**[MS-OAUT]:**

**OLE Automation Protocol**

Intellectual Property Rights Notice for Open Specifications Documentation

* **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
* **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
* **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
* **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](http://go.microsoft.com/fwlink/?LinkId=214448). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
* **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
* **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights**. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

**Tools**. The Open Specifications documentation 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. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 12/18/2006 | 0.1 | New | Version 0.1 release |
| 3/2/2007 | 1.0 | Major | Version 1.0 release |
| 4/3/2007 | 1.1 | Minor | Version 1.1 release |
| 5/11/2007 | 1.2 | Minor | Version 1.2 release |
| 6/1/2007 | 1.2.1 | Editorial | Changed language and formatting in the technical content. |
| 7/20/2007 | 2.0 | Major | Updated and revised the technical content. |
| 8/10/2007 | 2.0.1 | Editorial | Changed language and formatting in the technical content. |
| 9/28/2007 | 2.1 | Minor | Clarified the meaning of the technical content. |
| 10/23/2007 | 3.0 | Major | Updated and revised the technical content. |
| 11/30/2007 | 4.0 | Major | Updated and revised the technical content. |
| 1/25/2008 | 5.0 | Major | Updated and revised the technical content. |
| 3/14/2008 | 6.0 | Major | Updated and revised the technical content. |
| 5/16/2008 | 7.0 | Major | Updated and revised the technical content. |
| 6/20/2008 | 8.0 | Major | Updated and revised the technical content. |
| 7/25/2008 | 8.1 | Minor | Clarified the meaning of the technical content. |
| 8/29/2008 | 8.1.1 | Editorial | Changed language and formatting in the technical content. |
| 10/24/2008 | 8.2 | Minor | Clarified the meaning of the technical content. |
| 12/5/2008 | 8.3 | Minor | Added Windows 7 Applicability. |
| 1/16/2009 | 8.3.1 | Editorial | Changed language and formatting in the technical content. |
| 2/27/2009 | 8.4 | Minor | Clarified the meaning of the technical content. |
| 4/10/2009 | 8.4.1 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 8.4.2 | Editorial | Changed language and formatting in the technical content. |
| 7/2/2009 | 9.0 | Major | Updated and revised the technical content. |
| 8/14/2009 | 9.0.1 | Editorial | Changed language and formatting in the technical content. |
| 9/25/2009 | 9.1 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 10.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 11.0 | Major | Updated and revised the technical content. |
| 1/29/2010 | 11.0.1 | Editorial | Changed language and formatting in the technical content. |
| 3/12/2010 | 12.0 | Major | Updated and revised the technical content. |
| 4/23/2010 | 12.0.1 | Editorial | Changed language and formatting in the technical content. |
| 6/4/2010 | 12.0.2 | Editorial | Changed language and formatting in the technical content. |
| 7/16/2010 | 12.0.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/27/2010 | 12.0.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2010 | 12.0.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010 | 12.0.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/7/2011 | 13.0 | Major | Updated and revised the technical content. |
| 2/11/2011 | 13.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/25/2011 | 13.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/6/2011 | 13.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/17/2011 | 13.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 13.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 14.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 14.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/12/2012 | 14.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/25/2012 | 14.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/31/2013 | 14.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 15.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 15.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 15.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 15.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 16.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 16.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 16.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/1/2017 | 16.0 | None | No changes to the meaning, language, or formatting of the technical content. |

Table of Contents

[1 Introduction 11](#_Toc483456357)

[1.1 Glossary 11](#_Toc483456358)

[1.2 References 14](#_Toc483456359)

[1.2.1 Normative References 14](#_Toc483456360)

[1.2.2 Informative References 14](#_Toc483456361)

[1.3 Overview 15](#_Toc483456362)

[1.4 Relationship to Other Protocols 18](#_Toc483456363)

[1.5 Prerequisites/Preconditions 18](#_Toc483456364)

[1.6 Applicability Statement 18](#_Toc483456365)

[1.7 Versioning and Capability Negotiation 18](#_Toc483456366)

[1.8 Vendor-Extensible Fields 18](#_Toc483456367)

[1.9 Standards Assignments 18](#_Toc483456368)

[2 Messages 20](#_Toc483456369)

[2.1 Transport 20](#_Toc483456370)

[2.2 Common Data Types 20](#_Toc483456371)

[2.2.1 BYTE 20](#_Toc483456372)

[2.2.2 IID 20](#_Toc483456373)

[2.2.3 LPOLESTR 20](#_Toc483456374)

[2.2.4 REFIID 21](#_Toc483456375)

[2.2.5 REFGUID 21](#_Toc483456376)

[2.2.6 PSAFEARRAY, LPSAFEARRAY 21](#_Toc483456377)

[2.2.7 VARIANT Type Constants 21](#_Toc483456378)

[2.2.8 SAFEARRAY Feature Constants 26](#_Toc483456379)

[2.2.9 ADVFEATUREFLAGS Advanced Feature Flags 28](#_Toc483456380)

[2.2.10 CALLCONV Calling Convention Constants 29](#_Toc483456381)

[2.2.11 FUNCFLAGS Function Feature Constants 29](#_Toc483456382)

[2.2.12 FUNCKIND Function Access Constants 30](#_Toc483456383)

[2.2.13 IMPLTYPEFLAGS Feature Constants 30](#_Toc483456384)

[2.2.14 INVOKEKIND Function Invocation Constants 31](#_Toc483456385)

[2.2.15 PARAMFLAGS Parameter Feature Constants 32](#_Toc483456386)

[2.2.16 TYPEFLAGS Type Feature Constants 32](#_Toc483456387)

[2.2.17 TYPEKIND Type Kind Constants 33](#_Toc483456388)

[2.2.18 VARFLAGS Variable Feature Constants 34](#_Toc483456389)

[2.2.19 VARKIND Variable Kind Constants 35](#_Toc483456390)

[2.2.20 LIBFLAGS Type Library Feature Constants 36](#_Toc483456391)

[2.2.21 SYSKIND System Pointer Size Constants 36](#_Toc483456392)

[2.2.22 DESCKIND Name Description Constants 36](#_Toc483456393)

[2.2.23 BSTR 37](#_Toc483456394)

[2.2.23.1 FLAGGED\_WORD\_BLOB 37](#_Toc483456395)

[2.2.23.2 BSTR Type Definition 38](#_Toc483456396)

[2.2.23.3 Mapping Between Presented and Transmitted BSTRs 38](#_Toc483456397)

[2.2.24 CURRENCY 38](#_Toc483456398)

[2.2.25 DATE 38](#_Toc483456399)

[2.2.26 DECIMAL 39](#_Toc483456400)

[2.2.27 VARIANT\_BOOL 39](#_Toc483456401)

[2.2.28 User-Defined Data Types and BRECORD 40](#_Toc483456402)

[2.2.28.1 User-Defined Data Types 40](#_Toc483456403)

[2.2.28.2 BRECORD 40](#_Toc483456404)

[2.2.28.2.1 \_wireBRECORD 40](#_Toc483456405)

[2.2.28.2.2 BRECORD 41](#_Toc483456406)

[2.2.29 VARIANT 41](#_Toc483456407)

[2.2.29.1 \_wireVARIANT 41](#_Toc483456408)

[2.2.29.2 VARIANT 43](#_Toc483456409)

[2.2.30 SAFEARRAY 43](#_Toc483456410)

[2.2.30.1 SAFEARRAYBOUND 43](#_Toc483456411)

[2.2.30.2 SAFEARR\_BSTR 44](#_Toc483456412)

[2.2.30.3 SAFEARR\_UNKNOWN 44](#_Toc483456413)

[2.2.30.4 SAFEARR\_DISPATCH 44](#_Toc483456414)

[2.2.30.5 SAFEARR\_VARIANT 45](#_Toc483456415)

[2.2.30.6 SAFEARR\_BRECORD 45](#_Toc483456416)

[2.2.30.7 SAFEARR\_HAVEIID 45](#_Toc483456417)

[2.2.30.8 Scalar-Sized Arrays 46](#_Toc483456418)

[2.2.30.8.1 BYTE\_SIZEDARR 46](#_Toc483456419)

[2.2.30.8.2 WORD\_SIZEDARR 46](#_Toc483456420)

[2.2.30.8.3 DWORD\_SIZEDARR 46](#_Toc483456421)

[2.2.30.8.4 HYPER\_SIZEDARR 47](#_Toc483456422)

[2.2.30.9 SAFEARRAYUNION 47](#_Toc483456423)

[2.2.30.10 SAFEARRAY 47](#_Toc483456424)

[2.2.31 RecordInfoData 49](#_Toc483456425)

[2.2.32 DISPID 50](#_Toc483456426)

[2.2.32.1 Reserved DISPIDs 50](#_Toc483456427)

[2.2.33 DISPPARAMS 51](#_Toc483456428)

[2.2.34 EXCEPINFO 51](#_Toc483456429)

[2.2.35 MEMBERID 52](#_Toc483456430)

[2.2.35.1 Reserved MEMBERIDs 52](#_Toc483456431)

[2.2.36 HREFTYPE 52](#_Toc483456432)

[2.2.37 TYPEDESC 53](#_Toc483456433)

[2.2.38 ARRAYDESC 53](#_Toc483456434)

[2.2.39 PARAMDESCEX 54](#_Toc483456435)

[2.2.40 PARAMDESC 54](#_Toc483456436)

[2.2.41 ELEMDESC 54](#_Toc483456437)

[2.2.42 FUNCDESC 54](#_Toc483456438)

[2.2.43 VARDESC 55](#_Toc483456439)

[2.2.44 TYPEATTR 56](#_Toc483456440)

[2.2.45 TLIBATTR 58](#_Toc483456441)

[2.2.46 CUSTDATAITEM 59](#_Toc483456442)

[2.2.47 CUSTDATA 59](#_Toc483456443)

[2.2.48 SCODE 60](#_Toc483456444)

[2.2.49 IDL Syntax Extensions 60](#_Toc483456445)

[2.2.49.1 COM Server Categories 63](#_Toc483456446)

[2.2.49.1.1 Aggregatable Servers 63](#_Toc483456447)

[2.2.49.1.2 Connectable Servers 63](#_Toc483456448)

[2.2.49.1.3 Bindable Servers 63](#_Toc483456449)

[2.2.49.2 IDL Automation Scope 64](#_Toc483456450)

[2.2.49.3 Automation-Compatible Types 66](#_Toc483456451)

[2.2.49.4 Automation Interfaces 68](#_Toc483456452)

[2.2.49.4.1 Automation-Compatible Interfaces 68](#_Toc483456453)

[2.2.49.4.2 Dual Interfaces 69](#_Toc483456454)

[2.2.49.4.3 Dispinterface Interfaces 69](#_Toc483456455)

[2.2.49.5 Automation Members 69](#_Toc483456456)

[2.2.49.5.1 Interfaces Automation Members 69](#_Toc483456457)

[2.2.49.5.2 Bindable Properties 70](#_Toc483456458)

[2.2.49.5.3 Dispinterfaces Automation Members 71](#_Toc483456459)

[2.2.49.6 Automation Parameters 72](#_Toc483456460)

[2.2.49.7 AIDL Interfaces and ODL Dispinterfaces 72](#_Toc483456461)

[2.2.49.7.1 Property Equivalence 73](#_Toc483456462)

[2.2.49.7.2 Method Equivalence 73](#_Toc483456463)

[2.2.49.8 Coclass Specifications 73](#_Toc483456464)

[2.2.49.9 Module Specifications 75](#_Toc483456465)

[2.2.49.10 Referencing External Types 76](#_Toc483456466)

[2.2.50 String Handling 76](#_Toc483456467)

[2.2.50.1 String Equivalence 76](#_Toc483456468)

[2.2.50.2 Globalization 76](#_Toc483456469)

[2.2.51 Automation Hash Values 77](#_Toc483456470)

[2.2.51.1 ComputeHash Method 77](#_Toc483456471)

[2.2.51.2 ComputeHashDBCS Method 78](#_Toc483456472)

[2.2.51.3 MapDBChar Method 81](#_Toc483456473)

[2.2.51.4 Locale Names 81](#_Toc483456474)

[2.2.51.5 Primary Lookup Tables 81](#_Toc483456475)

[2.2.51.6 DBCS Substitution Tables 85](#_Toc483456476)

[3 Protocol Details 88](#_Toc483456477)

[3.1 Automation Server Details 88](#_Toc483456478)

[3.1.1 Abstract Data Model 88](#_Toc483456479)

[3.1.2 Timers 88](#_Toc483456480)

[3.1.3 Initialization 88](#_Toc483456481)

[3.1.4 Message Processing Events and Sequencing Rules 89](#_Toc483456482)

[3.1.4.1 IDispatch::GetTypeInfoCount (Opnum 3) 89](#_Toc483456483)

[3.1.4.2 IDispatch::GetTypeInfo (Opnum 4) 89](#_Toc483456484)

[3.1.4.3 IDispatch::GetIDsOfNames (Opnum 5) 90](#_Toc483456485)

[3.1.4.4 IDispatch::Invoke (Opnum 6) 91](#_Toc483456486)

[3.1.4.4.1 Invoke Consistency Checks 93](#_Toc483456487)

[3.1.4.4.2 Invoke Argument-Parameter Mapping 94](#_Toc483456488)

[3.1.4.4.3 Handling Default Value and Optional Arguments 94](#_Toc483456489)

[3.1.4.4.4 Argument Coercion 94](#_Toc483456490)

[3.1.5 Timer Events 94](#_Toc483456491)

[3.1.6 Other Local Events 94](#_Toc483456492)

[3.2 Automation Client Details 95](#_Toc483456493)

[3.2.1 Abstract Data Model 95](#_Toc483456494)

[3.2.2 Timers 95](#_Toc483456495)

[3.2.3 Initialization 95](#_Toc483456496)

[3.2.4 Message Processing and Sequencing Rules 95](#_Toc483456497)

[3.2.5 Timer Events 95](#_Toc483456498)

[3.2.6 Other Local Events 96](#_Toc483456499)

[3.3 IEnumVARIANT Server Details 96](#_Toc483456500)

[3.3.1 Abstract Data Model 96](#_Toc483456501)

[3.3.2 Timers 96](#_Toc483456502)

[3.3.3 Initialization 96](#_Toc483456503)

[3.3.4 Message Processing and Sequencing Rules 96](#_Toc483456504)

[3.3.4.1 IEnumVARIANT::Next (Opnum 3) 97](#_Toc483456505)

[3.3.4.2 IEnumVARIANT::Skip (Opnum 4) 98](#_Toc483456506)

[3.3.4.3 IEnumVARIANT::Reset (Opnum 5) 98](#_Toc483456507)

[3.3.4.4 IEnumVARIANT::Clone (Opnum 6) 98](#_Toc483456508)

[3.3.5 Timer Events 99](#_Toc483456509)

[3.3.6 Other Local Events 99](#_Toc483456510)

[3.4 IEnumVARIANT Client Details 99](#_Toc483456511)

[3.4.1 Abstract Data Model 99](#_Toc483456512)

[3.4.2 Timers 99](#_Toc483456513)

[3.4.3 Initialization 99](#_Toc483456514)

[3.4.4 Message Processing and Sequencing Rules 99](#_Toc483456515)

[3.4.5 Timer Events 100](#_Toc483456516)

[3.4.6 Other Local Events 100](#_Toc483456517)

[3.5 ITypeComp Server Details 100](#_Toc483456518)

[3.5.1 Abstract Data Model 100](#_Toc483456519)

[3.5.2 Timers 100](#_Toc483456520)

[3.5.3 Initialization 100](#_Toc483456521)

[3.5.4 Message Processing Events and Sequencing Rules 100](#_Toc483456522)

[3.5.4.1 ITypeComp::Bind (Opnum 3) 101](#_Toc483456523)

[3.5.4.1.1 Binding Context 102](#_Toc483456524)

[3.5.4.1.1.1 Automation Type Library Binding Context 102](#_Toc483456525)

[3.5.4.1.1.2 Automation Type Description Binding Context 103](#_Toc483456526)

[3.5.4.1.2 Types Returned with Bound Elements 103](#_Toc483456527)

[3.5.4.1.2.1 Types Returned with ITypeLib Members 103](#_Toc483456528)

[3.5.4.1.2.2 Types Returned with ITypeInfo Members 104](#_Toc483456529)

[3.5.4.2 ITypeComp::BindType (Opnum 4) 104](#_Toc483456530)

[3.5.5 Timer Events 105](#_Toc483456531)

[3.5.6 Other Local Events 105](#_Toc483456532)

[3.6 ITypeComp Client Details 105](#_Toc483456533)

[3.6.1 Abstract Data Model 105](#_Toc483456534)

[3.6.2 Timers 105](#_Toc483456535)

[3.6.3 Initialization 105](#_Toc483456536)

[3.6.4 Message Processing Events and Sequencing Rules 105](#_Toc483456537)

[3.6.5 Timer Events 105](#_Toc483456538)

[3.6.6 Other Local Events 105](#_Toc483456539)

[3.7 ITypeInfo Server Details 105](#_Toc483456540)

[3.7.1 Abstract Data Model 105](#_Toc483456541)

[3.7.1.1 Common Automation Type Description Elements 106](#_Toc483456542)

[3.7.1.2 TYPEKIND Dependent Automation Type Description Elements 107](#_Toc483456543)

[3.7.2 Timers 108](#_Toc483456544)

[3.7.3 Initialization 108](#_Toc483456545)

[3.7.4 Message Processing Events and Sequencing Rules 108](#_Toc483456546)

[3.7.4.1 ITypeInfo::GetTypeAttr (Opnum 3) 110](#_Toc483456547)

[3.7.4.2 ITypeInfo::GetTypeComp (Opnum 4) 110](#_Toc483456548)

[3.7.4.3 ITypeInfo::GetFuncDesc (Opnum 5) 110](#_Toc483456549)

[3.7.4.4 ITypeInfo::GetVarDesc (Opnum 6) 111](#_Toc483456550)

[3.7.4.5 ITypeInfo::GetNames (Opnum 7) 112](#_Toc483456551)

[3.7.4.6 ITypeInfo::GetRefTypeOfImplType (Opnum 8) 113](#_Toc483456552)

[3.7.4.7 ITypeInfo::GetImplTypeFlags (Opnum 9) 114](#_Toc483456553)

[3.7.4.8 ITypeInfo::GetDocumentation (Opnum 12) 114](#_Toc483456554)

[3.7.4.9 ITypeInfo::GetDllEntry (Opnum 13) 115](#_Toc483456555)

[3.7.4.10 ITypeInfo::GetRefTypeInfo (Opnum 14) 117](#_Toc483456556)

[3.7.4.11 ITypeInfo::CreateInstance (Opnum 16) 117](#_Toc483456557)

[3.7.4.12 ITypeInfo::GetMops (Opnum 17) 118](#_Toc483456558)

[3.7.4.13 ITypeInfo::GetContainingTypeLib (Opnum 18) 118](#_Toc483456559)

[3.7.5 Timer Events 119](#_Toc483456560)

[3.7.6 Other Local Events 119](#_Toc483456561)

[3.8 ITypeInfo Client Details 119](#_Toc483456562)

[3.8.1 Abstract Data Model 119](#_Toc483456563)

[3.8.2 Timers 119](#_Toc483456564)

[3.8.3 Initialization 119](#_Toc483456565)

[3.8.4 Message Processing Events and Sequencing Rules 119](#_Toc483456566)

[3.8.5 Timer Events 119](#_Toc483456567)

[3.8.6 Other Local Events 120](#_Toc483456568)

[3.9 ITypeInfo2 Server Details 120](#_Toc483456569)

[3.9.1 Abstract Data Model 120](#_Toc483456570)

[3.9.2 Timers 120](#_Toc483456571)

[3.9.3 Initialization 120](#_Toc483456572)

[3.9.4 Message Processing Events and Sequencing Rules 120](#_Toc483456573)

[3.9.4.1 ITypeInfo2::GetTypeKind (Opnum 22) 122](#_Toc483456574)

[3.9.4.2 ITypeInfo2::GetTypeFlags (Opnum 23) 122](#_Toc483456575)

[3.9.4.3 ITypeInfo2::GetFuncIndexOfMemId (Opnum 24) 122](#_Toc483456576)

[3.9.4.4 ITypeInfo2::GetVarIndexOfMemId (Opnum 25) 123](#_Toc483456577)

[3.9.4.5 ITypeInfo2::GetCustData (Opnum 26) 124](#_Toc483456578)

[3.9.4.6 ITypeInfo2::GetFuncCustData (Opnum 27) 124](#_Toc483456579)

[3.9.4.7 ITypeInfo2::GetParamCustData (Opnum 28) 125](#_Toc483456580)

[3.9.4.8 ITypeInfo2::GetVarCustData (Opnum 29) 126](#_Toc483456581)

