvalidValue(

[out, string, retval] BSTR * pbstrValue

);

[propget]

HRESULT

ValidationFailureReason(

[out, string, retval] BSTR * pbstrValidationReason

);

[propget]

HRESULT

ValidationFailureParameters(

[out, retval] SAFEARRAY(VARIANT) * pParameterArray

);

}

[

object,

uuid( c8550bff-5281-4b1e-ac34-99b6fa38464d ),

pointer_default( unique ),

helpstring( "IAppHostElementCollection" )

]

interface IAppHostElementCollection : IUnknown

{

[propget]

HRESULT

Count(

[out, retval] DWORD * pcElementCount


```

```

);

[propget, id(DISPID_VALUE)]

HRESULT

Item(

[in] VARIANT cIndex,

[out, retval] IAppHostElement ** ppElement

);

HRESULT

AddElement(

[in] IAppHostElement * pElement,

[in, defaultvalue(-1)] int cPosition

);

HRESULT

DeleteElement(

[in] VARIANT cIndex

);

HRESULT

Clear(

);

HRESULT

CreateNewElement(

[in, string, defaultvalue("")] BSTR bstrElementName,

[out, retval] IAppHostElement ** ppElement

);

[propget]

HRESULT

Schema(

[out, retval] IAppHostCollectionSchema** ppSchema

);

```

```

}

[
    object,
    uuid( 64ff8ccc-b287-4dae-b08a-a72cbf45f453 ),
    pointer_default( unique ),
    helpstring( "IAppHostElement" )
]

interface IAppHostElement : IUnknown
{
    [propget]
    HRESULT
    Name(
        [out, retval] BSTR * pbstrName
    );

    [propget]
    HRESULT
    Collection(
        [out, retval] IAppHostElementCollection ** ppCollection
    );

    [propget]
    HRESULT
    Properties(
        [out, retval] IAppHostPropertyCollection ** ppProperties
    );

    [propget]
    HRESULT
    ChildElements(
        [out, retval] IAppHostChildElementCollection ** ppElements
    );
}

```

```

HRESULT

GetMetadata(

[in, string] BSTR bstrMetadataType,

[out, retval] VARIANT * pValue

);

HRESULT

SetMetadata(

[in, string] BSTR bstrMetadataType,

[in] VARIANT value

);

[propget]

HRESULT

Schema(

[out, retval] IAppHostElementSchema ** ppSchema

);

HRESULT

GetElementByName(

[in, string] BSTR bstrSubName,

[out, retval] IAppHostElement ** ppElement

);

HRESULT

GetPropertyByName(

[in, string] BSTR bstrSubName,

[out, retval] IAppHostProperty ** ppProperty

);

HRESULT

Clear(

);

[propget]

```

```

HRESULT

Methods(

[out, retval] IAppHostMethodCollection ** ppMethods

);

}

[

object,

uuid( ed35f7a1-5024-4e7b-a44d-07ddaf4b524d ),

pointer_default( unique ),

helpstring( "IAppHostProperty" )

]

interface IAppHostProperty : IUnknown

{

[propget]

HRESULT

Name(

[out, retval] BSTR * pbstrName

);

[propget]

HRESULT

Value(

[out, retval] VARIANT * pVariant

);

[propput]

HRESULT

Value(

[in] VARIANT value

);

HRESULT

```



```

Clear(
);

[propget]
HRESULT
StringValue(
[out, retval] BSTR * pbstrValue
);

[propget]
HRESULT
Exception(
[out, retval] IAppHostPropertyException ** ppException
);

HRESULT
GetMetadata(
[in, string] BSTR bstrMetadataType,
[out, retval] VARIANT * pValue
);

HRESULT
SetMetadata(
[in, string] BSTR bstrMetadataType,
[in] VARIANT value
);

[propget]
HRESULT
Schema(
[out, retval] IAppHostPropertySchema ** ppSchema
);
}

[

```

```

object,

uuid( 370af178-7758-4dad-8146-7391f6e18585 ),

pointer_default( unique ),

helpstring( "IAppHostConfigLocation" )

]

interface IAppHostConfigLocation : IUnknown

{

    [propget]

    HRESULT

    Path(

        [out, string, retval] BSTR * pbstrLocationPath

    );

    [propget]

    HRESULT

    Count(

        [out, retval] DWORD * pcCount

    );

    [propget, id(DISPID_VALUE)]

    HRESULT

    Item(

        [in] VARIANT cIndex,

        [out, retval] IAppHostElement ** ppSection

    );

    HRESULT

    AddConfigSection(

        [in] BSTR bstrSectionName,

        [out, retval] IAppHostElement ** ppAdminElement

    );