[3.9.4.9 ITypeInfo2::GetImplTypeCustData (Opnum 30) 126](#_Toc483456582)

[3.9.4.10 ITypeInfo2::GetDocumentation2 (Opnum 31) 127](#_Toc483456583)

[3.9.4.11 ITypeInfo2::GetAllCustData (Opnum 32) 128](#_Toc483456584)

[3.9.4.12 ITypeInfo2::GetAllFuncCustData (Opnum 33) 128](#_Toc483456585)

[3.9.4.13 ITypeInfo2::GetAllParamCustData (Opnum 34) 129](#_Toc483456586)

[3.9.4.14 ITypeInfo2::GetAllVarCustData (Opnum 35) 130](#_Toc483456587)

[3.9.4.15 ITypeInfo2::GetAllImplTypeCustData (Opnum 36) 130](#_Toc483456588)

[3.9.5 Timer Events 131](#_Toc483456589)

[3.9.6 Other Local Events 131](#_Toc483456590)

[3.10 ITypeInfo2 Client Details 131](#_Toc483456591)

[3.10.1 Abstract Data Model 131](#_Toc483456592)

[3.10.2 Timers 131](#_Toc483456593)

[3.10.3 Initialization 131](#_Toc483456594)

[3.10.4 Message Processing Events and Sequencing Rules 131](#_Toc483456595)

[3.10.5 Timer Events 131](#_Toc483456596)

[3.10.6 Other Local Events 132](#_Toc483456597)

[3.11 ITypeLib Server Details 132](#_Toc483456598)

[3.11.1 Abstract Data Model 132](#_Toc483456599)

[3.11.2 Timers 133](#_Toc483456600)

[3.11.3 Initialization 133](#_Toc483456601)

[3.11.4 Message Processing Events and Sequencing Rules 133](#_Toc483456602)

[3.11.4.1 ITypeLib::GetTypeInfoCount (Opnum 3) 134](#_Toc483456603)

[3.11.4.2 ITypeLib::GetTypeInfo (Opnum 4) 135](#_Toc483456604)

[3.11.4.3 ITypeLib::GetTypeInfoType (Opnum 5) 135](#_Toc483456605)

[3.11.4.4 ITypeLib::GetTypeInfoOfGuid (Opnum 6) 136](#_Toc483456606)

[3.11.4.5 ITypeLib::GetLibAttr (Opnum 7) 136](#_Toc483456607)

[3.11.4.6 ITypeLib::GetTypeComp (Opnum 8) 137](#_Toc483456608)

[3.11.4.7 ITypeLib::GetDocumentation (Opnum 9) 137](#_Toc483456609)

[3.11.4.8 ITypeLib::IsName (Opnum 10) 138](#_Toc483456610)

[3.11.4.9 ITypeLib::FindName (Opnum 11) 139](#_Toc483456611)

[3.11.5 Timer Events 140](#_Toc483456612)

[3.11.6 Other Local Events 140](#_Toc483456613)

[3.12 ITypeLib Client Details 140](#_Toc483456614)

[3.12.1 Abstract Data Model 140](#_Toc483456615)

[3.12.2 Timers 140](#_Toc483456616)

[3.12.3 Initialization 140](#_Toc483456617)

[3.12.4 Message Processing Events and Sequencing Rules 140](#_Toc483456618)

[3.12.5 Timer Events 141](#_Toc483456619)

[3.12.6 Other Local Events 141](#_Toc483456620)

[3.13 ITypeLib2 Server Details 141](#_Toc483456621)

[3.13.1 Abstract Data Model 141](#_Toc483456622)

[3.13.2 Timers 141](#_Toc483456623)

[3.13.3 Initialization 141](#_Toc483456624)

[3.13.4 Message Processing Events and Sequencing Rules 141](#_Toc483456625)

[3.13.4.1 ITypeLib2::GetCustData (Opnum 13) 142](#_Toc483456626)

[3.13.4.2 ITypeLib2::GetLibStatistics (Opnum 14) 142](#_Toc483456627)

[3.13.4.3 ITypeLib2::GetDocumentation2 (Opnum 15) 143](#_Toc483456628)

[3.13.4.4 ITypeLib2::GetAllCustData (Opnum 16) 144](#_Toc483456629)

[3.13.5 Timer Events 144](#_Toc483456630)

[3.13.6 Other Local Events 144](#_Toc483456631)

[3.14 ITypeLib2 Client Details 144](#_Toc483456632)

[3.14.1 Abstract Data Model 144](#_Toc483456633)

[3.14.2 Timers 144](#_Toc483456634)

[3.14.3 Initialization 144](#_Toc483456635)

[3.14.4 Message Processing Events and Sequencing Rules 144](#_Toc483456636)

[3.14.5 Timer Events 145](#_Toc483456637)

[3.14.6 Other Local Events 145](#_Toc483456638)

[4 Protocol Examples 146](#_Toc483456639)

[4.1 AIDL-ODL Property Equivalence 146](#_Toc483456640)

[4.2 AIDL-ODL Method Equivalence 146](#_Toc483456641)

[4.3 Invoke Argument Parameter Mapping 146](#_Toc483456642)

[4.4 Getting the Value of a Property 147](#_Toc483456643)

[4.5 Setting the Value of a Property 148](#_Toc483456644)

[4.6 Calling a Method with Byref and Optional Arguments 149](#_Toc483456645)

[4.7 IEnumVARIANT Example 150](#_Toc483456646)

[4.7.1 IEnumVARIANT Next() Example 151](#_Toc483456647)

[4.7.2 IEnumVARIANT Skip() Example 152](#_Toc483456648)

[4.7.3 IEnumVARIANT Reset() Example 153](#_Toc483456649)

[4.7.4 IEnumVARIANT Clone() Example 153](#_Toc483456650)

[4.8 Reading Type Information 154](#_Toc483456651)

[4.8.1 Getting ITypeLib Implementations from Automation Server 154](#_Toc483456652)

[4.8.2 Enumerating on All Types in a Type Library 155](#_Toc483456653)

[4.8.3 Enumerating on All Enumerations in a Type Library 155](#_Toc483456654)

[4.8.4 Enumerating All Nonsource Interfaces in a Coclass 155](#_Toc483456655)

[4.8.5 Enumerating All Methods in an Interface 156](#_Toc483456656)

[4.8.6 Retrieving Type Information 157](#_Toc483456657)

[4.8.7 Binding to a Member of a Default Nonsource Interface of an Appobject Coclass 158](#_Toc483456658)

[4.8.8 Binding to a Member of a Partner Interface 158](#_Toc483456659)

[5 Security 160](#_Toc483456660)

[5.1 Security Considerations for Implementer 160](#_Toc483456661)

[5.2 Index of Security Parameters 160](#_Toc483456662)

[6 Appendix A: Full IDL 161](#_Toc483456663)

[7 Appendix B: Product Behavior 176](#_Toc483456664)

[8 Appendix C: Full ABNF 181](#_Toc483456665)

[9 Change Tracking 187](#_Toc483456666)

[10 Index 188](#_Toc483456667)

# Introduction

The OLE Automation Protocol uses [**Distributed Component Object Model (DCOM)**](#gt_ae2a9876-7fed-4f0d-a390-bf78f76c0736) as its transport layer. It provides support for an additional set of types, a late-bound calling mechanism, and type description and discovery. The late-bound calling mechanism is based on dispatch identifiers and a dispatching table that maps the identifiers to specific operations. The dispatch identifiers and the dispatching table are specified by using [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) extensions specified in this document. Type description and discovery are based on a set of IDL extensions and a set of interfaces that are implemented by type library and type description servers.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

## Glossary

This document uses the following terms:

**aggregatable server**: A [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that can be contained by another [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) and can allow its [**interfaces**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) to be used as if they were defined by the containing [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703).

**automation client**: An application that can manipulate objects exposed by other applications, which are also called [**automation servers**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

**automation interface**: An [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) that supports the OLE Automation Protocol.

**Automation Interface Definition Language (AIDL) interface**: An [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) that is not defined with the syntax of properties and methods.

**automation scope**: An [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) scope that provides a context for [**automation types**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e) that are defined or referenced.

**automation scope family**: A set of [**automation scopes**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) that share the same [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1).

**automation scope generation**: A set of [**automation scopes**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) that belong to the same automation family and share the same version.

**automation server**: An application that exposes its functionality through COM interfaces to other applications, which are also called [**automation clients**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28).

**automation type browser**: A COM [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) that uses [**automation type descriptions**](#gt_fb6a1829-c102-482c-902f-51c197b22860) to examine the functionality provided by an [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) or an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

**automation type description**: A [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that describes and provides access to the members of a type that is defined or referenced in an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a).

**automation type library**: A [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that provides descriptions of the automation-compatible types that are defined or referenced in an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a).

**automation types**: Types that support the OLE Automation Protocol.

**bindable server**: A [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) that is able to notify a [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) whenever the value of a specified [**property**](#gt_f930baab-25f1-4142-bced-5effc9f62d45) is changed.

**byref argument**: An argument to be modified by the invoked automation method. Such an argument is represented as a VARIANT with the VT\_BYREF flag set.

**class identifier (CLSID)**: A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that identifies a software component; for instance, a DCOM object class or a COM class.

**client**: An execution environment that holds object references and issues object RPC (ORPC) calls.

**coclass**: A component [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (an association between a [**class identifier (CLSID)**](#gt_e433c806-6cb6-46a2-bb95-523df8818c99) and a set of named implementations of IUnknown) that is defined using the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) keyword.

**COM server**: A [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) that provides access to a component [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (an association between a [**CLSID**](#gt_e433c806-6cb6-46a2-bb95-523df8818c99) and a set of named implementations of IUnknown).

**connectable server**: A [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) that uses specified source [**interfaces**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) to communicate with clients that implement those [**interfaces**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9).

**DCOM interface**: An ORPC [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9).

**dispatch ID (DISPID)**: A 32-bit signed integer used in [**automation interfaces**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) to identify methods, properties, and arguments.

**dispinterface**: An [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) defined by using the [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) keyword or as part of a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9).

**Distributed Component Object Model (DCOM)**: The Microsoft Component Object Model (COM) specification that defines how components communicate over networks, as specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0).

**dual interface**: An interface that can act either as a dispinterface or a Distributed Component Object Model (DCOM) interface.

**dynamic endpoint**: A network-specific server address that is requested and assigned at run time. For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824).

**endpoint**: A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence ncacn\_ip\_tcp), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence ncacn\_np), an endpoint might be the name of a named pipe. For more information, see [C706].

**globally unique identifier (GUID)**: A term used interchangeably with [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [[RFC4122]](https://go.microsoft.com/fwlink/?LinkId=90460) or [C706] must be used for generating the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). See also [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3).

**HRESULT**: An integer value that indicates the result or status of an operation. A particular HRESULT can have different meanings depending on the protocol using it. See [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1 and specific protocol documents for further details.

**interface**: A specification in a Component Object Model (COM) server that describes how to access the methods of a class. For more information, see [MS-DCOM].

**Interface Definition Language (IDL)**: The International Standards Organization (ISO) standard language for specifying the [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) for remote procedure calls. For more information, see [C706] section 4.

**interface identifier (IID)**: A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that identifies an [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9).

**language code identifier (LCID)**: A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

**Microsoft Interface Definition Language (MIDL)**: The Microsoft implementation and extension of the OSF-DCE [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824). [**MIDL**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) can also mean the [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) compiler provided by Microsoft. For more information, see [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

**named argument**: An argument specified in a call both by its value and by its [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474). [**Named arguments**](#gt_0d91f93b-e04c-47d2-ae86-35d7e1bf382a) always follow positional arguments.

**Network Data Representation (NDR)**: A specification that defines a mapping from [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) data types onto octet streams. [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) also refers to the runtime environment that implements the mapping facilities (for example, data provided to [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d)). For more information, see [MS-RPCE] and [C706] section 14.

**object**: In COM, a software entity that implements the IUnknown interface and zero or more additional interfaces that may be obtained from each other using the IUnknown interface. A COM [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) can be exposed to remote clients via the DCOM protocol, in which case it is also a [**DCOM**](#gt_ae2a9876-7fed-4f0d-a390-bf78f76c0736) object.

**OBJREF**: The marshaled form of an object reference.

**ODL dispinterface**: An Object Description Language (ODL) [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) defined using the syntax of properties and methods.

**opnum**: An operation number or numeric identifier that is used to identify a specific [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

**partner dispinterface**: An [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) that exposes the members of a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) as a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5).

**partner interface**: An [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) that exposes the members of a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) as a [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408).

**property**: A data field within a Common Information Model (CIM) class definition. This consists of a simple name, a type, and a value.

**reference dispinterface**: A [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) defined by referencing a [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408).

**remote procedure call (RPC)**: A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (\*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (\*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (\*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

**server**: A computer on which the [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) server is executing.

**source interface**: An [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) that is defined by a [**connectable server**](#gt_106828f1-1bf9-4cdb-9d12-d48b52495114), and implemented by a [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) to enable the [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) to initiate communication with the [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506).

**universally unique identifier (UUID)**: A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) objects. UUIDs are highly likely to be unique. UUIDs are also known as [**globally unique identifiers (GUIDs)**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

**user-defined type (UDT)**: A group of related data items that is declared as one type of information in an [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) file.

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](http://msdn.microsoft.com/en-us/library/dn781092.aspx).

### 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](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, [https://www2.opengroup.org/ogsys/catalog/c706](https://go.microsoft.com/fwlink/?LinkId=89824)

[MS-DCOM] Microsoft Corporation, "[Distributed Component Object Model (DCOM) Remote Protocol](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0)".

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2)".

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90)".

[MS-OAUT] Microsoft Corporation, "[OLE Automation Protocol](%5bMS-OAUT%5d.pdf#Section_bbb05720f72445c78d17f83c3d1a3961)".

[MS-RPCE] Microsoft Corporation, "[Remote Procedure Call Protocol Extensions](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)".

[MS-UCODEREF] Microsoft Corporation, "[Windows Protocols Unicode Reference](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, [http://www.rfc-editor.org/rfc/rfc2119.txt](https://go.microsoft.com/fwlink/?LinkId=90317)

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, [http://www.rfc-editor.org/rfc/rfc4234.txt](https://go.microsoft.com/fwlink/?LinkId=90462)

### Informative References

[MSDN-CALLCONV] Microsoft Corporation, "C++ Calling Conventions", [http://msdn.microsoft.com/en-us/library/k2b2ssfy(VS.80).aspx](https://go.microsoft.com/fwlink/?LinkId=94981)

[MSDN-COM] Microsoft Corporation, "Component Object Model", [http://msdn.microsoft.com/en-us/library/aa286559.aspx](https://go.microsoft.com/fwlink/?LinkId=89977)

[MSDN-ErrorHandling] Microsoft Corporation, "Win32 and Com Development Error Handling", [http://msdn.microsoft.com/en-us/library/ms679320.aspx](https://go.microsoft.com/fwlink/?LinkId=94931)

[MSDN-SafeArrayAccessData] Microsoft Corporation, "SafeArrayAccessData Function", [http://msdn.microsoft.com/en-us/library/ms221620.aspx](https://go.microsoft.com/fwlink/?LinkId=90119)

[MSDN-WinHelp] Microsoft Corporation, "WinHelp function", [http://msdn.microsoft.com/en-us/library/ms647953.aspx](https://go.microsoft.com/fwlink/?LinkId=90163)

[MSDN] Microsoft Corporation, "MSDN Home Page", [http://msdn.microsoft.com/en-us/default.aspx](https://go.microsoft.com/fwlink/?LinkId=124362)

[SCODE] Microsoft Corporation, "SCODE", [http://msdn.microsoft.com/en-us/library/ms527117.aspx](https://go.microsoft.com/fwlink/?LinkId=90511)

## Overview

The OLE Automation Protocol extends the use of [**DCOM**](#gt_ae2a9876-7fed-4f0d-a390-bf78f76c0736) by providing support for marshaling an additional set of types (known as [**automation types**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e)) and by providing support for exposing COM components to [**automation clients**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) (such as scripting engines) through a late-bound calling alternative to the traditional COM calls. Additionally, the OLE Automation Protocol specifies how a type browser can discover and interpret type information provided by a type description [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703).

The automation client and server can be present on the same machine, or on different machines connected by a network. Automation takes advantage of functionality provided by the Microsoft Component Object Model (for more information, see [[MSDN-COM]](https://go.microsoft.com/fwlink/?LinkId=89977)) and the Microsoft Distributed Component Object Model (as specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0)) for creating, activating, and managing the lifetime of the objects exposed by an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

To support late-bound calling, the OLE Automation Protocol specifies the following:

* How a server defines a set of automation methods that can be dispatched, based on a [**dispatch ID (DISPID)**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474).
* How the server provides a way to map a method name to the DISPID.
* How the server performs the late-bound call, based on the DISPID.

The automation methods are defined by using extensions to the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) language specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) sections 6, 7, 8, 9, 10, 11, 12, 13, and 14. These extensions provide the definition of [**automation interfaces**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) containing automation methods and properties. Each IDL definition of an automation method and [**property**](#gt_f930baab-25f1-4142-bced-5effc9f62d45) can have a unique (per [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9)) integer value associated with it. This value is defined as a DISPID and is statically discoverable (from the IDL specification of the method), and dynamically discoverable (through a call to [IDispatch::GetIDsOfNames (section 3.1.4.3)](#Section_7166d6ffb8514216bfaa34128274a242)). This value is then used by automation clients to invoke the automation method, or to set or retrieve an automation property (through a call to IDispatch::Invoke).

To support this late-bound calling mechanism, Automation defines a set of types, [VARIANT (section 2.2.29)](#Section_b2ee2b50665e43e6a92c8f2a29fd7add) being the most important of them. A VARIANT can be thought of as a discriminated union of all automation-supported types. The OLE Automation Protocol imposes the following restriction on the automation interfaces: All types of method arguments and properties can be stored as VARIANT structures.

The following illustration shows a generic automation call sequence:

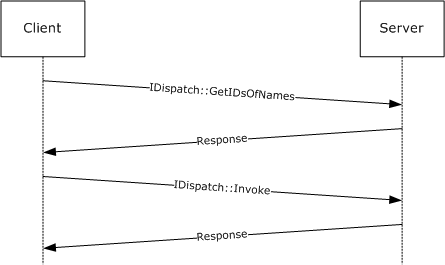


Figure 1: Generic Automation call

This automation call response can be optimized if the automation client knows the DISPIDs associated with an automation server's method at compile time, and forgoes the initial call to GetIDsOfNames:

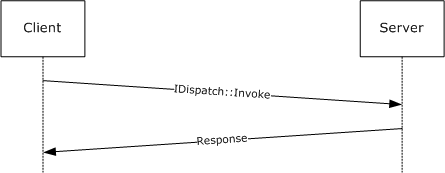


Figure 2: Optimized Automation call

To support type description and discovery, the OLE Automation Protocol specifies:

* How the automation server supports queries for type-description support.
* How to specify an extended set of capabilities and relationships using automation IDL extensions.
* How a server can provide access to the information specified in the IDL.

An automation server specifies that it provides type information by implementing [IDispatch::GetTypeInfoCount (section 3.1.4.1)](#Section_d3233e5b657f4c988a6156449c96fe16), and exposes access to the type description server that describes the server's [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) by implementing [IDispatch::GetTypeInfo (section 3.1.4.2)](#Section_d1791851649142898c5725967ef7b9ed).

The following diagram illustrates a generic query for type information.

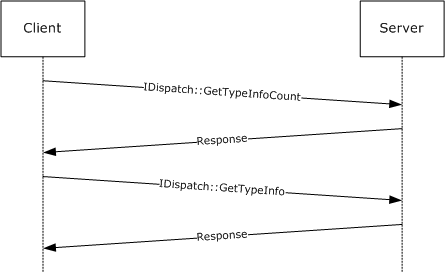


Figure 3: Generic query for type information

To support exposing type information related to [**COM servers**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c), the OLE Automation Protocol specifies the set of types that are used to encapsulate semantic information associated with a COM server, with the interfaces it implements, and with the methods and properties defined on those interfaces ([TYPEATTR (section 2.2.44)](#Section_0ca10d0861d2405991097bbaf545715e), [FUNCDESC (section 2.2.42)](#Section_d3349d25e11d4095ba86de3fda178c4e), and [VARDESC (section 2.2.43)](#Section_ae7791d243994dffb7c6b0d4f3dce982) being the most important), in addition to the set of interfaces that a server has to implement to provide COM clients with access to the type information (ITypeInfo being the central one).

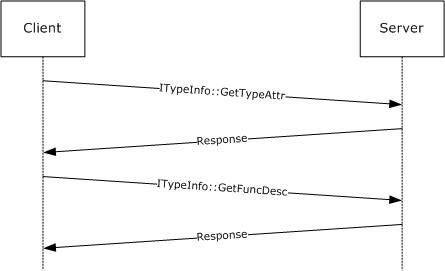


Figure 4: Generic query for type information related to COM servers

## Relationship to Other Protocols

The OLE Automation Protocol uses the DCOM Remote Protocol, as specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0).