    HRESULT

```

```

DeleteConfigSection(
    [in] VARIANT cIndex
);
}

[
    object,
    uuid( c5c04795-321c-4014-8fd6-d44658799393 ),
    pointer_default( unique ),
    helpstring( "IAppHostSectionDefinition" )
]
interface IAppHostSectionDefinition : IUnknown
{
    [propget]
    HRESULT
    Name(
        [out, retval] BSTR * pbstrName
    );
    [propget]
    HRESULT
    Type(
        [out, retval] BSTR * pbstrType
    );
    [propput]
    HRESULT
    Type(
        [in] BSTR bstrType
    );
    [propget]
    HRESULT

```

```

OverrideModeDefault(
    [out, retval] BSTR * pbstrOverrideModeDefault
);

[propget]
HRESULT
OverrideModeDefault(
    [in] BSTR bstrOverrideModeDefault
);

[propget]
HRESULT
AllowDefinition(
    [out, retval] BSTR * pbstrAllowDefinition
);

[propget]
HRESULT
AllowDefinition(
    [in] BSTR bstrAllowDefinition
);

[propget]
HRESULT
AllowLocation(
    [out, retval] BSTR * pbstrAllowLocation
);

[propget]
HRESULT
AllowLocation(
    [in] BSTR bstrAllowLocation
);
}

```

```

[
    object,
    uuid( b7d381ee-8860-47a1-8af4-1f33b2b1f325 ),
    pointer_default( unique ),
    helpstring( "IAppHostSectionDefinitionCollection" )
]

interface IAppHostSectionDefinitionCollection : IUnknown
{
    [propget]
    HRESULT
    Count(
        [out, retval] unsigned long * pcCount
    );
    [propget, id(DISPID_VALUE)]
    HRESULT
    Item(
        [in] VARIANT varIndex,
        [out, retval] IAppHostSectionDefinition ** ppConfigSection
    );
    HRESULT
    AddSection(
        [in, string] BSTR bstrSectionName,
        [out, retval] IAppHostSectionDefinition ** ppConfigSection
    );
    HRESULT
    DeleteSection(
        [in] VARIANT varIndex
    );
}

```

```

[
    object,
    uuid( 0dd8a158-ebe6-4008-ald9-b7ecc8f1104b ),
    pointer_default( unique ),
    helpstring( "IAppHostSectionGroup" )
]

interface IAppHostSectionGroup : IUnknown
{
    [propget]
    HRESULT
    Count(
        [out, retval] unsigned long * pcSectionGroup
    );
    [propget, id(DISPID_VALUE)]
    HRESULT
    Item(
        [in] VARIANT varIndex,
        [out, retval] IAppHostSectionGroup ** ppSectionGroup
    );
    [propget]
    HRESULT
    Sections(
        [out, retval] IAppHostSectionDefinitionCollection ** ppSections
    );
    HRESULT
    AddSectionGroup(
        [in, string] BSTR bstrSectionGroupName,
        [out, retval] IAppHostSectionGroup ** ppSectionGroup
    );

```

```

HRESULT

DeleteSectionGroup(

[in] VARIANT varIndex

);

[propget]

HRESULT

Name(

[out, retval] BSTR * pbstrName

);

[propget]

HRESULT

Type(

[out, retval] BSTR * pbstrType

);

[propput]

HRESULT

Type(

[in] BSTR bstrType

);

}

[

object,

uuid( ada4e6fb-e025-401e-a5d0-c3134a281f07 ),

pointer_default( unique ),

helpstring( "IAppHostConfigFile" )

]

interface IAppHostConfigFile : IUnknown

{

[propget]

```

```

HRESULT

ConfigPath(

[out, string, retval] BSTR * pbstrConfigPath

);

[propget]

HRESULT

FilePath(

[out, string, retval] BSTR * pbstrFilePath

);

[propget]

HRESULT

Locations(

[out, retval] IAppHostConfigLocationCollection ** ppLocations

);

HRESULT

GetAdminSection(

[in, string] BSTR bstrSectionName,

[in, string] BSTR bstrPath,

[out, retval] IAppHostElement ** ppAdminSection

);

HRESULT

GetMetadata(

[in, string] BSTR bstrMetadataType,

[out, retval] VARIANT * pValue

);

HRESULT

SetMetadata(

[in, string] BSTR bstrMetadataType,

[in] VARIANT value

```



```

);

HRESULT

ClearInvalidSections(

);

[propget]

HRESULT

RootSectionGroup(

[out, retval] IAppHostSectionGroup ** ppSectionGroups

);

}

[

object,

uuid( e7927575-5cc3-403b-822e-328a6b904bee ),

pointer_default( unique ),

helpstring( "IAppHostPathMapper" )

]

interface IAppHostPathMapper : IUnknown

{

HRESULT

MapPath(

[in, string] BSTR bstrConfigPath,

[in, string] BSTR bstrMappedPhysicalPath,

[out, string, retval] BSTR * pbstrNewPhysicalPath

);

}

[

object,

uuid( 09829352-87c2-418d-8d79-4133969a489d ),

pointer_default( unique ),

```

```

helpstring( "IAppHostChangeHandler" )

]

interface IAppHostChangeHandler : IUnknown
{
    HRESULT
    OnSectionChanges(
        [in, string] BSTR bstrSectionName,
        [in, string] BSTR bstrConfigPath
    );
}

[
    object,
    uuid( 9be77978-73ed-4a9a-87fd-13f09fec1b13 ),
    pointer_default( unique ),
    helpstring( "IAppHostAdminManager Interface" )
]

interface IAppHostAdminManager : IUnknown
{
    HRESULT
    GetAdminSection(
        [in, string] BSTR bstrSectionName,
        [in, string] BSTR bstrPath,
        [out, retval] IAppHostElement ** ppAdminSection
    );
    HRESULT
    GetMetadata(
        [in, string] BSTR bstrMetadataType,
        [out, retval] VARIANT * pValue
    );
}

```

```

HRESULT

SetMetadata(

[in, string] BSTR bstrMetadataType,

[in] VARIANT value

);

[propget]

HRESULT

ConfigManager(

[out, retval] IAppHostConfigManager ** ppConfigManager

);

}

[

object,

uuid( fa7660f6-7b3f-4237-a8bf-ed0ad0dcbbd9 ),

pointer_default( unique ),

helpstring( "IAppHostWritableAdminManager Interface" )

]

interface IAppHostWritableAdminManager : IAppHostAdminManager

{

HRESULT

CommitChanges(

);

[propget]

HRESULT

CommitPath(

[out, retval, string] BSTR * pbstrCommitPath

);

[propput]

HRESULT

```

```

CommitPath(

[in, string] BSTR bstrCommitPath

);

}

[

object,

uuid( 8f6d760f-f0cb-4d69-b5f6-848b33e9bdc6 ),

pointer_default( unique ),

helpstring( "IAppHostConfigManager Interface" )

]

interface IAppHostConfigManager : IUnknown

{

HRESULT

GetConfigFile(

[in, string] BSTR bstrConfigPath,

[out, retval] IAppHostConfigFile ** ppConfigFile

);

HRESULT

GetUniqueConfigPath(

[in, string] BSTR bstrConfigPath,

[out, retval, string] BSTR * pbstrUniquePath

);