## Prerequisites/Preconditions

The DCOM Remote Protocol, as specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0), must be installed on both the [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) and [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

## Applicability Statement

The OLE Automation Protocol is useful for exposing application functionality to scripting languages and across a distributed environment.

Exposing COM objects as [**automation servers**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) offers several benefits:

* Exposed objects from many applications are available in a single programming environment. Software implementers can choose from these objects to create solutions that span applications.
* Exposed objects are accessible from any scripting environment or programming tool that implements automation. Systems integrators are not limited to the programming language in which the objects were developed. Instead, they can choose the programming tool or scripting language that best suits their needs and capabilities.
* Object names can remain consistent across versions of an application and can conform automatically to the user's language.

## Versioning and Capability Negotiation

Supported Transports: This protocol uses the DCOM Remote Protocol, as specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0), as its transport.

Protocol Version: This protocol comprises two DCOM interfaces: IDispatch and IEnumVARIANT, which are both version 0.0.

## Vendor-Extensible Fields

This protocol uses HRESULTs, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). Vendors can choose their own values for this field as long as the C bit (0x20000000) is set, which indicates that it is a customer code.

This protocol uses Win32 error codes. These values are taken from the Windows error number space, as specified in [MS-ERREF]. Vendors SHOULD[<1>](#Appendix_A_1" \o "Product behavior note 1) reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision.

This protocol uses [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474), which are vendor-extensible. Vendors are free to choose their own values, as long as the method and [**property**](#gt_f930baab-25f1-4142-bced-5effc9f62d45) DISPIDs are strictly positive 32-bit values. For more information about DISPIDs, see section [2.2.32](#Section_b0b43e39b0804edda26d7134075c75cd).

## Standards Assignments

The following [**GUIDs**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) are used by the OLE Automation Protocol.

| Constant/value | Description |
| --- | --- |
| CLSID\_RecordInfo  {0000002F-0000-0000-C000-000000000046} | The OBJREF\_CUSTOM unmarshaler [**CLSID**](#gt_e433c806-6cb6-46a2-bb95-523df8818c99) for [RecordInfoData (section 2.2.31)](#Section_deb939dfef4d49c384677265669e89ed). |
| IID\_IRecordInfo  {0000002F-0000-0000-C000-000000000046} | The value of the **IID** field of the pRecInfo OBJREF structure (see section [2.2.28.2.1](#Section_d9237563093e4bc9b8244c306bfc19e3)). |
| IID\_IDispatch  {00020400-0000-0000-C000-000000000046} | The GUID associated with the IDispatch [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) (see section [3.1](#Section_c2c7dbe2bafa49da93a77b75499ef90a)). |
| IID\_ITypeComp  {00020403-0000-0000-C000-000000000046} | The GUID associated with the ITypeComp interface (see section [3.5](#Section_7894019fde1e455eb2aa3b899c2e50f6)). |
| IID\_ITypeInfo  {00020401-0000-0000-C000-000000000046} | The GUID associated with the ITypeInfo interface (see section [3.7](#Section_99504cf916d8401ea87383b85d1ee4aa)). |
| IID\_ITypeInfo2  {00020412-0000-0000-C000-000000000046} | The GUID associated with the ITypeInfo2 interface (see section [3.9](#Section_2d6024dad2294d78bbb0b9d5bf6459b7)). |
| IID\_ITypeLib  {00020402-0000-0000-C000-000000000046} | The GUID associated with the ITypeLib interface (see section [3.11](#Section_5daecf67bc6e4e17bcf8797bdba1748b)). |
| IID\_ITypeLib2  {00020411-0000-0000-C000-000000000046} | The GUID associated with the ITypeLib2 interface (see section [3.13](#Section_4bb9bc733cf540a185c7aafaff4874cc)). |
| IID\_IUnknown  {00000000-0000-0000-C000-000000000046} | The GUID associated with the IUnknown interface. |
| IID\_IEnumVARIANT  {00020404-0000-0000-C000-000000000046} | The GUID associated with the IEnumVARIANT interface (see section [3.3](#Section_716d04d1cd1640659b191b8808b3df31)). |
| IID\_NULL  {00000000-0000-0000-0000-000000000000} | The GUID that identifies a NULL value (as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section A1 nil [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3)). |

# Messages

The following sections specify how OLE Automation Protocol messages are encapsulated on the wire, common OLE Automation Protocol data types, and the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) extensions that support late-bound calling.

## Transport

This protocol uses [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) [**dynamic endpoints**](#gt_46da887f-3f66-4941-a854-e51c52cf4c56) (as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) sections 6, 7, 8, 9, 10, 11, 12, 13, and 14) and the DCOM Remote Protocol, as specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0).

To access an [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9), the [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) MUST request a DCOM connection to its well-known [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) on the [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703), as specified in section [1.9](#Section_58504586e4af44a3be04f1dc281b7429).

The RPC version number for all interfaces MUST be 0.0.

## Common Data Types

This protocol MUST indicate to the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime that it is to support the [**Network Data Representation (NDR)**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) transfer syntax only, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) sections 6, 7, 8, 9, 10, 11, 12, 13, and 14.

In addition to RPC base types and definitions specified in [C706] and [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2), additional data types are defined in the following sections.

### BYTE

A BYTE is an 8-bit, unsigned value that corresponds to a single octet in a network protocol.

This type is declared as follows:

1. typedef byte BYTE;

### IID

The IID type specifies an [**interface identifier (IID)**](#gt_76ad3105-3f05-479d-a40c-c9c8fa2ebd83).

A [**globally unique identifier (GUID)**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2), section 2.3.4.

This type is declared as follows:

1. typedef GUID IID;

### LPOLESTR

The following is the type definition for the LPOLESTR type.

This type is declared as follows:

1. typedef [string] wchar\_t\* LPOLESTR;

### REFIID

The following is the type definition for the REFIID type.

This type is declared as follows:

1. typedef IID\* REFIID;

### REFGUID

The following is the type definition for the REFGUID type.

This type is declared as follows:

1. typedef GUID\* REFGUID;

### PSAFEARRAY, LPSAFEARRAY

Definitions for the PSAFEARRAY and LPSAFEARRAY types follow:

This type is declared as follows:

1. typedef [unique] SAFEARRAY\* PSAFEARRAY, \*LPSAFEARRAY;

### VARIANT Type Constants

The VARENUM enumeration constants are used in the discriminant field, **vt**, of the VARIANT type specified in section [2.2.29.2](#Section_a6a540af38ac48bebd4092e2c01e9aa6). When present, the VT\_BYREF flag MUST be in an OR relation with another value to specify the [**byref argument**](#gt_969e6685-c90a-4da6-99be-1a25efb6d1cd) for the VARIANT. The VT\_EMPTY and VT\_NULL values MUST NOT be specified with the VT\_BYREF bit flag.

The following values are also used in the discriminant field, **vt**, of the **TYPEDESC** structure specified in section [2.2.37](#Section_95bb92a7f783477facbcc947d754fa8b).

The meaning of each VARIANT type constant is specified as follows. The Context column specifies the context in which each constant is used. A constant MUST NOT be used in a VARIANT unless it is specified with a "V". A constant MUST NOT be used in a SAFEARRAY unless it is specified with an "S". A constant MUST NOT be used in a TYPEDESC unless it is specified with a "T".

1. typedef enum tagVARENUM
2. {
3. VT\_EMPTY = 0x0000,
4. VT\_NULL = 0x0001,
5. VT\_I2 = 0x0002,
6. VT\_I4 = 0x0003,
7. VT\_R4 = 0x0004,
8. VT\_R8 = 0x0005,
9. VT\_CY = 0x0006,
10. VT\_DATE = 0x0007,
11. VT\_BSTR = 0x0008,
12. VT\_DISPATCH = 0x0009,
13. VT\_ERROR = 0x000A,
14. VT\_BOOL = 0x000B,
15. VT\_VARIANT = 0x000C,
16. VT\_UNKNOWN = 0x000D,
17. VT\_DECIMAL = 0x000E,
18. VT\_I1 = 0x0010,
19. VT\_UI1 = 0x0011,
20. VT\_UI2 = 0x0012,
21. VT\_UI4 = 0x0013,
22. VT\_I8 = 0x0014,
23. VT\_UI8 = 0x0015,
24. VT\_INT = 0x0016,
25. VT\_UINT = 0x0017,
26. VT\_VOID = 0x0018,
27. VT\_HRESULT = 0x0019,
28. VT\_PTR = 0x001A,
29. VT\_SAFEARRAY = 0x001B,
30. VT\_CARRAY = 0x001C,
31. VT\_USERDEFINED = 0x001D,
32. VT\_LPSTR = 0x001E,
33. VT\_LPWSTR = 0x001F,
34. VT\_RECORD = 0x0024,
35. VT\_INT\_PTR = 0x0025,
36. VT\_UINT\_PTR = 0x0026,
37. VT\_ARRAY = 0x2000,
38. VT\_BYREF = 0x4000
39. } VARENUM;

**VT\_EMPTY:**

| Context | Description |
| --- | --- |
| V | The type of the contained field is undefined. When this flag is specified, the VARIANT MUST NOT contain a data field. The VARIANT definition is specified in section 2.2.29.2. |

**VT\_NULL:**

| Context | Description |
| --- | --- |
| V | The type of the contained field is NULL. When this flag is specified, the VARIANT MUST NOT contain a data field. The VARIANT definition is specified in section 2.2.29.2. |

**VT\_I2:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 2-byte signed integer. |

**VT\_I4:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 4-byte signed integer. |

**VT\_R4:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 4-byte IEEE floating-point number. |

**VT\_R8:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be an 8-byte IEEE floating-point number. |

**VT\_CY:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be CURRENCY (see section [2.2.24](#Section_5a2b34c4d109438e9ec884816d8de40d)). |

**VT\_DATE:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be DATE (see section [2.2.25](#Section_35c9bf2db8e84d7da50f367da0d99fce)). |

**VT\_BSTR:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be BSTR (see section [2.2.23](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1)). |

**VT\_DISPATCH:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a pointer to IDispatch (see section [3.1.4](#Section_ac9c502bac1c42028ad4048ac98afcc9)). |

**VT\_ERROR:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be HRESULT. |

**VT\_BOOL:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be VARIANT\_BOOL (see section [2.2.27](#Section_7b39eb249d39498abcd875c38e5823d0)). |

**VT\_VARIANT:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be VARIANT (see section [2.2.29](#Section_b2ee2b50665e43e6a92c8f2a29fd7add)). It MUST appear with the bit flag VT\_BYREF. |

**VT\_UNKNOWN:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a pointer to IUnknown. |

**VT\_DECIMAL:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be DECIMAL (see section [2.2.26](#Section_b5493025e447410993a8ac29c48d018d)). |

**VT\_I1:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 1-byte integer. |

**VT\_UI1:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 1-byte unsigned integer. |

**VT\_UI2:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 2-byte unsigned integer. |

**VT\_UI4:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 4-byte unsigned integer. |

**VT\_I8:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be an 8-byte signed integer. |

**VT\_UI8:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be an 8-byte unsigned integer. |

**VT\_INT:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 4-byte signed integer. |

**VT\_UINT:**

| Context | Description |
| --- | --- |
| V, S, T | Either the specified type, or the type of the element or contained field MUST be a 4-byte unsigned integer. |

**VT\_VOID:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be void. |

**VT\_HRESULT:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be HRESULT. |

**VT\_PTR:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be a unique pointer, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 4.2.20.2. |

**VT\_SAFEARRAY:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be [SAFEARRAY (section 2.2.30)](#Section_04e72b3f573145089bb4de29fbd0f781). |

**VT\_CARRAY:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be a fixed-size array. |

**VT\_USERDEFINED:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be user defined. |

**VT\_LPSTR:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be a NULL-terminated string, as specified in [C706] section 14.3.4. |

**VT\_LPWSTR:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be a zero-terminated string of UNICODE characters, as specified in [C706], section 14.3.4. |

**VT\_RECORD:**

| Context | Description |
| --- | --- |
| V, S | The type of the element or contained field MUST be a BRECORD (see section [2.2.28.2](#Section_ea064b3d9fb3448699924fe2463e83e8)). |

**VT\_INT\_PTR:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be either a 4-byte or an 8-byte signed integer. The size of the integer is platform specific and determines the system pointer size value, as specified in section [2.2.21](#Section_0d81289ef0ef474d8e61dedae9ea5a08). |

**VT\_UINT\_PTR:**

| Context | Description |
| --- | --- |
| T | The specified type MUST be either a 4 byte or an 8 byte unsigned integer. The size of the integer is platform specific and determines the system pointer size value, as specified in section 2.2.21. |

**VT\_ARRAY:**

| Context | Description |
| --- | --- |
| V, S | The type of the element or contained field MUST be a SAFEARRAY (see section [2.2.30.10](#Section_2e87a537930541c6a88bb79809b3703a)). |

**VT\_BYREF:**

| Context | Description |
| --- | --- |
| V, S | The type of the element or contained field MUST be a pointer to one of the types listed in the previous rows of this table. If present, this bit flag MUST appear in a VARIANT discriminant (see section [2.2.28](#Section_29ce0a4f478649c9a3125522c1e9b44d)) with one of the previous flags. |

### SAFEARRAY Feature Constants

The SF\_TYPE enumeration values are used in the discriminant field, **sfType**, of a [SAFEARRAYUNION](#Section_5eaa490ff6c54d58b368cf2d0ea74572) structure.

The SAFEARRAY feature constants are defined in the SF\_TYPE enumeration.

1. typedef [v1\_enum] enum tagSF\_TYPE
2. {
3. SF\_ERROR = VT\_ERROR,
4. SF\_I1 = VT\_I1,
5. SF\_I2 = VT\_I2,
6. SF\_I4 = VT\_I4,
7. SF\_I8 = VT\_I8,
8. SF\_BSTR = VT\_BSTR,
9. SF\_UNKNOWN = VT\_UNKNOWN,
10. SF\_DISPATCH = VT\_DISPATCH,
11. SF\_VARIANT = VT\_VARIANT,
12. SF\_RECORD = VT\_RECORD,
13. SF\_HAVEIID = VT\_UNKNOWN | 0x8000
14. } SF\_TYPE;

**SF\_ERROR:** This value means that the SAFEARRAY was incorrectly marshaled. The receiver MUST reject any call that has a SAFEARRAY argument with this flag specified, by raising an RPC\_X\_BAD\_STUB\_DATA [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) exception.

Hex value is 0x0000000A.

Element marshaling size: N/A

**SF\_I1:** The type of the elements contained in the SAFEARRAY MUST be a 1-byte integer.

Hex value is 0x00000010.

Element marshaling size in bytes: 1

**SF\_I2:** The type of the elements contained in the SAFEARRAY MUST be a 2-byte integer.

Hex value is 0x00000002.

Element marshaling size in bytes: 2

**SF\_I4:** The type of the elements contained in the SAFEARRAY MUST be a 4-byte integer.

Hex value is 0x00000003.

Element marshaling size in bytes: 4

**SF\_I8:** The type of the elements contained in the SAFEARRAY MUST be an 8-byte integer.

Hex value is 0x00000014.

Element marshaling size in bytes: 8

**SF\_BSTR:** The type of the elements contained in the SAFEARRAY MUST be a BSTR.

Hex value is 0x00000008.

Element marshaling size in bytes: 4

**SF\_UNKNOWN:** The type of the elements contained in the SAFEARRAY MUST be a pointer to IUnknown.

Hex value is 0x0000000D.

Element marshaling size in bytes: 4

**SF\_DISPATCH:** The type of the elements contained in the SAFEARRAY MUST be a pointer to IDispatch (see section [3.1.4](#Section_ac9c502bac1c42028ad4048ac98afcc9)).

Hex value is 0x00000009.

Element marshaling size in bytes: 4

**SF\_VARIANT:** The type of the elements contained in the SAFEARRAY MUST be VARIANT.

Hex value is 0x0000000C.

Element marshaling size in bytes: 16

**SF\_RECORD:** The type of the elements contained in the SAFEARRAY is a [**user-defined type (UDT)**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727) (as defined in section [2.2.28.1](#Section_7b86dfb8ca9b437bad8abd9f0aadc266).

Hex value is 0x00000024.

Element marshaling size in bytes: 4

**SF\_HAVEIID:** The type of the elements contained in the SAFEARRAY MUST be an MInterfacePointer.

Hex value is 0x0000800D.

Element marshaling size in bytes: 4

### ADVFEATUREFLAGS Advanced Feature Flags

The following values are used in the field **fFeatures** of a [SAFEARRAY (section 2.2.30.10)](#Section_2e87a537930541c6a88bb79809b3703a) data type.

1. typedef enum tagADVFEATUREFLAGS
2. {
3. FADF\_AUTO = 0x0001,
4. FADF\_STATIC = 0x0002,
5. FADF\_EMBEDDED = 0x0004,
6. FADF\_FIXEDSIZE = 0x0010,
7. FADF\_RECORD = 0x0020,
8. FADF\_HAVEIID = 0x0040,
9. FADF\_HAVEVARTYPE = 0x0080,
10. FADF\_BSTR = 0x0100,
11. FADF\_UNKNOWN = 0x0200,
12. FADF\_DISPATCH = 0x0400,
13. FADF\_VARIANT = 0x0800
14. } ADVFEATUREFLAGS;

**FADF\_AUTO:** MUST be set if the SAFEARRAY is allocated on the stack. This flag MUST be ignored on receipt.

**FADF\_STATIC:** MUST be set if the SAFEARRAY is statically allocated. This flag MUST be ignored on receipt.

**FADF\_EMBEDDED:** MUST be set if the SAFEARRAY is embedded in a structure. This flag MUST be ignored on receipt.

**FADF\_FIXEDSIZE:** MUST be set if the SAFEARRAY cannot be resized or reallocated. This flag MUST be ignored on receipt.

**FADF\_RECORD:** The [SAFEARRAY](#Section_04e72b3f573145089bb4de29fbd0f781) MUST contain elements of a [**UDT**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727) (see section [2.2.28.1](#Section_7b86dfb8ca9b437bad8abd9f0aadc266))

**FADF\_HAVEIID:** The SAFEARRAY MUST contain MInterfacePointers elements.

**FADF\_HAVEVARTYPE:** If this bit flag is set, the high word of the **cLocks** field of the SAFEARRAY MUST contain a VARIANT type constant that describes the type of the array's elements (see sections [2.2.7](#Section_3fe7db9f58034dc49d145425d3f5461f) and 2.2.30.10).

**FADF\_BSTR:** The SAFEARRAY MUST contain an array of BSTR elements (see section [2.2.23](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1)).

**FADF\_UNKNOWN:** The SAFEARRAY MUST contain an array of pointers to IUnknown.

**FADF\_DISPATCH:** The SAFEARRAY MUST contain an array of pointers to IDispatch (see section [3.1.4](#Section_ac9c502bac1c42028ad4048ac98afcc9)).

**FADF\_VARIANT:** The SAFEARRAY MUST contain an array of VARIANT instances.

### CALLCONV Calling Convention Constants

The CALLCONV enumeration values are used in the **callconv** field of a FUNCDESC to identify the calling convention of a local method defined in the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) module, as specified in sections [2.2.42](#Section_d3349d25e11d4095ba86de3fda178c4e) and [2.2.49.9](#Section_82f9465bae46474e87ffe65e9751affb) .

The following calling convention constants are defined in the CALLCONV enumeration:

1. typedef [v1\_enum] enum tagCALLCONV
2. {
3. CC\_CDECL = 1,
4. CC\_PASCAL = 2,
5. CC\_STDCALL = 4
6. } CALLCONV;

**CC\_CDECL:** MUST be set if the method was declared with the cdecl keyword.

**CC\_PASCAL:** MUST be set if the method was declared with the pascal keyword.

**CC\_STDCALL:** MUST be set if the method was declared with the stdcall keyword.

### FUNCFLAGS Function Feature Constants

The FUNCFLAGS enumeration values are used in the **wFuncFlags** field of a FUNCDESC to identify features of a function, as specified in section [2.2.42](#Section_d3349d25e11d4095ba86de3fda178c4e).

The function feature constants are defined in the FUNCFLAGS enumeration.