}

```

7 Appendix B: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Vista SP1
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

8 Index

A

[Abstract data model - server](#)
[AddConfigSection method](#)
[AddElement method](#)
[AddElementNames method](#)
[AddLocation method](#)
[AddSection method](#)
[AddSectionGroup method](#)
[AllowDefinition method](#) ([section 3.1.4.27.6](#), [section 3.1.4.27.7](#))
[AllowLocation method](#) ([section 3.1.4.27.8](#), [section 3.1.4.27.9](#))
[Applicability](#)

C

[Capability negotiation](#)
[ChildElements method](#)
[ChildElementSchemas method](#)
[Clear method](#) ([section 3.1.4.12.10](#), [section 3.1.4.13.5](#), [section 3.1.4.22.4](#))
[ClearElementSchema method](#)
[ClearInvalidSections method](#)
[Collection method](#)
[CollectionSchema method](#)
[CommitChanges method](#)
[CommitPath method](#) ([section 3.1.4.30.2](#), [section 3.1.4.30.3](#))
[Common data types](#)
[ConfigManager method](#)
[ConfigPath method](#) ([section 3.1.4.5.3](#), [section 3.1.4.6.1](#))
[Count method](#) ([section 3.1.4.3.1](#), [section 3.1.4.7.2](#), [section 3.1.4.8.1](#), [section 3.1.4.11.1](#), [section 3.1.4.13.1](#), [section 3.1.4.15.1](#), [section 3.1.4.18.1](#), [section 3.1.4.23.1](#), [section 3.1.4.26.1](#), [section 3.1.4.28.1](#), [section 3.1.4.29.1](#))
[CreateInstance method](#)
[CreateNewElement method](#)

D

[Data model - abstract - server](#)
[Data types](#)
[DefaultValue method](#)
[DeleteConfigSection method](#)
[DeleteElement method](#)
[DeleteLocation method](#)
[DeleteSection method](#)
[DeleteSectionGroup method](#)
[DoesAllowDuplicates method](#)
[DoesAllowInfinite method](#)
[DoesAllowUnschemasizedProperties method](#)

E

[ErrorLine method](#)
[ErrorString method](#)

[Examples - overview](#)
[Exception method](#)
[Execute method](#)

F

[Fields - vendor-extensible](#)
[FileName method](#)
[FilePath method](#)

G

[GetAddElementSchema method](#)
[GetAdminSection method](#) ([section 3.1.4.1.1](#), [section 3.1.4.6.4](#))
[GetConfigFile method](#)
[GetElementByName method](#)
[GetMetadata method](#) ([section 3.1.4.1.2](#), [section 3.1.4.4.6](#), [section 3.1.4.6.5](#), [section 3.1.4.12.5](#), [section 3.1.4.14.3](#), [section 3.1.4.19.4](#), [section 3.1.4.20.4](#), [section 3.1.4.22.7](#), [section 3.1.4.25.10](#))
[GetPropertyByName method](#)
[GetSiteElementFromSiteId method](#)
[GetSiteIdFromSiteName method](#)
[GetSiteNameFromSiteId method](#)
[GetUniqueConfigPath method](#)
[Glossary](#)

I

[Implementer - security considerations](#)
[Index of security parameters](#)
[Informative references](#)
[Initialization - server](#)
[Input method](#)
[InputSchema method](#)
[Introduction](#)
[InvalidValue method](#)
[IsCaseSensitive method](#)
[IsCollectionDefault method](#)
[IsCombinedKey method](#)
[IsEncrypted method](#)
[IsExpanded method](#)
[IsMergeAppend method](#)
[IsRequired method](#)
[IsUniqueKey method](#)
[Item method](#) ([section 3.1.4.3.2](#), [section 3.1.4.7.3](#), [section 3.1.4.8.2](#), [section 3.1.4.11.2](#), [section 3.1.4.13.2](#), [section 3.1.4.15.2](#), [section 3.1.4.18.2](#), [section 3.1.4.23.2](#), [section 3.1.4.26.2](#), [section 3.1.4.28.2](#), [section 3.1.4.29.2](#))

L

[LineNumber method](#)
[Local events - server](#)
[Locations method](#)

M

MapPath method ([section 3.1.4.16.4](#), [section 3.1.4.21.1](#), [section 3.1.4.34.1](#))
[Message processing - server](#)

Messages
 [data types](#)
 [overview](#)
 [transport](#)
[Methods method](#)

N

Name method ([section 3.1.4.10.1](#), [section 3.1.4.12.1](#), [section 3.1.4.14.1](#), [section 3.1.4.17.1](#), [section 3.1.4.20.1](#), [section 3.1.4.22.1](#), [section 3.1.4.25.1](#), [section 3.1.4.27.1](#), [section 3.1.4.29.6](#))
[Normative references](#)

O

[OnSectionChanges method](#)
[Output method](#)
[OutputSchema method](#)
OverrideModeDefault method ([section 3.1.4.27.4](#), [section 3.1.4.27.5](#))
[Overview \(synopsis\)](#)

P

[Parameters - security index](#)
[Path method](#)
[PossibleValues method](#)
[PostErrorLine method](#)
[Preconditions](#)
[PreErrorLine method](#)
[Prerequisites](#)
[Properties method](#)
[PropertySchemas method](#)
[ProvideElement method](#)
[ProvideGetProperty method](#)
[ProvideMethod method](#)

R

References
 [informative](#)
 [normative](#)
 [overview](#)
[Relationship to other protocols](#)
[RemoveElementSchema method](#)
[RootSectionGroup method](#)

S

Schema method ([section 3.1.4.12.7](#), [section 3.1.4.13.7](#), [section 3.1.4.17.2](#), [section 3.1.4.22.9](#))
[Sections method](#)
Security
 [implementer considerations](#)
 [overview](#)

[parameter index](#)
[Sequencing rules - server](#)

Server

[abstract data model](#)
 [initialization](#)
 [local events](#)
 [message processing](#)
 [overview](#)
 [sequencing rules](#)
 [timer events](#)
 [timers](#)

SetMetadata method ([section 3.1.4.1.3](#), [section 3.1.4.6.6](#), [section 3.1.4.12.6](#), [section 3.1.4.19.5](#), [section 3.1.4.22.8](#))
[Standards assignments](#)
[StringValue method](#)

T

[Timer events - server](#)
[Timers - server](#)
[TimeSpanFormat method](#)
[Transport](#)
Type method ([section 3.1.4.25.2](#), [section 3.1.4.27.2](#), [section 3.1.4.27.3](#), [section 3.1.4.29.7](#), [section 3.1.4.29.8](#))

V

[ValidationFailureParameters method](#)
[ValidationFailureReason method](#)
[ValidationParameter method](#)
[ValidationType method](#)
Value method ([section 3.1.4.10.2](#), [section 3.1.4.22.2](#), [section 3.1.4.22.3](#))
[Vendor-extensible fields](#)
[Versioning](#)

W

[Windows behavior](#)