1. typedef enum tagFUNCFLAGS
2. {
3. FUNCFLAG\_FRESTRICTED = 1,
4. FUNCFLAG\_FSOURCE = 0x2,
5. FUNCFLAG\_FBINDABLE = 0x4,
6. FUNCFLAG\_FREQUESTEDIT = 0x8,
7. FUNCFLAG\_FDISPLAYBIND = 0x10,
8. FUNCFLAG\_FDEFAULTBIND = 0x20,
9. FUNCFLAG\_FHIDDEN = 0x40,
10. FUNCFLAG\_FUSESGETLASTERROR = 0x80,
11. FUNCFLAG\_FDEFAULTCOLLELEM = 0x100,
12. FUNCFLAG\_FUIDEFAULT = 0x200,
13. FUNCFLAG\_FNONBROWSABLE = 0x400,
14. FUNCFLAG\_FREPLACEABLE = 0x800,
15. FUNCFLAG\_FIMMEDIATEBIND = 0x1000
16. } FUNCFLAGS;

**FUNCFLAG\_FRESTRICTED:** MUST be set if the method or [**property**](#gt_f930baab-25f1-4142-bced-5effc9f62d45) was declared with the **[restricted]** attribute (as specified in section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f)).

**FUNCFLAG\_FSOURCE:** MUST be set if the method or property is a member of an [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) declared with the **[source]** attribute (as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)).

**FUNCFLAG\_FBINDABLE:** MUST be set if the property was declared with the **[bindable]** attribute (as specified in section [2.2.49.5.2](#Section_ac18004e7af74feb8ae6d42e7a14267c)).

**FUNCFLAG\_FREQUESTEDIT:** MUST be set if the property was declared with the **[requestedit]** attribute (as specified in section 2.2.49.5.2).

**FUNCFLAG\_FDISPLAYBIND:** MUST be set if the property was declared with the **[displaybind]** attribute (as specified in section 2.2.49.5.2).

**FUNCFLAG\_FDEFAULTBIND:** MUST be set if the property was declared with the **[defaultbind]** attribute (as specified in section 2.2.49.5.2).

**FUNCFLAG\_FHIDDEN:** MUST be set if the method or property was declared with the **[hidden]** attribute (as specified in section 2.2.49.5.1).

**FUNCFLAG\_FUSESGETLASTERROR:** MUST be set if the method or property was declared with the **[usesgetlasterror]** attribute (as specified in section [2.2.49.9](#Section_82f9465bae46474e87ffe65e9751affb)) and MUST be ignored on receipt.

**FUNCFLAG\_FDEFAULTCOLLELEM:** MUST be set if the method or property was declared with the **[defaultcollelem]** attribute (as specified in section 2.2.49.5.1).

**FUNCFLAG\_FUIDEFAULT:** MUST be set if the method or property was declared with the **[uidefault]** attribute (as specified in section 2.2.49.5.1).

**FUNCFLAG\_FNONBROWSABLE:** MUST be set if the property was declared with the **[nonbrowsable]** attribute (as specified in section 2.2.49.5.1).

**FUNCFLAG\_FREPLACEABLE:** MUST be set if the property was declared with the **[replaceable]** attribute (as specified in section 2.2.49.5.1). MUST be ignored on receipt.

**FUNCFLAG\_FIMMEDIATEBIND:** MUST be set if the property was declared with the **[immediatebind]** attribute (as specified in section 2.2.49.5.2).

### FUNCKIND Function Access Constants

The FUNCKIND enumeration values are used in the **funckind** field of a [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e) to specify the way that a method is accessed, as specified in section 2.2.42.

The following function access constants are defined in the FUNCKIND enumeration.

1. typedef [v1\_enum] enum tagFUNCKIND
2. {
3. FUNC\_PUREVIRTUAL = 1,
4. FUNC\_STATIC = 3,
5. FUNC\_DISPATCH = 4
6. } FUNCKIND;

**FUNC\_PUREVIRTUAL:** MUST be set if the method described by the **FUNCDESC** structure is a member of an [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) whose associated TYPEKIND value is TKIND\_INTERFACE (as specified in section [2.2.17](#Section_78ccbd1cd8ff43019afcdf562372fb33)).

**FUNC\_STATIC:** MUST be set if the method described by the **FUNCDESC** structure is a method member of the module defined with the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) (as specified in section [2.2.49.9](#Section_82f9465bae46474e87ffe65e9751affb)).

**FUNC\_DISPATCH:** MUST be set if the method described by the **FUNCDESC** structure is a member of an interface whose associated TYPEKIND value is TKIND\_DISPATCH (as specified in section 2.2.17). MUST NOT be set if the FUNC\_PUREVIRTUAL flag is set.

### IMPLTYPEFLAGS Feature Constants

The IMPLTYPEFLAGS enumeration values are stored in the pImplTypeFlags parameter of the [ITypeInfo::GetImplTypeFlags](#Section_4232aaa957a844bfb0d0a02dbdb3e9bc) method to specify the implementation features of a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c), as specified in section 3.7.4.7.

The following implementation type feature constants are defined in the IMPLTYPEFLAGS enumeration.

1. typedef enum tagIMPLTYPEFLAGS
2. {
3. IMPLTYPEFLAG\_FDEFAULT = 0x1,
4. IMPLTYPEFLAG\_FSOURCE = 0x2,
5. IMPLTYPEFLAG\_FRESTRICTED = 0x4,
6. IMPLTYPEFLAG\_FDEFAULTVTABLE = 0x8
7. } IMPLTYPEFLAGS;

**IMPLTYPEFLAG\_FDEFAULT:** MUST be set if the interface was declared with the **[default]** attribute (as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)).

**IMPLTYPEFLAG\_FSOURCE:** MUST be set if the interface was declared with the **[source]** or **[defaultvtable]** attributes (as specified in section 2.2.49.8).

**IMPLTYPEFLAG\_FRESTRICTED:** MUST be set if the interface was declared with the **[restricted]** attribute (as specified in section 2.2.49.8).

**IMPLTYPEFLAG\_FDEFAULTVTABLE:** MUST be set if the interface was declared with the **[defaultvtable]** attribute (as specified in section 2.2.49.8).

### INVOKEKIND Function Invocation Constants

The INVOKEKIND enumeration values are used in the **invkind** field of a [FUNCDESC (section 2.2.42)](#Section_d3349d25e11d4095ba86de3fda178c4e) to specify the way that a method is invoked using [IDispatch::Invoke (section 3.1.4.4)](#Section_5c2a199760d7496d8d9aed940bbb82eb). They are also used in the [ITypeInfo2::GetFuncIndexOfMemId](#Section_9b417eae849f460889f060e3ea04d8a6), [ITypeInfo::GetDllEntry](#Section_d82eb39db2184484a1587b582ab65e5c) and [ITypeComp::Bind](#Section_476f00da080640d9bbf36059154abbb7) methods to distinguish between properties and [**property**](#gt_f930baab-25f1-4142-bced-5effc9f62d45) accessor methods that have the same [MEMBERID (section 2.2.35)](#Section_ace8758fee2b4cb68645973994d12530) but are invoked differently.

Fields and parameters that contain function invocation constants MUST contain a single INVOKEKIND value, and MUST NOT contain bitwise combinations of multiple INVOKEKIND values.

The function invocation constants are defined in the INVOKEKIND enumeration.

1. typedef [v1\_enum] enum tagINVOKEKIND
2. {
3. INVOKE\_FUNC = 0x1,
4. INVOKE\_PROPERTYGET = 0x2,
5. INVOKE\_PROPERTYPUT = 0x4,
6. INVOKE\_PROPERTYPUTREF = 0x8
7. } INVOKEKIND;

**INVOKE\_FUNC:** MUST be set if the type member is a method declared without the **[propget]**, **[propput]**, or **[propputref]** attributes, or to specify that a [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) method request MUST NOT return a property.

**INVOKE\_PROPERTYGET:** MUST be set if the type member is a property declared with the **[propget]** attribute (as specified in section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f)), or to specify that a client method request MUST NOT return anything but an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) property (as specified in section [2.2.49.5.3](#Section_e23aaa6d3ad44886b6520203a1a50c58)) or a property declared with the **[propget]** attribute.

**INVOKE\_PROPERTYPUT:** MUST be set if the type member is a property declared with the **[propput]** attribute (as specified in section 2.2.49.5.1), or to specify that a client method request MUST NOT return anything but a property declared with the **[propput]** attribute.

**INVOKE\_PROPERTYPUTREF:** MUST be set if the type member is a property declared with the **[propputref]** attribute (as specified in section 2.2.49.5.1), or to specify that a client method request MUST NOT return anything but a property declared with the **[propputref]** attribute.

### PARAMFLAGS Parameter Feature Constants

The PARAMFLAGS enumeration values are used in the **wParamFlags** field of a PARAMFLAGS to identify the features of a method parameter, as specified in section [2.2.40](#Section_a965ce8e6c064d7cab302f14b1d8488a).

The following parameter feature constants are defined in the PARAMFLAGS enumeration.

1. typedef enum tagPARAMFLAGS
2. {
3. PARAMFLAG\_NONE = 0,
4. PARAMFLAG\_FIN = 0x1,
5. PARAMFLAG\_FOUT = 0x2,
6. PARAMFLAG\_FLCID = 0x4,
7. PARAMFLAG\_FRETVAL = 0x8,
8. PARAMFLAG\_FOPT = 0x10,
9. PARAMFLAG\_FHASDEFAULT = 0x20,
10. PARAMFLAG\_FHASCUSTDATA = 0x40
11. } PARAMFLAGS;

**PARAMFLAG\_NONE:** The behavior of the parameter is not specified.

**PARAMFLAG\_FIN:** MUST be set if the parameter was declared by using the [in] attribute (for more information, see section [2.2.49.6](#Section_ff2bd74bcb4d48b3ab896bb32cda3833)).

**PARAMFLAG\_FOUT:** MUST be set if the parameter was declared by using the [out] attribute (for more information, see section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862)).

**PARAMFLAG\_FLCID:** MUST be set if the parameter was declared by using the [lcid] attribute (for more information, see section 2.2.49.6).

**PARAMFLAG\_FRETVAL:** MUST be set if the parameter was declared by using the [retval] attribute (for more information, see section 2.2.49.6).

**PARAMFLAG\_FOPT:** MUST be set if the parameter was declared by using the [optional] attribute (for more information, see section 2.2.49.6). MUST be set if the PARAMFLAG\_FHASDEFAULT flag is set.

**PARAMFLAG\_FHASDEFAULT:** MUST be set if the parameter was declared by using the [defaultvalue] attribute (for more information, see section 2.2.49.6).

**PARAMFLAG\_FHASCUSTDATA:** MAY[<2>](#Appendix_A_2" \o "Product behavior note 2) be set if the parameter was declared by using the [custom] attribute (for more information, see section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)).

### TYPEFLAGS Type Feature Constants

The TYPEFLAGS enumeration values are used in the **wTypeFlags** field of a TYPEATTR to specify the features of a type, as specified in section [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e). They also are used in the *pTypeFlags* parameter of the [ITypeInfo2::GetTypeFlags](#Section_5e6482f38bb044038dd3fce5c1071c30) method.

The function invocation constants are defined in the TYPEFLAGS enumeration.

1. typedef enum tagTYPEFLAGS
2. {
3. TYPEFLAG\_FAPPOBJECT = 0x1,
4. TYPEFLAG\_FCANCREATE = 0x2,
5. TYPEFLAG\_FLICENSED = 0x4,
6. TYPEFLAG\_FPREDECLID = 0x8,
7. TYPEFLAG\_FHIDDEN = 0x10,
8. TYPEFLAG\_FCONTROL = 0x20,
9. TYPEFLAG\_FDUAL = 0x40,
10. TYPEFLAG\_FNONEXTENSIBLE = 0x80,
11. TYPEFLAG\_FOLEAUTOMATION = 0x100,
12. TYPEFLAG\_FRESTRICTED = 0x200,
13. TYPEFLAG\_FAGGREGATABLE = 0x400,
14. TYPEFLAG\_FREPLACEABLE = 0x800,
15. TYPEFLAG\_FDISPATCHABLE = 0x1000,
16. TYPEFLAG\_FPROXY = 0x4000
17. } TYPEFLAGS;

**TYPEFLAG\_FAPPOBJECT:** MUST be set if the type was declared with the **[appobject]** attribute (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)).

**TYPEFLAG\_FCANCREATE:** MUST NOT be set if the type was declared with the **[noncreatable]** attribute (see section 2.2.49.8). Otherwise, MUST be set.

**TYPEFLAG\_FLICENSED:** MUST be set if the type was declared with the **[licensed]** attribute (see section 2.2.49.8).

**TYPEFLAG\_FPREDECLID:** MUST be set if the type was declared with the **[predeclid]** or **[appobject]** attributes (see section 2.2.49.8).

**TYPEFLAG\_FHIDDEN:** MUST be set if the type was declared with the **[hidden]** attribute (see section 2.2.49.8).

**TYPEFLAG\_FCONTROL:** MUST be set if the type was declared with the **[control]** attribute (see section 2.2.49.8).

**TYPEFLAG\_FDUAL:** MUST be set if the type was declared with the **[dual]** attribute (see section [2.2.49.4.2](#Section_bd30db0d6c384d549c4467c0e9d25551)).

**TYPEFLAG\_FNONEXTENSIBLE:** MUST be set if the type was declared with the **[nonextensible]** attribute (see section [2.2.49.4](#Section_3b4b512c8c9445a1810dba4ff0152698)).

**TYPEFLAG\_FOLEAUTOMATION:** MUST be set if the type is a [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408) that was declared with the **[oleautomation]** or **[dual]** attributes (see section 2.2.49.4). MUST NOT be set if the type is a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5).

**TYPEFLAG\_FRESTRICTED:** MUST be set if the type was declared with the **[restricted]** attribute (see section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f)).

**TYPEFLAG\_FAGGREGATABLE:** MUST be set if the type was declared with the **[aggregatable]** attribute (see section 2.2.49.8).

**TYPEFLAG\_FREPLACEABLE:** MUST be set if the type contains a member that was declared with the **[replaceable]** attribute (see section 2.2.49.5.1). MUST be ignored on receipt.

**TYPEFLAG\_FDISPATCHABLE:** MUST be set if the type derives from IDispatch, either directly or indirectly. MUST be set if the type is a dispinterface or [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) [<3>](#Appendix_A_3" \o "Product behavior note 3) (see section 2.2.49.4.2).

**TYPEFLAG\_FPROXY:** MUST be set if the type was declared with the **[proxy]** attribute (see section 2.2.49.4). MUST be ignored on receipt.

### TYPEKIND Type Kind Constants

The TYPEKIND enumeration values are used in the **typekind** field of a TYPEATTR to specify the features of a type, as specified in section [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e). They are also used in the *pTypeKind* parameter of the [ITypeInfo2::GetTypeKind](#Section_6a03300ebd2d45e4b15e2a4c121554e5) method, as specified in section 3.9.4.1.

The type kind constants are defined in the TYPEKIND enumeration.

1. typedef [v1\_enum] enum tagTYPEKIND
2. {
3. TKIND\_ENUM = 0x0,
4. TKIND\_RECORD = 0x1,
5. TKIND\_MODULE = 0x2,
6. TKIND\_INTERFACE = 0x3,
7. TKIND\_DISPATCH = 0x4,
8. TKIND\_COCLASS = 0x5,
9. TKIND\_ALIAS = 0x6,
10. TKIND\_UNION = 0x7
11. } TYPEKIND;

**TKIND\_ENUM:** MUST be used if the type is an enumeration that was defined with the **typedef** and **enum** keywords.

**TKIND\_RECORD:** MUST be used if the type is a structure that was defined with the **typedef** and **struct** keywords.

**TKIND\_MODULE:** MUST be used if the type is a module that was defined with the **module** keyword.

**TKIND\_INTERFACE:** MUST be used if the type is a [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408) that was defined with the interface keyword.

**TKIND\_DISPATCH:** MUST be used if the type is a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) that was defined with either the **dispinterface** keyword or the **interface** keyword with the **[dual]** attribute.

**TKIND\_COCLASS:** MUST be used if the type is a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that was defined with the **coclass** keyword.

**TKIND\_ALIAS:** MUST be used if the type is an alias for a predefined type that was defined with the **typedef** keyword and added to the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) by using the **[public]** attribute as specified in section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e).

**TKIND\_UNION:** MUST be used if the type is a union that was defined with the **typedef** and **union** keywords.

### VARFLAGS Variable Feature Constants

The VARFLAGS enumeration values are used in the **wVarFlags** field of a [VARDESC](#Section_ae7791d243994dffb7c6b0d4f3dce982) to specify the features of a field, constant, or [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) [**property**](#gt_f930baab-25f1-4142-bced-5effc9f62d45), as specified in section 2.2.43.

The variable feature constants are defined in the VARFLAGS enumeration.

1. typedef enum tagVARFLAGS
2. {
3. VARFLAG\_FREADONLY = 0x1,
4. VARFLAG\_FSOURCE = 0x2,
5. VARFLAG\_FBINDABLE = 0x4,
6. VARFLAG\_FREQUESTEDIT = 0x8,
7. VARFLAG\_FDISPLAYBIND = 0x10,
8. VARFLAG\_FDEFAULTBIND = 0x20,
9. VARFLAG\_FHIDDEN = 0x40,
10. VARFLAG\_FRESTRICTED = 0x80,
11. VARFLAG\_FDEFAULTCOLLELEM = 0x100,
12. VARFLAG\_FUIDEFAULT = 0x200,
13. VARFLAG\_FNONBROWSABLE = 0x400,
14. VARFLAG\_FREPLACEABLE = 0x800,
15. VARFLAG\_FIMMEDIATEBIND = 0x1000
16. } VARFLAGS;

**VARFLAG\_FREADONLY:** MUST be set if the variable is an ODL dispinterface property that was declared with the [readonly] attribute (see section [2.2.49.5.3](#Section_e23aaa6d3ad44886b6520203a1a50c58)).

**VARFLAG\_FSOURCE:** MUST be set if the variable is a property member of an ODL interface that was declared with the [source] attribute (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)).

**VARFLAG\_FBINDABLE:** MUST be set if the variable is an ODL dispinterface property that was declared with the [bindable] attribute (see section [2.2.49.5.2](#Section_ac18004e7af74feb8ae6d42e7a14267c)).

**VARFLAG\_FREQUESTEDIT:** MUST be set if the variable is an ODL dispinterface property that was declared with the [requestedit] attribute (see section 2.2.49.5.2).

**VARFLAG\_FDISPLAYBIND:** MUST be set if the variable is an ODL dispinterface property that was declared with the [displaybind] attribute (see section 2.2.49.5.2).

**VARFLAG\_FDEFAULTBIND:** MUST be set if the variable is an ODL dispinterface property that was declared with the [defaultbind] attribute (see section 2.2.49.5.2).

**VARFLAG\_FHIDDEN:** MUST be set if the variable is a member of a type that was declared with the [hidden] attribute (see section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f)).

**VARFLAG\_FRESTRICTED:** MUST be set if the variable is a member of a type that was declared with the [restricted] attribute (see section 2.2.49.5.1).

**VARFLAG\_FDEFAULTCOLLELEM:** MUST be set if the variable is an ODL dispinterface property that was declared with the [defaultcollelem] attribute (see section 2.2.49.5.1).

**VARFLAG\_FUIDEFAULT:** MUST be set if the variable is an ODL dispinterface property that was declared with the [uidefault] attribute (see section 2.2.49.5.1).

**VARFLAG\_FNONBROWSABLE:** MUST be set if the variable is an ODL dispinterface property that was declared with the [nonbrowsable] attribute (see section 2.2.49.5.1).

**VARFLAG\_FREPLACEABLE:** MUST be set if the variable is an ODL dispinterface property that was declared with the [replaceable] attribute (see section 2.2.49.5.1). MUST be ignored on receipt.

**VARFLAG\_FIMMEDIATEBIND:** MUST be set if the variable is an ODL dispinterface property that was declared with the [immediatebind] attribute (see section 2.2.49.5.2).

### VARKIND Variable Kind Constants

The VARKIND enumeration values are used in the **varkind** field of a [VARDESC](#Section_ae7791d243994dffb7c6b0d4f3dce982) to specify the kind of element that is described by the VARDESC, as specified in section 2.2.43.

The variable kind constants are defined in the VARKIND enumeration:

1. typedef [v1\_enum] enum tagVARKIND
2. {
3. VAR\_PERINSTANCE = 0,
4. VAR\_STATIC = (VAR\_PERINSTANCE + 1),
5. VAR\_CONST = (VAR\_STATIC + 1),
6. VAR\_DISPATCH = (VAR\_CONST + 1)
7. } VARKIND;

**VAR\_PERINSTANCE:** MUST be used if the VARDESC describes a member of a structure or union.

**VAR\_STATIC:** MUST be used if the VARDESC describes an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)).

**VAR\_CONST:** MUST be used if the VARDESC describes a member of a module or enumeration.

**VAR\_DISPATCH:** MUST be used if the VARDESC describes an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) property (see section [2.2.49.5.3](#Section_e23aaa6d3ad44886b6520203a1a50c58)).

### LIBFLAGS Type Library Feature Constants

The LIBFLAGS enumeration values are used in the **wLibFlags** field of a TLIBATTR to specify the features of the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) of an ITypeLib [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703), as specified in section [2.2.45](#Section_b568f4be95e5431bbb3b08dc56e9b224).

The Type library feature constants are defined in the LIBFLAGS enumeration.

1. typedef [v1\_enum] enum tagLIBFLAGS
2. {
3. LIBFLAG\_FRESTRICTED = 0x01,
4. LIBFLAG\_FCONTROL = 0x02,
5. LIBFLAG\_FHIDDEN = 0x04,
6. LIBFLAG\_FHASDISKIMAGE = 0x08
7. } LIBFLAGS;

**LIBFLAG\_FRESTRICTED:** MUST be set if the automation scope was declared with the **[restricted]** attribute (as specified in section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)).

**LIBFLAG\_FCONTROL:** MUST be set if the automation scope was declared with the **[control]** attribute (as specified in section 2.2.49.2).

**LIBFLAG\_FHIDDEN:** MUST be set if the automation scope was declared with the **[hidden]** attribute (as specified in section 2.2.49.2).

**LIBFLAG\_FHASDISKIMAGE:** MAY be set [<4>](#Appendix_A_4" \o "Product behavior note 4)and MUST be ignored on receipt.

### SYSKIND System Pointer Size Constants

SYSKIND is used in the **syskind** field of a [TLIBATTR](#Section_b568f4be95e5431bbb3b08dc56e9b224) to specify the system pointer size value, as specified in section 2.2.45.

The system pointer size constants are defined in the SYSKIND enumeration.

1. typedef [v1\_enum] enum tagSYSKIND
2. {
3. SYS\_WIN32 = 1,
4. SYS\_WIN64 = 3
5. } SYSKIND;

**SYS\_WIN32:** MUST be set if the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) uses 32 bits for pointer-sized values.

**SYS\_WIN64:** MUST be set if the automation type library uses 64 bits for pointer-sized values.

### DESCKIND Name Description Constants

The DESCKIND Name Description Constants enumeration values are used by the [ITypeComp::Bind](#Section_476f00da080640d9bbf36059154abbb7) method to indicate the kind of element to which a name has been bound, as specified in section 3.5.4.1.

The name description constants are defined in the DESCKIND enumeration.

1. typedef [v1\_enum] enum tagDESCKIND
2. {
3. DESCKIND\_NONE = 0,
4. DESCKIND\_FUNCDESC = 1,
5. DESCKIND\_VARDESC = 2,
6. DESCKIND\_TYPECOMP = 3,
7. DESCKIND\_IMPLICITAPPOBJ = 4
8. } DESCKIND;

**DESCKIND\_NONE:** MUST be set if there is no element bound to the name.

**DESCKIND\_FUNCDESC:** MUST be set if the name is bound to a method or property accessor method. MUST NOT be set if the name is bound to an [ODL dispinterface](#Section_5583e1b8454c41479f56f72416a15bee) property.

**DESCKIND\_VARDESC:** MUST be set if the name is bound to a data element or ODL dispinterface property.

**DESCKIND\_TYPECOMP:** MUST be set if the name is bound to an enumeration or module.

**DESCKIND\_IMPLICITAPPOBJ:** MUST be set if the name is bound to an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)) or a member of its default nonsource interface (also see 2.2.49.8).

### BSTR

BSTR is an OLE [**automation type**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e) for transferring length-prefixed strings, either Unicode or ANSI, as well as length-prefixed binary data.

The **BSTR** type defined in this section specifies the wire representation of a length-prefixed data block whose memory representation is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.5. To clarify, the memory specification will be referred to as the presented **BSTR**, and the wire specification will be referred to as the transmitted **BSTR**.

For any document referencing both [MS-DTYP] and [[MS-OAUT]](%5bMS-OAUT%5d.pdf#Section_bbb05720f72445c78d17f83c3d1a3961), specifying **BSTR** in a wire representation context MUST be considered as a reference to the transmitted **BSTR** type, while specifying **BSTR** in a memory representation context MUST be considered as a reference to the presented **BSTR** type ([MS-DTYP] section 2.2.5). Reflecting the terminology used for presented **BSTRs**, a **NULL** **BSTR**, or NULL transmitted **BSTR**, is defined as the wire representation of a NULL presented BSTR; and an empty **BSTR**, or empty transmitted **BSTR**, is defined as the wire representation of a zero-length presented **BSTR**. Preserving this distinction in the wire representation enables clients and servers to distinguish between NULL presented **BSTRs** and zero-length presented **BSTRs**, and thus associate possibly different, application-specific semantics to these two values.

#### FLAGGED\_WORD\_BLOB

The FLAGGED\_WORD\_BLOB structure defines a type for transferring length-prefixed data.

1. typedef struct \_FLAGGED\_WORD\_BLOB {
2. unsigned long cBytes;
3. unsigned long clSize;
4. [size\_is(clSize)] unsigned short asData[];
5. } FLAGGED\_WORD\_BLOB;

**cBytes:**  MUST be the size, in bytes, of the **asData** array.

**Note**  A value of 0xFFFFFFFF MUST be considered as representing a null **BSTR**.

**clSize:**  MUST be the total number of unsigned shorts in the asData array. This value MUST be half the value of **cBytes**, rounded up, unless this is a null **BSTR**. In the latter case, a value of 0 MUST be used.

**asData:**  An array of unsigned shorts. If **clSize** is 0, **asData** MUST not contain any elements.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824), section 14, with the exception that it MUST be marshaled by using a little-endian data representation regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### BSTR Type Definition

BSTR Type Definition is an OLE [**Automation type**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e) for transferring length-prefixed data.

This type is declared as follows:

1. typedef [unique] FLAGGED\_WORD\_BLOB\* BSTR;

A null transmitted **BSTR** is a pointer to a FLAGGED\_WORD\_BLOB whose **cBytes** field MUST equal 0xFFFFFFFF, **clSize** field MUST equal 0, and **asData** MUST NOT contain any elements.An empty transmitted **BSTR** is a pointer to a [FLAGGED\_WORD\_BLOB](#Section_f547135ad76a42c3916f30b4e46d79bc) whose **cBytes** and **clSize** fields MUST both equal 0, and **asData** MUST NOT contain any elements.

#### Mapping Between Presented and Transmitted BSTRs

The mapping from the memory representation of a [BSTR](#Section_1c9d2cfccf7d4f4b95bf584be5defd81) to the wire representation MUST be performed by setting the fields of **FLAGGED\_WORD\_BLOB** referred by the transmitted **BSTR** as specified below:

**cBytes:** MUST be set to the value of the presented **BSTR**'s length prefix, or 0xFFFFFFFF for a NULL presented **BSTR**.

**clSize:** MUST be half the value of **cBytes** rounded up, or 0 for a NULL presented **BSTR**.

**asData:** MUST contain exactly **clSize** elements, and MUST be filled from the contents of the data pointed to by the presented **BSTR**.

The mapping from the transmitted **BSTR** to the presented **BSTR** MUST be performed as follows:

* The length prefix MUST be set to the value of the **cBytes** field of the transmitted **BSTR**'s **FLAGGED\_WORD\_BLOB**.
* The data pointed to by the presented BSTR MUST be set to the contents of the **asData** field of the transmitted **BSTR**'s **FLAGGED\_WORD\_BLOB**.

### CURRENCY

The CURRENCY type specifies currency information. It is represented as an 8-byte integer, scaled by 10,000, to give a fixed-point number with 15 digits to the left of the decimal point, and four digits to the right. This representation provides a range of 922337203685477.5807 to –922337203685477.5808. For example, $5.25 is stored as the value 52500.

1. typedef struct tagCY {
2. \_\_int64 int64;
3. } CURRENCY;

### DATE

DATE is a type that specifies date and time information. It is represented as an 8-byte floating-point number.

This type is declared as follows:

1. typedef double DATE;

The date information is represented by whole-number increments, starting with December 30, 1899 midnight as time zero. The time information is represented by the fraction of a day since the preceding midnight. For example, 6:00 A.M. on January 4, 1900 would be represented by the value 5.25 (5 and 1/4 of a day past December 30, 1899).

### DECIMAL

The DECIMAL structure specifies a sign and scale for a number. Decimal variables are represented as 96-bit unsigned integers that are scaled by a variable power of 10.

1. typedef struct tagDEC {
2. WORD wReserved;
3. BYTE scale;
4. BYTE sign;
5. ULONG Hi32;
6. ULONGLONG Lo64;
7. } DECIMAL;

**wReserved:**  MUST be set to 0 and MUST be ignored by the recipient.

**scale:**  MUST be the power of 10 by which to divide the 96-bit integer represented by Hi32 \* 2^64 + Lo64. The value MUST be in the range of 0 to 28, inclusive.

| Value | Meaning |
| --- | --- |
| 0 — 28 | Order of magnitude of the decimal number. |

**sign:**  MUST equal one of the following values.

| Value | Meaning |
| --- | --- |
| 0 | The decimal contains a positive value. |
| 0x80 | The decimal contains a negative value. |

**Hi32:**  MUST be the high 32 bits of the 96-bit integer that is scaled and signed to represent the final DECIMAL value.

**Lo64:**  MUST be the low 64 bits of the 96-bit integer that is scaled and signed to represent the final DECIMAL value.

### VARIANT\_BOOL

The VARIANT\_BOOL type specifies Boolean values.

This type is declared as follows:

1. typedef short VARIANT\_BOOL;

The values MUST be defined as:

| Value | Meaning |
| --- | --- |
| VARIANT\_TRUE  0xFFFF | MUST indicate a Boolean value of TRUE. |
| VARIANT\_FALSE  0x0000 | MUST indicate a Boolean value of FALSE. |

### User-Defined Data Types and BRECORD

#### User-Defined Data Types

The OLE Automation Protocol supports the specification of [**UDTs**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727), in the form of structures of related data elements. The data elements MUST consist of scalar and OLE [**automation types**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e). The structure is considered to be a single type and is referred to as a UDT.

In the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824), a UDT MUST be identified by the uuid attribute (see section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)). The GUID from this attribute MUST uniquely identify the UDT.

A UDT MUST be specified or referenced from an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) (see section 2.2.49.2). For more information about the wire representation of the UDT, see sections [2.2.31](#Section_deb939dfef4d49c384677265669e89ed) and 2.2.49.2.

#### BRECORD

A BRECORD is the type used for the representation of [**UDTs**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727) on the wire.

##### \_wireBRECORD

The \_wireBRECORD structure is the wire representation of a collection of [**UDTs**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727) of the same type. This representation MUST be used when the UDTs appear inside a [SAFEARRAY (section 2.2.30.10)](#Section_2e87a537930541c6a88bb79809b3703a) or inside a [VARIANT (section 2.2.29.2)](#Section_a6a540af38ac48bebd4092e2c01e9aa6). Otherwise, the UDTs MUST be [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d)-marshaled as specified by their [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824). For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14.

1. typedef struct \_wireBRECORD {
2. ULONG fFlags;
3. ULONG clSize;
4. MInterfacePointer\* pRecInfo;
5. [size\_is(clSize)] byte\* pRecord;
6. } wireBRECORDStr;

**fFlags:**   MUST be 0 if pRecord is NULL. Otherwise, the value MUST be 1.

**clSize:**  MUST be 0 if **pRecord** is NULL. Otherwise, the value MUST equal the size (in bytes) of the UDTs contained in **pRecord**, plus 4 bytes to account for the prefix contained in **pRecord**.

**pRecInfo:**   MUST specify an MInterfacePointer that MUST contain an OBJREF\_CUSTOM with a CLSID field set to CLSID\_RecordInfo ([1.9](#Section_58504586e4af44a3be04f1dc281b7429)) and a pObjectData field that MUST contain a RecordInfoData binary large object (BLOB) ([2.2.31](#Section_deb939dfef4d49c384677265669e89ed)). The iid field of the OBJREF portion of the structure MUST be set to IID\_IRecordInfo (1.9). An implementation MAY use this value as the IID of a local-only interface.[<5>](#Appendix_A_5" \o "Product behavior note 5)

**pRecord:**  MUST be NULL if there are no UDTs. Otherwise, the value MUST contain the NDR-marshaled representation of the UDTs, prefixed by a 4-byte unsigned integer that specifies the size, in bytes. This integer MUST equal the value of **clSize**.

Data of this type MUST be marshaled as specified in [C706] section 14, with the exception that the fields **fFlags**, **clSize**, and the 4-byte prefix in **pRecord** MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

##### BRECORD

The following is the type definition for the BRECORD type.

This type is declared as follows:

1. typedef [unique] struct \_wireBRECORD\* BRECORD;

### VARIANT

VARIANT is a container for a union that can hold many types of data.

#### \_wireVARIANT

The \_wireVARIANT is a container for a union that in turn contains scalar and OLE Automation data types.

1. typedef struct \_wireVARIANT {
2. DWORD clSize;
3. DWORD rpcReserved;
4. USHORT vt;
5. USHORT wReserved1;
6. USHORT wReserved2;
7. USHORT wReserved3;
8. [switch\_type(ULONG), switch\_is(vt)]
9. union {
10. [case(VT\_I8)]
11. LONGLONG llVal;
12. [case(VT\_I4)]
13. LONG lVal;
14. [case(VT\_UI1)]
15. BYTE bVal;
16. [case(VT\_I2)]
17. SHORT iVal;
18. [case(VT\_R4)]
19. FLOAT fltVal;
20. [case(VT\_R8)]
21. DOUBLE dblVal;
22. [case(VT\_BOOL)]
23. VARIANT\_BOOL boolVal;
24. [case(VT\_ERROR)]
25. HRESULT scode;
26. [case(VT\_CY)]
27. CURRENCY cyVal;
28. [case(VT\_DATE)]
29. DATE date;
30. [case(VT\_BSTR)]
31. BSTR bstrVal;
32. [case(VT\_UNKNOWN)]
33. IUnknown\* punkVal;
34. [case(VT\_DISPATCH)]
35. IDispatch\* pdispVal;
36. [case(VT\_ARRAY)]
37. PSAFEARRAY parray;
38. [case(VT\_RECORD, VT\_RECORD|VT\_BYREF)]
39. BRECORD brecVal;
40. [case(VT\_UI1|VT\_BYREF)]
41. BYTE\* pbVal;
42. [case(VT\_I2|VT\_BYREF)]
43. SHORT\* piVal;
44. [case(VT\_I4|VT\_BYREF)]
45. LONG\* plVal;
46. [case(VT\_I8|VT\_BYREF)]
47. LONGLONG\* pllVal;
48. [case(VT\_R4|VT\_BYREF)]
49. FLOAT\* pfltVal;
50. [case(VT\_R8|VT\_BYREF)]
51. DOUBLE\* pdblVal;
52. [case(VT\_BOOL|VT\_BYREF)]
53. VARIANT\_BOOL\* pboolVal;
54. [case(VT\_ERROR|VT\_BYREF)]
55. HRESULT\* pscode;
56. [case(VT\_CY|VT\_BYREF)]
57. CURRENCY\* pcyVal;
58. [case(VT\_DATE|VT\_BYREF)]
59. DATE\* pdate;
60. [case(VT\_BSTR|VT\_BYREF)]
61. BSTR\* pbstrVal;
62. [case(VT\_UNKNOWN|VT\_BYREF)]
63. IUnknown\*\* ppunkVal;
64. [case(VT\_DISPATCH|VT\_BYREF)]
65. IDispatch\*\* ppdispVal;
66. [case(VT\_ARRAY|VT\_BYREF)]
67. PSAFEARRAY\* pparray;
68. [case(VT\_VARIANT|VT\_BYREF)]
69. VARIANT\* pvarVal;
70. [case(VT\_I1)]
71. CHAR cVal;
72. [case(VT\_UI2)]
73. USHORT uiVal;
74. [case(VT\_UI4)]
75. ULONG ulVal;
76. [case(VT\_UI8)]
77. ULONGLONG ullVal;
78. [case(VT\_INT)]
79. INT intVal;
80. [case(VT\_UINT)]
81. UINT uintVal;
82. [case(VT\_DECIMAL)]
83. DECIMAL decVal;
84. [case(VT\_I1|VT\_BYREF)]
85. CHAR\* pcVal;
86. [case(VT\_UI2|VT\_BYREF)]
87. USHORT\* puiVal;
88. [case(VT\_UI4|VT\_BYREF)]
89. ULONG\* pulVal;
90. [case(VT\_UI8|VT\_BYREF)]
91. ULONGLONG\* pullVal;
92. [case(VT\_INT|VT\_BYREF)]
93. INT\* pintVal;
94. [case(VT\_UINT|VT\_BYREF)]
95. UINT\* puintVal;
96. [case(VT\_DECIMAL|VT\_BYREF)]
97. DECIMAL\* pdecVal;
98. [case(VT\_EMPTY)]
99. ;
100. [case(VT\_NULL)]
101. ;
102. } \_varUnion;
103. } wireVARIANTStr;

**clSize:**  MUST be set to the size, in quad words (64 bits), of the structure.

**rpcReserved:**  MUST be set to 0 and MUST be ignored by the recipient.

**vt:**   MUST be set to one of the values specified with a "V" in the Context column of the table in section [2.2.7](#Section_3fe7db9f58034dc49d145425d3f5461f).

**wReserved1:**  MAY be set to 0 and MUST be ignored by the recipient.[<6>](#Appendix_A_6" \o "Product behavior note 6)

**wReserved2:**  MAY be set to 0 and MUST be ignored by the recipient.[<7>](#Appendix_A_7" \o "Product behavior note 7)

**wReserved3:**  MAY be set to 0 and MUST be ignored by the recipient.[<8>](#Appendix_A_8" \o "Product behavior note 8)

**\_varUnion:**  MUST contain an instance of the type, according to the value in the **vt** field.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the following additional restrictions.

* All fields except **\_varUnion** MUST be marshaled using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.
* If the **vt** field has the flag VT\_ARRAY set, then **\_varUnion** MUST be marshaled according to [2.2.30](#Section_04e72b3f573145089bb4de29fbd0f781).
* If the **vt** field has the flags VT\_UNKNOWN or VT\_DISPATCH set, then **\_varUnion** MUST be marshaled according to [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) section 1.3.2.
* If the **vt** field has the flag VT\_RECORD set, then **\_varUnion** field MUST be marshaled according to [2.2.28](#Section_29ce0a4f478649c9a3125522c1e9b44d).
* If the **vt** field has the flag VT\_BSTR set, then **\_varUnion** MUST be marshaled according to [2.2.23](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1).
* If none of the preceding flags is specified in the **vt** field, the **\_varUnion** field MUST be marshaled by using a little-endian data representation, regardless of the data representation format label.

#### VARIANT

The VARIANT type is defined as follows. Also, the definitions of the VARIANT data type provided in this section correspond to the wire formats of these data types.[<9>](#Appendix_A_9" \o "Product behavior note 9)

This type is declared as follows:

1. typedef [unique] struct \_wireVARIANT\* VARIANT;

### SAFEARRAY

A SAFEARRAY specifies a multidimensional array of OLE [**Automation types**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e).

#### SAFEARRAYBOUND

The SAFEARRAYBOUND structure specifies the bounds of one dimension of a [SAFEARRAY](#Section_2e87a537930541c6a88bb79809b3703a).

1. typedef struct tagSAFEARRAYBOUND {
2. ULONG cElements;
3. LONG lLbound;
4. } SAFEARRAYBOUND,
5. \*LPSAFEARRAYBOUND;

**cElements:**  MUST be set to the number of elements in the current dimension. MUST be nonzero.

**lLbound:**  MUST be set to the lower bound of the current dimension.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, except that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARR\_BSTR

The SAFEARR\_BSTR structure specifies an array of BSTRs (see section [2.2.23](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1)).

1. typedef struct \_wireSAFEARR\_BSTR {
2. ULONG Size;
3. [size\_is(Size), ref] BSTR\* aBstr;
4. } SAFEARR\_BSTR;

**Size:**  MUST be set to the total number of elements in the array.

**aBstr:**  MUST be an array of BSTRs (see section 2.2.23).

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARR\_UNKNOWN

The SAFEARR\_UNKNOWN structure specifies an array of MInterfacePointers elements (see [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) section 2.2.14) whose IPID is IID\_IUnknown (see section [1.9](#Section_58504586e4af44a3be04f1dc281b7429)).

1. typedef struct \_wireSAFEARR\_UNKNOWN {
2. ULONG Size;
3. [size\_is(Size), ref] IUnknown\*\* apUnknown;
4. } SAFEARR\_UNKNOWN;

**Size:**  MUST be set to the total number of elements in the array.

**apUnknown:**  MUST be an array of MInterfacePointer (see [MS-DCOM], section 2.2.1.10). The iid field in the [**OBJREF**](#gt_d2b5b331-4e5f-417f-92c5-38eef54176c2) MUST be IID\_IUnknown (see section 1.9).

The **Size** field of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARR\_DISPATCH

The SAFEARR\_DISPATCH structure specifies an array of MInterfacePointer elements (see [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) section 2.2.14) whose IPID is IID\_IDispatch (see section [1.9](#Section_58504586e4af44a3be04f1dc281b7429)).

1. typedef struct \_wireSAFEARR\_DISPATCH {
2. ULONG Size;
3. [size\_is(Size), ref] IDispatch\*\* apDispatch;
4. } SAFEARR\_DISPATCH;

**Size:**  MUST be set to the total number of elements in the array.

**apDispatch:**  MUST be an array of MInterfacePointer elements (see [MS-DCOM] section 2.2.14). The **iid** field in the [**OBJREF**](#gt_d2b5b331-4e5f-417f-92c5-38eef54176c2) MUST be IID\_IDispatch (see section 1.9).

The **Size** field of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARR\_VARIANT

The SAFEARR\_VARIANT structure specifies an array of [VARIANT](#Section_a6a540af38ac48bebd4092e2c01e9aa6) types.

1. typedef struct \_wireSAFEARR\_VARIANT {
2. ULONG Size;
3. [size\_is(Size), ref] VARIANT\* aVariant;
4. } SAFEARR\_VARIANT;

**Size:**  MUST be set to the total number of elements in the array. MUST be nonzero.

**aVariant:**  MUST be an array of VARIANT types. For more information, see section 2.2.29.2.

The **Size** field of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARR\_BRECORD

The SAFEARR\_BRECORD structure specifies an array of [**UDTs**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727).

1. typedef struct \_wireSAFEARR\_BRECORD {
2. ULONG Size;
3. [size\_is(Size), ref] BRECORD\* aRecord;
4. } SAFEARR\_BRECORD;

**Size:**  The number of BRECORD elements in the aRecord array. This MUST be set to 1.

**aRecord:**  MUST be the collection of UDTs as specified in BRECORD (see section [2.2.28.2](#Section_ea064b3d9fb3448699924fe2463e83e8)).

The **Size** field of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARR\_HAVEIID

The SAFEARR\_HAVEIID structure defines an array of MInterfacePointers (see [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) section 2.2.14

1. typedef struct \_wireSAFEARR\_HAVEIID {
2. ULONG Size;
3. [size\_is(Size), ref] IUnknown\*\* apUnknown;
4. IID iid;
5. } SAFEARR\_HAVEIID;

**Size:**  MUST be set to the total number of elements in the array. This MUST be nonzero.

**apUnknown:**  MUST be an array of MInterfacePointer elements. The [**OBJREF**](#gt_d2b5b331-4e5f-417f-92c5-38eef54176c2) **iid** field MUST be the same as the value of the **iid**.

**iid:**  MUST specify the [IID](#Section_94ea3f27ba2945f3a821323f0a0c6d60) of each of the elements in the SAFEARRAY.

The **Size** and **iid** fields of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### Scalar-Sized Arrays

The following four types represent byte-sized, short-sized, long-sized, and hyper-sized arrays.

##### BYTE\_SIZEDARR

The BYTE\_SIZEDARR structure specifies a BYTE array.

1. typedef struct \_BYTE\_SIZEDARR {
2. unsigned long clSize;
3. [size\_is(clSize)] byte\* pData;
4. } BYTE\_SIZEDARR;

**clSize:**  MUST be set to the total number of elements in the array. This MUST be nonzero.

**pData:**  MUST be an array of BYTEs.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

##### WORD\_SIZEDARR

The WORD\_SIZEDARR structure defines an array of unsigned 16-bit integers.

1. typedef struct \_SHORT\_SIZEDARR {
2. unsigned long clSize;
3. [size\_is(clSize)] unsigned short\* pData;
4. } WORD\_SIZEDARR;

**clSize:**  MUST be set to the total number of elements in the array. This MUST be nonzero.

**pData:**  MUST be an array of WORD elements.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

##### DWORD\_SIZEDARR

The DWORD\_SIZEDARR structure defines an array of unsigned 32-bit integers.

1. typedef struct \_LONG\_SIZEDARR {
2. unsigned long clSize;
3. [size\_is(clSize)] unsigned long\* pData;
4. } DWORD\_SIZEDARR;

**clSize:**  MUST be set to the number of elements within the array. This MUST be nonzero.

**pData:**  MUST be an array of DWORD elements.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

##### HYPER\_SIZEDARR

The HYPER\_SIZEDARR structure defines an array of 64-bit integers.

1. typedef struct \_HYPER\_SIZEDARR {
2. unsigned long clSize;
3. [size\_is(clSize)] hyper\* pData;
4. } HYPER\_SIZEDARR;

**clSize:**  MUST be set to the total number of elements in the array. This MUST be nonzero.

**pData:**  MUST be an array of hyper elements.

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARRAYUNION

The SAFEARRAYUNION union defines the mapping between the discriminant value and the contained array data.

1. typedef
2. union \_wireSAFEARRAY\_UNION switch(unsigned long sfType) u {
3. case SF\_BSTR : SAFEARR\_BSTR BstrStr;
4. case SF\_UNKNOWN : SAFEARR\_UNKNOWN UnknownStr;
5. case SF\_DISPATCH : SAFEARR\_DISPATCH DispatchStr;
6. case SF\_VARIANT : SAFEARR\_VARIANT VariantStr;
7. case SF\_RECORD : SAFEARR\_BRECORD RecordStr;
8. case SF\_HAVEIID : SAFEARR\_HAVEIID HaveIidStr;
9. case SF\_I1 : BYTE\_SIZEDARR ByteStr;
10. case SF\_I2 : WORD\_SIZEDARR WordStr;
11. case SF\_I4 : DWORD\_SIZEDARR LongStr;
12. case SF\_I8 : HYPER\_SIZEDARR HyperStr;
13. } SAFEARRAYUNION;

\_wireSAFEARRAY\_UNION: MUST contain an instance of the type, according to the value of the union discriminant.

The **sfType** field MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

#### SAFEARRAY

The SAFEARRAY structure defines a multidimensional array of OLE [**automation types**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e). The definitions of SAFEARRAY data types provided in this section correspond to the wire formats of these data types.[<10>](#Appendix_A_10" \o "Product behavior note 10)

1. typedef[unique]
2. struct \_wireSAFEARRAY {
3. USHORT cDims;
4. USHORT fFeatures;
5. ULONG cbElements;
6. ULONG cLocks;
7. SAFEARRAYUNION uArrayStructs;
8. [size\_is(cDims)] SAFEARRAYBOUND rgsabound[];
9. } \*SAFEARRAY;

**cDims:**  MUST be set to the number of dimensions of the array. cDims MUST NOT be set to 0.

**fFeatures:**  MUST be set to a combination of the bit flags specified in section [2.2.9](#Section_f06ee3d2a61f4e0ba9299369d334ea33).

**cbElements:**  MUST be set to the size, in bytes, of an element in the SAFEARRAY, as specified in the table in section [2.2.8](#Section_8c78fede6f6c4822a5f80fcbbc8d8242).

**cLocks:**  If the **fFeatures** field contains FADF\_HAVEVARTYPE (see section 2.2.9), the **cLocks** field MUST contain a [VARIANT (section 2.2.7)](#Section_3fe7db9f58034dc49d145425d3f5461f) type constant in its high word that specifies the type of the elements in the array. Otherwise, the high word of the **cLocks** field MUST be set to 0.

The low word of the **cLocks** field MAY[<11>](#Appendix_A_11" \o "Product behavior note 11) be set to an implementation-specific value, and MUST be ignored on receipt.

**uArrayStructs:**  MUST be a [SAFEARRAYUNION (section 2.2.30.9)](#Section_5eaa490ff6c54d58b368cf2d0ea74572).

**rgsabound:**  MUST contain an array of bounds, specifying the shape of the array. This array MUST be represented in reverse order. That is, for an array [0:5][0:2][0:10], the bounds would be represented as (10, 0), (2, 0), (5, 0).

The following consistency statements MUST hold, where **sfType** is the discriminant field in the SAFEARRAYUNION data member.

| If sfType equals | fFeatures MUST be set to |
| --- | --- |
| SF\_HAVEIID | FADF\_UNKNOWN | FADF\_HAVEIID or  FADF\_DISPATCH | FADF\_HAVEIID |
| SF\_BSTR | FADF\_BSTR or  FADF\_BSTR | FADF\_HAVEVARTYPE |
| SF\_UNKNOWN | FADF\_UNKNOWN or  FADF\_UNKNOWN | FADF\_HAVEVARTYPE or  FADF\_UNKNOWN | FADF\_HAVEIID |
| SF\_DISPATCH | FADF\_DISPATCH or  FADF\_DISPATCH | FADF\_HAVEVARTYPE or  FADF\_DISPATCH | FADF\_HAVEIID |
| SF\_VARIANT | FADF\_VARIANT or  FADF\_VARIANT | FADF\_HAVEVARTYPE |
| SF\_RECORD | FADF\_RECORD |

If **fFeatures** field specifies FADF\_HAVEVARTYPE, the following additional statements MUST hold, where **vt** is the high word of the **cLocks** field.

| If vt (the high word of cLocks) equals | sfType MUST be set to |
| --- | --- |
| VT\_I1  VT\_UI1 | SF\_I1 |
| VT\_I2  VT\_UI2  VT\_BOOL | SF\_I2 |
| VT\_ERROR  VT\_I4  VT\_UI4  VT\_R4  VT\_INT  VT\_UINT | SF\_I4 |
| VT\_I8  VT\_UI8  VT\_R8  VT\_CY  VT\_DATE | SF\_I8 |
| VT\_BSTR | SF\_BSTR |
| VT\_VARIANT | SF\_VARIANT |
| VT\_UNKNOWN | SF\_UNKNOWN or SF\_HAVEIID |
| VT\_DISPATCH | SF\_DISPATCH or SF\_HAVEIID |
| VT\_RECORD | SF\_RECORD |

In addition, the type of **vt** MUST NOT equal VT\_DECIMAL.

If any of the consistency checks fail, the protocol implementation SHOULD[<12>](#Appendix_A_12" \o "Product behavior note 12)

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

### RecordInfoData

The RecordInfoData structure specifies information that is needed to identify the definition of a [**UDT**](#gt_10a36f2b-2a1d-4d7f-b57d-261afca73727), such as the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) and version of the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) that defines the UDT, and the GUID of the type itself.

1. typedef struct tagRecordInfo {
2. GUID libraryGuid;
3. DWORD verMajor;
4. GUID recGuid;
5. DWORD verMinor;
6. DWORD Lcid;
7. } RecordInfo;

**libraryGuid:**   MUST be set to a GUID that identifies the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) automation scope of the UDT (see section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)).

**verMajor:**   MUST be set to the major version of the UDT automation scope (see section 2.2.49.2).

**recGuid:**  MUST be set to the GUID of the UDT.

**verMinor:**   MUST be set to the minor version of the UDT's automation scope (see section 2.2.49.2).

**Lcid:**   MUST be set to the locale ID of the UDT's automation scope (see section 2.2.49.2).

RecordInfoData structures allow a client and a server to fully specify the identity of the UDT type being marshaled in the containing [BRECORD (section 2.2.28.2)](#Section_ea064b3d9fb3448699924fe2463e83e8). The client and the server MUST be able to reference the same type definition, by sharing through an unspecified mechanism a consistent view of the IDL automation scope of the UDT.[<13>](#Appendix_A_13" \o "Product behavior note 13)

Data of this type MUST be marshaled as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14, with the exception that it MUST be marshaled by using a little-endian data representation, regardless of the data representation format label. For more information, see [C706] section 14.2.5.

### DISPID

[DISPID](#Section_b0b43e39b0804edda26d7134075c75cd) is used by [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) to identify methods, properties, and [**named arguments**](#gt_0d91f93b-e04c-47d2-ae86-35d7e1bf382a).

This type is declared as follows:

1. typedef LONG DISPID;

The [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) values associated with methods, properties, and named arguments are retrieved as specified in section [3.1.4.3](#Section_7166d6ffb8514216bfaa34128274a242).

#### Reserved DISPIDs

The following [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) are defined by the OLE Automation Protocol, as specified in [[MS-OAUT]](%5bMS-OAUT%5d.pdf#Section_bbb05720f72445c78d17f83c3d1a3961). These DISPIDs are reserved and are meant to provide common, well-known DISPIDs associated with methods that have similar semantics across all [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) implementations. When an automation server needs to provide access to methods or properties that have specific semantics that map to one from the set below, they SHOULD use the DISPIDs specified in the following table.

| Constant/value | Description |
| --- | --- |
| DISPID\_VALUE  0 | This MUST designate the default member for the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The default member is the member that best represents the automation server. |
| DISPID\_UNKNOWN  -1 | The value MUST be returned by GetIDsOfNames (see section [3.1.4.3](#Section_7166d6ffb8514216bfaa34128274a242)) to indicate that a member or parameter name was not found. |
| DISPID\_PROPERTYPUT  -3 | This MUST designate the parameter that receives the value of an assignment in a DISPATCH PROPERTYPUT or DISPATCH PROPERTYPUTREF invocation (see section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)). |
| DISPID\_NEWENUM  -4 | This MUST designate the DISPID associated with a \_NewEnum method that MUST have the following signature.   1. HRESULT \_NewEnum([out,retval] IEnumVARIANT\*\* ppenum); 2. OR: 3. HRESULT \_NewEnum([out,retval] IUnknown\*\* ppenum);   For more information about IEnumVARIANT, see section [3.3](#Section_716d04d1cd1640659b191b8808b3df31). |

### DISPPARAMS

The DISPPARAMS structure is used by the Invoke method (see section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)) to contain the arguments passed to a method or property.

1. typedef struct tagDISPPARAMS {
2. [size\_is(cArgs)] VARIANT\* rgvarg;
3. [size\_is(cNamedArgs)] DISPID\* rgdispidNamedArgs;
4. UINT cArgs;
5. UINT cNamedArgs;
6. } DISPPARAMS;

**rgvarg:**  MUST be the array of arguments passed to the method or property call.

**rgdispidNamedArgs:**  MUST be the array of [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) corresponding to the [**named arguments**](#gt_0d91f93b-e04c-47d2-ae86-35d7e1bf382a) (see section 3.1.4.4).

**cArgs:**  MUST equal the number of arguments passed to the method.

**cNamedArgs:**  MUST equal the number of named arguments passed to the method. This value MUST be less than or equal to the value of **cArgs**.

The arguments passed in DISPPARAMS MUST be stored as specified in section [3.1.4.4.2](#Section_9cf379f7fb3141fe9f9cc9a0136616e0).

### EXCEPINFO

The EXCEPINFO structure is filled in by an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) to describe an exception that occurred during a call to Invoke (as specified in section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)). If no exception occurred, the [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) MUST set both wCode and scode to 0.

1. typedef struct tagEXCEPINFO {
2. WORD wCode;
3. WORD wReserved;
4. BSTR bstrSource;
5. BSTR bstrDescription;
6. BSTR bstrHelpFile;
7. DWORD dwHelpContext;
8. ULONG\_PTR pvReserved;
9. ULONG\_PTR pfnDeferredFillIn;
10. HRESULT scode;
11. } EXCEPINFO;

**wCode:**   An implementation-specific[<14>](#Appendix_A_14" \o "Product behavior note 14) value that identifies an error.

| Value | Meaning |
| --- | --- |
| 0 | The value MUST be zero for either of the following conditions:   * This field does not contain an error code. * The value in the **scode** field is nonzero. |
| 1000 < *value* | Implementation-specific error values MUST be greater than 1000. |

**wReserved:**  MUST be set to 0, and MUST be ignored on receipt.

**bstrSource:**  MUST[<15>](#Appendix_A_15" \o "Product behavior note 15) be set to an implementation-specific string that identifies the source of the exception.

**bstrDescription:**  MUST[<16>](#Appendix_A_16" \o "Product behavior note 16) be set to an implementation-specific string, or to a NULL BSTR if no description is available.

**bstrHelpFile:**  MUST[<17>](#Appendix_A_17" \o "Product behavior note 17) be set to an implementation-specific string, or to a NULL BSTR if no help is available.

**dwHelpContext:**  MUST[<18>](#Appendix_A_18" \o "Product behavior note 18) be set to an implementation-specific integer. If **bstrHelpFile** is NULL, this field MUST be set to 0, and MUST be ignored on receipt.

**pvReserved:**  MUST be set to NULL, and MUST be ignored on receipt.

**pfnDeferredFillIn:**  MAY be set to NULL, and MUST be ignored on receipt.[<19>](#Appendix_A_19" \o "Product behavior note 19)

**scode:**  MUST be set to a failure HRESULT that describes the error, or to 0 to indicate that it does not contain an error code. If **wCode** is nonzero, this field MUST be set to 0.

### MEMBERID

A MEMBERID type is a 32-bit value that identifies a data or method member of a type.

This type is declared as follows:

1. typedef DISPID MEMBERID;

The MEMBERID of an IDispatch interface MUST be the same as its [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474).

A type MUST NOT have more than one member with the same MEMBERID, unless the members are accessor methods for the same property. A property accessor methods with the same name and MEMBERID MUST specify different [INVOKEKIND](#Section_a0d3598da3ee440187fd17a7031b0b9a) constant values, as specified in section 2.2.14.

#### Reserved MEMBERIDs

The following [MEMBERIDs](#Section_ace8758fee2b4cb68645973994d12530) are defined by the OLE Automation Protocol. These MEMBERIDs are reserved and are meant to provide common, well-known MEMBERIDs that are associated with methods that have similar semantics across all [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) implementations. When an automation server needs to provide MEMBERID references that have specific semantics that map to one from the set below, they MUST use the MEMBERIDs specified in the following table.

| Constant/value | Description |
| --- | --- |
| MEMBERID\_NIL  -1 | This MUST designate the containing type in a context where it is also possible to refer to elements in its method or data member tables. |
| MEMBERID\_DEFAULTINST  -2 | This MUST designate an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) in a context where it is also possible to refer to members of its default nonsource interface (as specified in [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)). |

### HREFTYPE

An HREFTYPE is a 32-bit value that an [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) [**server**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) uses as a handle to associate a type that is defined or referenced in its [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) with an instance of an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) server.

This type is declared as follows:

1. typedef DWORD HREFTYPE;

### TYPEDESC

The TYPEDESC structure is used in the [ARRAYDESC](#Section_2e06e2b6054e48b1b867ad1e87a7ebe2), [ELEMDESC](#Section_e14ff3cf034a4884a498fc7586f7160c), and [TYPEATTR](#Section_0ca10d0861d2405991097bbaf545715e) structures to identify and describe the type of a data member, the return type of a method, or the type of a method parameter.

1. typedef struct tagTYPEDESC {
2. [switch\_type(USHORT), switch\_is(vt)]
3. union {
4. [case(VT\_PTR, VT\_SAFEARRAY)]
5. struct tagTYPEDESC\* lptdesc;
6. [case(VT\_CARRAY)]
7. struct tagARRAYDESC\* lpadesc;
8. [case(VT\_USERDEFINED)]
9. HREFTYPE hreftype;
10. [default]  ;
11. } \_tdUnion;
12. USHORT vt;
13. } TYPEDESC;

**\_tdUnion:**  MUST contain an instance of the type, according to the [VARENUM](#Section_3fe7db9f58034dc49d145425d3f5461f) value provided in the **vt** field.

**lptdesc:**   MUST refer to a TYPEDESC that specifies the element type. If the ELEMDESC is contained in a [VARDESC](#Section_ae7791d243994dffb7c6b0d4f3dce982) that describes an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc), the TYPEDESC MUST specify the type of the coclass.

**lpadesc:**   MUST refer to an ARRAYDESC that describes a fixed-length array.

**hreftype:**   MUST be set to an HREFTYPE that identifies the UDT (see section [2.2.28](#Section_29ce0a4f478649c9a3125522c1e9b44d)).

**vt:**  MUST be set to one of the values that are specified as available to a TYPEDESC and identified with a "T" in the Context column of the table in 2.2.7. MUST be set to VT\_PTR if the ELEMDESC is contained in a VARDESC that describes an appobject coclass, as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c).

### ARRAYDESC

The ARRAYDESC structure is used in a [TYPEDESC](#Section_95bb92a7f783477facbcc947d754fa8b) structure to specify the dimensions of an array and the type of its elements.

1. typedef struct tagARRAYDESC {
2. TYPEDESC tdescElem;
3. USHORT cDims;
4. [size\_is(cDims)] SAFEARRAYBOUND rgbounds[];
5. } ARRAYDESC;

**tdescElem:**  MUST contain a TYPEDESC that specifies the type of the elements in the array as specified in section 2.2.37.

**cDims:**  MUST be set to the number of dimensions in the array.

**rgbounds:**  MUST refer to a [SAFEARRAYBOUND](#Section_1941311d9b7d4a5bb2462d2eaad00f8c) that specifies the maximum index value for each dimension of the array, as specified in section 2.2.30.1.

### PARAMDESCEX

The PARAMDESCEX structure is used in a [PARAMDESC (section 2.2.40)](#Section_a965ce8e6c064d7cab302f14b1d8488a) structure to specify information about the default value of a parameter.

1. typedef struct tagPARAMDESCEX {
2. ULONG cBytes;
3. VARIANT varDefaultValue;
4. } PARAMDESCEX;

**cBytes:**  MUST be set to an implementation-specific value.[<20>](#Appendix_A_20" \o "Product behavior note 20)

**varDefaultValue:**  MUST contain a VARIANT that specifies the default value of the parameter.

### PARAMDESC

The PARAMDESC structure is used in an [ELEMDESC (section 2.2.41)](#Section_e14ff3cf034a4884a498fc7586f7160c) structure to specify the features of a method parameter.

1. typedef struct tagPARAMDESC {
2. PARAMDESCEX\* pparamdescex;
3. USHORT wParamFlags;
4. } PARAMDESC;

**pparamdescex:**  MUST refer to a [PARAMDESCEX](#Section_683c767d2e8e4d2f8804afeb3a73969a) structure that specifies the default value of the parameter if the PARAMFLAG\_FHASDEFAULT flag is set in the **wParamFlags** field. MUST be set to NULL otherwise.

**wParamFlags:**  MUST be set to a combination of the [PARAMFLAG (section 2.2.15)](#Section_4ca6f07bf89f469bba9e81fda041c8ac) bit flags if the PARAMDESC belongs to an element of the lprgelemdescParam array in a [FUNCDESC (section 2.2.42)](#Section_d3349d25e11d4095ba86de3fda178c4e) structure. MUST be set to 0 otherwise.

### ELEMDESC

The ELEMDESC structure is used in the [FUNCDESC (section 2.2.42)](#Section_d3349d25e11d4095ba86de3fda178c4e) and [VARDESC (section 2.2.43)](#Section_ae7791d243994dffb7c6b0d4f3dce982) structures to describe a member of a structure, a parameter, or the return value of a method.

1. typedef struct tagELEMDESC {
2. TYPEDESC tdesc;
3. PARAMDESC paramdesc;
4. } ELEMDESC;

**tdesc:**  MUST contain a [TYPEDESC (section 2.2.37)](#Section_95bb92a7f783477facbcc947d754fa8b) that describes the element, parameter, or return value.

**paramdesc:**  MUST contain a [PARAMDESC](#Section_a965ce8e6c064d7cab302f14b1d8488a) that has the values as specified in section 2.2.40, if the ELEMDESC is a member of the lprgelemdescParam array in a FUNCDESC (section 2.2.42) structure. Otherwise, it MUST contain a PARAMDESC that has the data fields pparamdescex and wParamFlags set to NULL and 0 respectively.

### FUNCDESC

The FUNCDESC structure is used by an [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) or [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) to describe a method, as specified in sections [3.5.4.1](#Section_476f00da080640d9bbf36059154abbb7) and [3.7.4.3](#Section_d54aca0905654fa8b5e4cf87723a89ed).

1. typedef struct tagFUNCDESC {
2. MEMBERID memid;
3. [size\_is(cReserved2)] SCODE\* lReserved1;
4. [size\_is(cParams)] ELEMDESC\* lprgelemdescParam;
5. FUNCKIND funckind;
6. INVOKEKIND invkind;
7. CALLCONV callconv;
8. SHORT cParams;
9. SHORT cParamsOpt;
10. SHORT oVft;
11. SHORT cReserved2;
12. ELEMDESC elemdescFunc;
13. WORD wFuncFlags;
14. } FUNCDESC,
15. \*LPFUNCDESC;

**memid:**  MUST be set to the [MEMBERID (section 2.2.35)](#Section_ace8758fee2b4cb68645973994d12530) of the method.

**lReserved1:**  MUST be set to 0 and ignored on receipt. An [**HRESULT**](#gt_799103ab-b3cb-4eab-8c55-322821b2b235) value is closely related, or identical to an [SCODE](#Section_7d93a429b6524c779b63d7a762521e4d).

**lprgelemdescParam:**  MUST refer to an array of [ELEMDESC](#Section_e14ff3cf034a4884a498fc7586f7160c) that contains one entry for each element in the method's parameter table.

The lprgelemdescParam array MUST NOT include parameters that are declared with the [lcid] or [retval] attributes if the value of funckind is FUNC\_DISPATCH (as specified in section [3.1.4.4.2](#Section_9cf379f7fb3141fe9f9cc9a0136616e0)).

**funckind:**  MUST be set to one of the values of the [FUNCKIND (section 2.2.12)](#Section_a33ebe360f9d4230bcbc466136f45e58) enumeration.

**invkind:**  MUST be set to one of the values of the [INVOKEKIND (section 2.2.14)](#Section_a0d3598da3ee440187fd17a7031b0b9a) enumeration.

**callconv:**  MUST be set to one of the values of the [CALLCONV (section 2.2.10)](#Section_2e3cc99094f241e5ae6fcdd00414b776) enumeration.

**cParams:**  MUST be set to the length of the lprgelemdescParam array.

**cParamsOpt:**  SHOULD be set to the number of optional VARIANT parameters[<21>](#Appendix_A_21" \o "Product behavior note 21). MUST be set to -1 if the method was declared with the [vararg] attribute. Otherwise, MUST be set to 0.

**oVft:**  MUST be set to either 0 or to the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) of the interface method multiplied by the system pointer size value (as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.11.1](#Section_87fd9a39606742a7b8e613637df3bd0d)).

**cReserved2:**  MUST be set to 0, and ignored on receipt.

**elemdescFunc:**  MUST contain an ELEMDESC that specifies the return type of the method, as specified in section 2.2.41.

**wFuncFlags:**  MUST be set to a combination of the [FUNCFLAGS](#Section_be8732b4f8d94e6da946311958d8173f) bit flags (as specified in section 2.2.11), or set to 0.

### VARDESC

The VARDESC structure is used by an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) or [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) to describe a data member, constant, or [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) property, as specified in sections [3.5.4.1](#Section_476f00da080640d9bbf36059154abbb7) and [3.7.4.4](#Section_a6b5857a38e446cda2e4bfbc7e21c787).

1. typedef struct tagVARDESC {
2. MEMBERID memid;
3. LPOLESTR lpstrReserved;
4. [switch\_type(VARKIND), switch\_is(varkind)]
5. union {
6. [case(VAR\_PERINSTANCE, VAR\_DISPATCH, VAR\_STATIC)]
7. ULONG oInst;
8. [case(VAR\_CONST)]
9. VARIANT\* lpvarValue;
10. } \_vdUnion;
11. ELEMDESC elemdescVar;
12. WORD wVarFlags;
13. VARKIND varkind;
14. } VARDESC,
15. \*LPVARDESC;

**memid:**  MUST be set to the [MEMBERID (section 2.2.35)](#Section_ace8758fee2b4cb68645973994d12530) of the data member, the constant, or the ODL dispinterface property. MUST be set to MEMBERID\_DEFAULTINST if the VARDESC describes an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc), as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c).

**lpstrReserved:**  MUST be set to NULL, and MUST be ignored by the recipient.

**\_vdUnion:**  MUST be set to an instance of the type, according to the value in the **varkind** field.

**oInst:**

* VAR\_PERINSTANCE: MUST be set to an implementation-specific value[<22>](#Appendix_A_22" \o "Product behavior note 22)
* VAR\_DISPATCH: MUST be set to 0.
* VAR\_STATIC: MUST be set to 0.

**lpVarValue:**  MUST be set to a reference to a [VARIANT](#Section_a6a540af38ac48bebd4092e2c01e9aa6) that specifies the value of the constant.

**elemdescVar:**  MUST contain an [ELEMDESC](#Section_e14ff3cf034a4884a498fc7586f7160c) that describes the data member, constant, or ODL dispinterface property and its type, as specified in section 2.2.41.

**wVarFlags:**  MUST be set to a combination of the [VARFLAGS](#Section_8ec5cfa4e710446aab896715dece4aec) bit flags (as specified in 2.2.18), or set to 0. MUST be set to 0 if the VARDESC describes an appobject coclass, as specified in section 2.2.49.8.

**varkind:**  MUST be set to a value of the [VARKIND](#Section_a0e9d46351a249cc8935a65c9338d3df) enumeration. MUST be set to VAR\_STATIC if the VARDESC describes an appobject coclass, as specified in section 2.2.49.8.

### TYPEATTR

The TYPEATTR structure is used by an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) to describe a type, as specified in section [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

1. typedef struct tagTYPEATTR {
2. GUID guid;
3. LCID lcid;
4. DWORD dwReserved1;
5. DWORD dwReserved2;
6. DWORD dwReserved3;
7. LPOLESTR lpstrReserved4;
8. ULONG cbSizeInstance;
9. TYPEKIND typekind;
10. WORD cFuncs;
11. WORD cVars;
12. WORD cImplTypes;
13. WORD cbSizeVft;
14. WORD cbAlignment;
15. WORD wTypeFlags;
16. WORD wMajorVerNum;
17. WORD wMinorVerNum;
18. TYPEDESC tdescAlias;
19. DWORD dwReserved5;
20. WORD wReserved6;
21. } TYPEATTR,
22. \*LPTYPEATTR;

**guid:**  MUST be set to the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that is associated with the type, or to IID\_NULL, if the type was not declared with the [uuid] attribute (see section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)).

**lcid:**  MUST be set to the locale ID of the type's member names and documentation strings (see section 2.2.49.2).

**dwReserved1:**  MUST be set to 0, and MUST be ignored on receipt.

**dwReserved2:**  MUST be set to -1, and MUST be ignored on receipt.

**dwReserved3:**  MUST be set to -1, and MUST be ignored on receipt.

**lpstrReserved4:**  MUST be set to NULL, and MUST be ignored on receipt.

**cbSizeInstance:**  MUST be set to a value that is specified by the value of **typekind**.

| Value of typekind | Value of cbSizeInstance |
| --- | --- |
| TKIND\_COCLASS | MUST be set to the system pointer size (see section [3.7.1.2](#Section_7b1b8bd1a0674edb9d726aa500d035a3)). |
| TKIND\_DISPATCH | MUST be set to the system pointer size (see section 3.7.1.2). |
| TKIND\_INTERFACE | MUST be set to the system pointer size (see section 3.7.1.2). |
| TKIND\_MODULE | MUST be set to 2. |
| TKIND\_ENUM | MUST be set to an implementation-specific value[<23>](#Appendix_A_23" \o "Product behavior note 23) that specifies the size of an integer. |
| TKIND\_UNION | MUST be set to an implementation-specific value[<24>](#Appendix_A_24" \o "Product behavior note 24) that specifies the size of its largest element. |
| TKIND\_RECORD | MUST be set to an implementation-specific value[<25>](#Appendix_A_25" \o "Product behavior note 25) that specifies the size in bytes, of the structure. |
| TKIND\_ALIAS | MUST be set to an implementation-specific value[<26>](#Appendix_A_26" \o "Product behavior note 26) that specifies the size, in bytes, of the predefined type for which this type is an alias. |

**typekind:**  MUST be set to a value of the [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33) enumeration, as specified in section 2.2.17.

**cFuncs:**  MUST be set to a value specified by the value of **typekind**.

| Value of typekind | Value of cfuncs |
| --- | --- |
| TKIND\_COCLASS | MUST be set to 0. |
| TKIND\_DISPATCH | MUST be set to the number of elements in the dispatch method table, as specified in section 3.7.1.2. |
| TKIND\_INTERFACE | MUST be set to the number of elements in the method table, as specified in section 3.7.1.2. |
| TKIND\_MODULE | MUST be set to the number of elements in the method table, as specified in section 3.7.1.2. |
| TKIND\_ENUM | MUST be set to 0. |
| TKIND\_UNION | MUST be set to 0. |
| TKIND\_RECORD | MUST be set to 0. |
| TKIND\_ALIAS | MUST be set to 0. |

**cVars:**  MUST be set to the number of elements in the data member table, as specified in section 3.7.1.2.

**cImplTypes:**  MUST be set to the number of elements in the interface table, as specified in section 3.7.1.2.

**cbSizeVft:**  MUST be set to a value specified by the value of **typekind**.

| Value of typekind | Value of cbSizeVft |
| --- | --- |
| TKIND\_COCLASS | MUST be set to 0. |
| TKIND\_DISPATCH | MUST be set to the system pointer size value (see section [2.2.45](#Section_b568f4be95e5431bbb3b08dc56e9b224)) multiplied by seven. |
| TKIND\_INTERFACE | MUST be set to the system pointer size value multiplied by the number of methods that are defined by the interface and all its inherited interfaces. |
| TKIND\_MODULE | MUST be set to 0. |
| TKIND\_ENUM | MUST be set to 0. |
| TKIND\_UNION | MUST be set to 0. |
| TKIND\_RECORD | MUST be set to 0. |
| TKIND\_ALIAS | MUST be set to 0. |

**cbAlignment:**  MUST be set to 0 or to an implementation-specific positive value.[<27>](#Appendix_A_27" \o "Product behavior note 27)

**wTypeFlags:**  MUST be either a combination of the [TYPEFLAGS](#Section_155c66e2ffe14f18b849f827ca989aa7) bit flags that are specified in section 2.2.16, or 0.

**wMajorVerNum:**  MUST be set to the major version number of the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) that is associated with the ITypeLib server, as specified in section 2.2.49.2.

**wMinorVerNum:**  MUST be set to the minor version number of the automation scope that is associated with the ITypeLib server, as specified in section 2.2.49.2.

**tdescAlias:**  MUST contain a [TYPEDESC (section 2.2.37)](#Section_95bb92a7f783477facbcc947d754fa8b) that describes the predefined type for which this type is an alias, if **typekind** is set to TKIND\_ALIAS. Otherwise, MUST contain a TYPEDESC with the **vt** field set to VT\_EMPTY.

**dwReserved5:**  MUST be set to 0, and MUST be ignored on receipt.

**wReserved6:**  MUST be set to 0, and MUST be ignored on receipt.

### TLIBATTR

The TLIBATTR structure is used to specify the attributes of an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b), as specified in section [3.11.4](#Section_a1436b20e676495ab4f39e9251a40e7b).

1. typedef struct tagTLIBATTR {
2. GUID guid;
3. LCID lcid;
4. SYSKIND syskind;
5. unsigned short wMajorVerNum;
6. unsigned short wMinorVerNum;
7. unsigned short wLibFlags;
8. } TLIBATTR,
9. \*LPTLIBATTR;

**guid:**  MUST be set to the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) of the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) that is associated with the ITypeLib server, as specified in section [2.2.49.1](#Section_b64411f397674fa3a62faa35d2209cc1).

**lcid:**  MUST be set to the [**LCID**](#gt_c7f99c66-592f-4053-b62a-878c189653b6) of the automation scope that is associated with the ITypeLib server, as specified in section 2.2.49.1.

**syskind:**  MUST be set to a value of the [SYSKIND](#Section_0d81289ef0ef474d8e61dedae9ea5a08) enumeration, as specified in section 2.2.21.

The value of syskind specifies the system pointer-size value. If syskind is SYS\_WIN32, the system pointer-size value is 4. If syskind is SYS\_WIN64, the system pointer-size value is 8.

The system pointer-size value MUST be the size, in bytes, of the VT\_INT\_PTR and VT\_UINT\_PTR type variables created by the server (see section [2.2.7](#Section_3fe7db9f58034dc49d145425d3f5461f)). It is used as a multiplier in the **oVft** field of a [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e) (see section 2.2.42) and in the **cbSizeVft** field of a [TYPEATTR](#Section_0ca10d0861d2405991097bbaf545715e) (see section 2.2.44).

**wMajorVerNum:**  MUST be set to the major version number of the automation scope that is associated with the ITypeLib server, as specified in section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4).

**wMinorVerNum:**  MUST be set to the minor version number of the automation scope that is associated with the ITypeLib server, as specified in section 2.2.49.2.

**wLibFlags:**  MUST be either a combination of the LIBFLAGS bit flags (as specified in section [2.2.20](#Section_08ca1c9a5ac54630aeafb09d495640b1)) or 0.

### CUSTDATAITEM

The CUSTDATAITEM structure is used in a [CUSTDATA](#Section_b74500e231534cc6bebf9e11320f7bed) structure to store custom data items, as specified in section 2.2.47.

1. typedef struct tagCUSTDATAITEM {
2. GUID guid;
3. VARIANT varValue;
4. } CUSTDATAITEM;

**guid:**  MUST be set to the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item that uses the [custom] attribute, as specified in section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f).

**varValue:**  MUST be set to the value of the custom data item.

### CUSTDATA

The CUSTDATA structure is used by an [ITypeInfo2 server](#Section_2d6024dad2294d78bbb0b9d5bf6459b7) or [ITypeLib2 server](#Section_4bb9bc733cf540a185c7aafaff4874cc) to retrieve custom data items, as specified in sections [3.9.4](#Section_dc0a9d195bc34fedb56aba2424379d33) and [3.13.4](#Section_64099fb5ded14584a82725af7a5f2b80).

1. typedef struct tagCUSTDATA {
2. DWORD cCustData;
3. [size\_is(cCustData)] CUSTDATAITEM\* prgCustData;
4. } CUSTDATA;

**cCustData:**  MUST be set to the number of custom data items in **prgCustData**.

**prgCustData:**  MUST refer to an array of [CUSTDATAITEM](#Section_02ca19b927cb48efb2ca7f105ba8f475) structures that contain custom data items, as specified in section 2.2.46.

### SCODE

The SCODE data type is a 32-bit status value that is used to describe an error or warning.

This type is declared as follows:

1. typedef LONG SCODE;

On 32-bit platforms, the **SCODE** data type is the same as the HRESULT data type. On 16-bit platforms, an **SCODE** value is used to generate an HRESULT value.

For more information, see [[SCODE]](https://go.microsoft.com/fwlink/?LinkId=90511).

### IDL Syntax Extensions

Automation provides a number of [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) extensions that support a seamless integration of [**automation servers**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) with generic [**automation clients**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28), scripting languages, and various development environments.[<28>](#Appendix_A_28" \o "Product behavior note 28) A [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) that uses [**automation type descriptions**](#gt_fb6a1829-c102-482c-902f-51c197b22860) to examine the functionality provided by an [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) or an automation server is an [**automation type browser**](#gt_e5259d30-22bf-4ea1-9cf7-befbeba9bab5).

This section specifies the [**automation types**](#gt_30a4192b-9daa-4a21-bd87-6cb0908a2a9e) and the IDL attributes and statements that have an impact on the wire communication between an automation client and server. It also specifies the attributes and statements that allow a client to discover rich type information regarding the automation servers that it is calling.

The areas covered are:

* COM server categories and behaviors that can be described by the IDL extensions: [**aggregatable servers**](#gt_cb1c7858-f49b-4030-9311-479026cbe832), connectable servers, and bindable servers.
* Automation-compatible types, which are relevant to marshaling the arguments as part of the call to IDispatch::Invoke (see section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)).
* Automation library scope, which provides context for marshaling UDTs (see sections [2.2.28](#Section_29ce0a4f478649c9a3125522c1e9b44d) and [2.2.31](#Section_deb939dfef4d49c384677265669e89ed)).
* Automation [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474), which provide the information needed for performing an automation call through IDispatch::Invoke (see section 3.1.4.4).
* Automation attributes and statements that specify a rich set of type information regarding an automation server (see sections [3.5](#Section_7894019fde1e455eb2aa3b899c2e50f6) through [3.14](#Section_8619681d434c41e38a9fe17493230ba4)).

The extensions to the IDL that are specified by the OLE Automation Protocol are derived from the now-obsolete Object Definition Language (ODL). The extensions take the following forms.

* A set of attributes that specify additional semantic meaning for the language element they decorate: version, lcid, oleautomation, dual, id, propget, propput, propputref, readonly, defaultvalue, optional, vararg, and retval.
* A set of statements that allow for additional information to be specified, or for an alternative way to define language elements. The statements are introduced by the following keywords: library, [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5), methods, and properties.
* A set of attributes that specify a rich group of properties of interest to the automation client: control, source, default, defaultvtable, bindable, defaultbind, immediatebind, displaybind, requestedit, public, uidefault, restricted, hidden, nonbrowsable, helpcontext, helpfile, helpstring, helpstringcontext, helpstringdll, appobject, predeclid, aggregatable, defaultcollelem, licensed, proxy, noncreatable, nonextensible, custom, and replaceable.
* A statement that allows specifying an automation server [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc).

Extensions to the IDL syntax that support the OLE Automation Protocol are specified by using the Augmented Backus-Naur Form (ABNF) notation, as specified in [[RFC4234]](https://go.microsoft.com/fwlink/?LinkId=90462) section 2.2.

The DCE 1.1: Remote Procedure Call, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824), specifies the syntax for IDL by using an extended BNF notation. The following table specifies the DCE 1.1: Remote Procedure Call (as specified in [C706]), the production names referenced later, and the corresponding ABNF name as it will be used in this specification.

| DCE 1.1: Remote Procedure Call production name | Equivalent ABNF production name |
| --- | --- |
| <Uuid\_rep> | uuid-rep |
| <integer\_const\_exp> | integer-const-exp |
| <param\_attribute> | param-attribute |
| <const\_exp> | const-exp |
| <type\_attribute> | type-attribute |
| <param\_attribute> | param-attribute |
| <operation\_attributes> | operation-attributes |
| <op\_declarator> | op-declarator |
| <interface\_attribute> | interface-attribute |
| <interface\_attributes> | interface-attributes |
| <interface> | interface |
| <import> | import |
| <export> | export |
| <string> | string |
| <param\_declarators> | param-declarators |

In addition, the productions use LWSP as the linear whitespace production rpcidl-defined for any production defined in [C706] section 4, and "kw-KEYWORD" as the production for the case-sensitive keyword **KEYWORD**:

1. kw-aggregatable = %d97.103.103.114.101.103.97.116.97.98.108.101
2. kw-appobject = %d97.112.112.111.98.106.101.99.116
3. kw-bindable = %d98.105.110.100.97.98.108.101
4. kw-boolean = %d98.111.111.108.101.97.110
5. kw-BSTR = %d66.83.84.82
6. kw-cdecl = %d99.100.101.99.108
7. kw-char = %d99.104.97.114
8. kw-coclass = %d99.111.99.108.97.115.115
9. kw-const = %d99.111.110.115.116
10. kw-control = %d99.111.110.116.114.111.108
11. kw-CURRENCY = %d67.85.82.82.69.78.67.89
12. kw-custom = %d99.117.115.116.111.109
13. kw-DATE = %d68.65.84.69
14. kw-Decimal = %d68.101.99.105.109.97.108
15. kw-default = %d100.101.102.97.117.108.116
16. kw-defaultbind = %d100.101.102.97.117.108.116.98.105.110.100
17. kw-defaultcollelem =
18. %d100.101.102.97.117.108.116.99.111.108.108.101.108.101.109
19. kw-defaultvalue =
20. %d100.101.102.97.117.108.116.118.97.108.117.101
21. kw-defaultvtable =
22. %d100.101.102.97.117.108.116.118.116.97.98.108.101
23. kw-dispinterface =
24. %d100.105.115.112.105.110.116.101.114.102.97.99.101
25. kw-displaybind =
26. %d100.105.115.112.108.97.121.98.105.110.100
27. kw-dllname = %d100.108.108.110.97.109.101
28. kw-double = %d100.111.117.98.108.101
29. kw-dual = %d100.117.97.108
30. kw-entry = %d101.110.116.114.121
31. kw-float = %d102.108.111.97.116
32. kw-helpcontext = %d104.101.108.112.99.111.110.116.101.120.116
33. kw-helpfile = %d104.101.108.112.102.105.108.101
34. kw-helpstring = %d104.101.108.112.115.116.114.105.110.103
35. kw-helpstringcontext =
36. %d104.101.108.112.115.116.114.105.110.103.99.111.110.116.101.120.116
37. kw-helpstringdll =
38. %d104.101.108.112.115.116.114.105.110.103.100.108.108
39. kw-hidden = %d104.105.100.100.101.110
40. kw-id = %d105.100
41. kw-immediatebind =
42. %d105.109.109.101.100.105.97.116.101.98.105.110.100
43. kw-importlib = %d105.109.112.111.114.116.108.105.98
44. kw-int = %d105.110.116
45. kw-interface = %d105.110.116.101.114.102.97.99.101
46. kw-lcid = %d108.99.105.100
47. kw-library = %d108.105.98.114.97.114.121
48. kw-licensed = %d108.105.99.101.110.115.101.100
49. kw-long = %d108.111.110.103
50. kw-methods = %d109.101.116.104.111.100.115
51. kw-module = %d109.111.100.117.108.101
52. kw-nonbrowsable = %d110.111.110.98.114.111.119.115.97.98.108.101
53. kw-noncreatable = %d110.111.110.99.114.101.97.116.97.98.108.101
54. kw-nonextensible =
55. %d110.111.110.101.120.116.101.110.115.105.98.108.101
56. kw-oleautomation =
57. %d111.108.101.97.117.116.111.109.97.116.105.111.110
58. kw-optional = %d111.112.116.105.111.110.97.108
59. kw-pascal = %d112.97.115.99.97.108
60. kw-predeclid = %d112.114.101.100.101.99.108.105.100
61. kw-properties = %d112.114.111.112.101.114.116.105.101.115
62. kw-propget = %d112.114.111.112.103.101.116
63. kw-propput = %d112.114.111.112.112.117.116
64. kw-propputref = %d112.114.111.112.112.117.116.114.101.102
65. kw-proxy = %d112.114.111.120.121
66. kw-public = %d112.117.98.108.105.99
67. kw-readonly = %d114.101.97.100.111.110.108.121
68. kw-replaceable = %d114.101.112.108.97.99.101.97.98.108.101
69. kw-requestedit = %d114.101.113.117.101.115.116.101.100.105.116
70. kw-restricted = %d114.101.115.116.114.105.99.116.101.100
71. kw-retval = %d114.101.116.118.97.108
72. kw-SAFEARRAY = %d83.65.70.69.65.82.82.65.89
73. kw-SCODE = %d83.67.79.68.69
74. kw-short = %d115.104.111.114.116
75. kw-source = %d115.111.117.114.99.101
76. kw-static = %d115.116.97.116.105.99
77. kw-stdcall = %d115.116.100.99.97.108.108
78. kw-uidefault = %d117.105.100.101.102.97.117.108.116
79. kw-unsigned = %d117.110.115.105.103.110.101.100
80. kw-uuid = %d117.117.105.100
81. kw-usesgetlasterror =
82. %d117.115.101.115.103.101.116.108.97.115.116.101.114.114.111.114
83. kw-vararg = %d118.97.114.97.114.103
84. kw-version = %d118.101.114.115.105.111.110

For the full ABNF specification of the extensions provided in this section, see [Appendix C](#Section_2b2b5513b36a41b7a114e955f80332b3).

#### COM Server Categories

The following are particular [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) categories that can be specified using the automation [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) extensions described in this section. Clients can use [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) interfaces to identify servers that advertise their capabilities and to communicate with them according to the categories they publicly support.

##### Aggregatable Servers

An [**aggregatable server**](#gt_cb1c7858-f49b-4030-9311-479026cbe832) is a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that can be contained by another COM server and that allows its interfaces to be used as if they were defined by the containing server.

##### Connectable Servers

A [**connectable server**](#gt_106828f1-1bf9-4cdb-9d12-d48b52495114) is a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that enables bidirectional communication with clients. Server-initiated communication MAY be specified using source interfaces (as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)) or bindable properties (as specified in section [2.2.49.1.3](#Section_2a89365308e7415a922628f8767c16ba)).

The automation IDL extensions allow a connectable server to define source interfaces for communicating with clients. If the server does not define source interfaces, server-initiated communication is limited to notifications related to bindable properties.

A connectable server provides the following functionality:[<29>](#Appendix_A_29" \o "Product behavior note 29)

* Identifies itself as a connectable server.
* Enables its clients to enumerate its available source interfaces.
* Enables its clients to register and unregister a client-implemented interface with any of its available [**source interfaces**](#gt_1036b31f-5af0-4298-a0e2-e3bf9933cd77).
* Enables its clients to enumerate the registered connections for each available source interface.

A [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) of the connectable server MUST implement one or more of the source interfaces that are specified by the server.

A connectable server MUST have one or more interfaces that are declared with the [source] attribute (as specified in section 2.2.49.8) or one or more properties that are declared with the [bindable] attribute (as specified in section [2.2.49.5.2](#Section_ac18004e7af74feb8ae6d42e7a14267c)).

##### Bindable Servers

A [**bindable server**](#gt_7d037fdd-af75-47fe-a235-d1bb937aa424) extends the functionality of a [**connectable server**](#gt_106828f1-1bf9-4cdb-9d12-d48b52495114). It associates clients with bindable properties and notifies the clients whenever the value of the property is changed.

A bindable server provides the same functionality as a connectable server (see section [2.2.49.1.2](#Section_4b96268c74984be6b20ed9220eb0057e)) and MUST have one or more properties declared with the [bindable] attribute (see section [2.2.49.5.2](#Section_ac18004e7af74feb8ae6d42e7a14267c)).

A [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) for a bindable server MUST provide the following functionality:[<30>](#Appendix_A_30" \o "Product behavior note 30)

* The client MUST receive notifications that the value of the associated property has changed.
* The client MUST specify whether it provides the functionality to determine whether a property value can be changed. If it does provide this functionality, it either grants or denies permission to change the value of the property upon request.

When the value of a bindable property is changed, the server notifies each registered client that the value of the property was changed.

#### IDL Automation Scope

An [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) is defined by the **library** keyword, as in the following.

1. oa-scope =
2. oa-library-header LWSP "{" oa-library-body "}" LWSP [";"]
3. oa-library-header =
4. "[" LWSP library-attributes LWSP "]"
5. LWSP kw-library LWSP Identifier
6. oa-library-body = \*oa-library-declarator
8. library-attributes =
9. library-attribute \*( "," LWSP library-attribute LWSP )
10. library-attribute = uuid-attr /
11. version-attr /
12. lcid-attr /
13. help-attr /
14. custom-attr /
15. kw-control /
16. kw-hidden /
17. kw-restricted
18. uuid-attr = kw-uuid LWSP "(" LWSP uuid-rep LWSP ")"
19. version-attr =
20. kw-version LWSP "(" LWSP 1\*DIGIT \*( "." 1\*DIGIT ) LWSP ")"
21. lcid-attr = kw-lcid LWSP "(" LWSP integer-const-exp LWSP ")"
22. help-attr = helpcontext-attr /
23. helpfile-attr /
24. helpstring-attr /
25. helpstringcontext-attr /
26. helpstringdll-attr
27. helpcontext-attr =
28. kw-helpcontext LWSP "(" LWSP integer-const-exp LWSP ")"
29. helpfile-attr = kw-helpfile LWSP "(" LWSP string LWSP ")"
30. helpstring-attr = kw-helpstring LWSP "(" LWSP string LWSP ")"
31. helpstringcontext-attr = kw-helpstringcontext LWSP "("
32. LWSP integer-const-exp LWSP ")"
33. helpstringdll-attr = kw-helpstringdll LWSP "(" LWSP string LWSP ")"
34. custom-attr = kw-custom LWSP
35. "(" uuid-rep LWSP "," LWSP const-exp LWSP ")"
36. oa-library-declarator = interface /
37. import /
38. export /
39. oa-importlib /
40. oa-module /
41. oa-dispinterface /
42. oa-coclass

* **oa-scope:** Specifies a new automation scope. There MUST be, at most, one automation scope defined in an IDL file.
* **uuid-attr:** Specifies a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that MUST identify the automation scope. This attribute MUST be present in the library scope.
* **version-attr:** Specifies the version of the automation scope. If this attribute is not specified, a version of 0.0 MUST be assumed for the automation scope. Otherwise, it MUST contain a major version that MUST be a decimal number between 0 and 65535 inclusive. If the minor version is missing, it MUST be treated as 0. Otherwise, it MUST be a decimal number between 0 and 65535 inclusive.

**lcid-attr:** Specifies the locale ID of the automation scope. If this attribute is not specified, the locale ID of 0x0409 MUST be used for the automation scope. Otherwise, this value MUST resolve to a valid locale ID.

The combination (<guid>, <vMajor>, <vMinor>, <lcidValue>) MUST uniquely identify an automation scope.

The <guid> value is the main component of the automation scope identity. Scopes with identical <guid> values MUST belong to the same [**automation scope family**](#gt_79854f42-4476-4d1f-9dad-944a7c81e4fb).

The <vMajor> and <vMinor> values MUST be used to specify different versions of an automation scope. automation scopes from the same automation scope family that also share the same version numbers MUST belong to the same [**automation scope generation**](#gt_2b5b8f9e-3e78-4572-9a03-525a264df2d8).

The <lcidValue> MUST be used to define multiple automation scopes within an automation scope generation. Two such automation scopes MUST define the same [**automation interfaces**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced), and the interfaces MUST differ only in the names that are used for methods, properties, and parameter names (see sections [2.2.49.5](#Section_da55c4194395453582c4bac998dae862) and [2.2.49.6](#Section_ff2bd74bcb4d48b3ab896bb32cda3833)). The equivalent names from the two automation scopes MUST map to the same [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474). This enables an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) to define multiple mappings from method/property names to a determined set of DISPIDs, with one mapping for each supported locale ID.

**helpcontext-attr**: Specifies an implementation-specific integer.[<31>](#Appendix_A_31" \o "Product behavior note 31) The value of this attribute MUST be a 32-bit integer. When used on nonlibrary elements, the language element it decorates MUST belong to an automation scope that is declared with the helpfile attribute.

**helpfile-attr**: Specifies an implementation-specific string.[<32>](#Appendix_A_32" \o "Product behavior note 32)

**helpstring-attr**: Specifies an implementation-specific string.[<33>](#Appendix_A_33" \o "Product behavior note 33)

**helpstringcontext-attr**: Specifies an implementation-specific integer.[<34>](#Appendix_A_34" \o "Product behavior note 34) The value of this attribute MUST be a 32-bit integer.

**helpstringdll-attr**: Specifies an implementation-specific string.[<35>](#Appendix_A_35" \o "Product behavior note 35)

**uuid-attr**: Specifies a GUID that MUST identify the type. Any Automation-compatible constructed types (enums, structs, or unions) that are not declared with a specifying GUID cannot be retrieved by using ITypeLib::GetTypeInfoOfGuid.

**kw-control**: Specifies that all COM servers in the automation scope are visual controls. A type browser [**client**](#gt_60e0e1fa-66fe-41e1-b5e3-ceab97e53506) can limit the visibility of elements that have this attribute.

**kw-hidden**: Specifies that the automation scope elements are not intended to be displayed to users. Type browser clients SHOULD NOT expose the functionality of elements with this attribute.

**kw-restricted**: Specifies that the element is not intended to be used under all conditions. Type browser clients MAY[<36>](#Appendix_A_36" \o "Product behavior note 36) place restrictions on the visibility or usability of elements that have this attribute. Elements that have the **[restricted]** attribute MUST NOT also have the **[default]** attribute.

**help-attr**: Specifies information associated with language elements that can be retrieved by using ITypeLib::GetDocumentation or ITypeInfo::GetDocumentation (as specified in section [3.11.4.7](#Section_ceb2d9eb975a47019a793bb9e6ad419b)).

**custom-attr**: Specifies that the attribute is user-defined, and that its meaning depends on its associated GUID.

Custom attributes are optional. If there is more than one custom attribute decorating an automation scope, each one MUST have a different identifying GUID.

The constant value associated with a [custom] attribute MUST be a value that can be stored in a \_wireVARIANT, as specified in section [2.2.29.2](#Section_a6a540af38ac48bebd4092e2c01e9aa6).

#### Automation-Compatible Types

The OLE Automation Protocol restricts the types of parameters that can be used in the definition of an [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) (see section [2.2.49.4](#Section_3b4b512c8c9445a1810dba4ff0152698)).

1. type-attribute = rpc-defined /
2. uuid-attr /
3. help-attr /
4. custom-attr /
5. kw-public /
6. kw-restricted
7. oa-type-spec = oa-base-type-spec /
8. oa-safearray-type-spec /
9. oa-ptr-type-spec /
10. Identifier
11. oa-base-type-spec = oa-base-nondecimal-type-spec / kw-Decimal
12. oa-base-nondecimal-type-spec = kw-boolean /
13. [kw-unsigned] LWSP kw-char /
14. [kw-unsigned] LWSP kw-short /
15. [kw-unsigned] LWSP kw-int /
16. [kw-unsigned] LWSP kw-long /
17. kw-double /
18. kw-float /
19. kw-BSTR /
20. kw-CURRENCY /
21. kw-DATE /
22. kw-SCODE
23. oa-safearray-type-spec =
24. kw-SAFEARRAY "(" oa-base-nondecimal-type-spec ")" /
25. kw-SAFEARRAY "(" oa-ptr-type-spec ")" /
26. kw-SAFEARRAY "(" Identifier ")"
27. oa-ptr-type-spec = oa-base-nondecimal-type-spec "\*" /
28. oa-safearray-type-spec "\*" /
29. Identifier "\*"

In the preceding productions, <Identifier> MUST be a type that is defined as an enumeration type, a UDT, or an automation-compatible interface (see section [2.2.49.4.1](#Section_222fe935db004920a7d9583a6b26c45b) for details).

The restriction that defines the automation-compatible types is that they MUST be representable as a field in a [VARIANT (section 2.2.29.2)](#Section_a6a540af38ac48bebd4092e2c01e9aa6).

The following table specifies the mapping between the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) keyword and the associated VARIANT type constant (see section [2.2.7](#Section_3fe7db9f58034dc49d145425d3f5461f)) that MUST[<37>](#Appendix_A_37" \o "Product behavior note 37) be used when storing the type in a VARIANT. These are the only types that are legal in an automation-compatible interface (see section 2.2.49.4.1).

| Type keyword | VARIANT type constant |
| --- | --- |
| boolean | VT\_BOOL |
| unsigned char | VT\_UI1 |
| char | VT\_I1 |
| double | VT\_R8 |
| float | VT\_R4 |
| unsigned int | VT\_UI4 |
| int | VT\_I4 |
| unsigned long | VT\_UI4 |
| long | VT\_I4 |
| unsigned short | VT\_UI2 |
| short | VT\_I2 |
| BSTR | VT\_BSTR |
| CURRENCY | VT\_CY |
| VARIANT | VT\_BYREF|VT\_VARIANT |
| DATE | VT\_DATE |
| Decimal | VT\_DECIMAL |
| SCODE | VT\_ERROR |
| HRESULT | VT\_ERROR |
| typedef enum myenum | VT\_I4 |
| typedef struct myUDT | VT\_RECORD or  VT\_BYREF|VT\_RECORD |
| interface IDispatch\* | VT\_DISPATCH |
| interface IUnknown\* | VT\_UNKNOWN |
| dispinterface Typename\* | VT\_DISPATCH |
| [oleautomation] interface Typename\* | VT\_UNKNOWN |
| SAFEARRAY(Typename) | VT\_ARRAY |
| Typename\* | VT\_BYREF | the variant type associated with Typename |
| [dual] interface Typename\* | VT\_DISPATCH |

The methods of an [**Automation Interface Definition Language (AIDL) interface**](#gt_13017420-f6a8-4150-9549-b0d754dbb128) MUST return an HRESULT or SCODE. The methods and properties of an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) MUST return only the types that are specified in the previous table. Otherwise, an automation interface MUST use types from the previous table. See [AIDL Interfaces and ODL Dispinterfaces (section 2.2.49.7)](#Section_18d74e75f9a744079fe83406679f7dd8) for more information.

**uuid-attr**: Specifies a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that MUST identify the type. Any automation-compatible constructed types (enums, structs, or unions) that are not declared with a specifying GUID cannot be retrieved using [ITypeLib::GetTypeInfoOfGuid (section 3.11.4.4)](#Section_01dd3fef481b4957b540baa469cbc3a7).

**help-attr**: Specifies information associated with the type that can be retrieved using [ITypeLib::GetDocumentation (section 3.11.4.7)](#Section_ceb2d9eb975a47019a793bb9e6ad419b) or [ITypeInfo::GetDocumentation (section 3.7.4.8)](#Section_2ea2f705bc334cecbbc7613d6ae0f0c6).

**kw-public**: Specifies that the element is an alias declared with the typedef keyword and explicitly included in an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a).

**kw-restricted**: Specifies that the element is not intended to be used under all conditions. Type browser clients MAY[<38>](#Appendix_A_38" \o "Product behavior note 38) place restrictions on the visibility or usability of elements that have this attribute. Elements with the **[restricted]** attribute MUST NOT also have the **[default]** attribute.

**kw-hidden**: Specifies that the type is not intended to be displayed to users. Type browser clients SHOULD NOT expose the functionality of elements that have this attribute.

#### Automation Interfaces

The OLE Automation Protocol allows interfaces to specify two degrees of Automation support:

* Automation-compatible interfaces are interfaces whose methods use only automation-compatible types.
* [**automation interfaces**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) are interfaces that define an automation behavior. That is, servers that implement the automation interfaces MUST provide access to the automation functionality through an implementation of IDispatch (see section [3.1.4](#Section_ac9c502bac1c42028ad4048ac98afcc9)).

In order for an interface to be used as an automation-compatible type (see section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e)), it MUST be defined as an automation-compatible interface.

All automation interfaces MUST be Automation-compatible.

The following attributes apply to interfaces defined in an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a):

1. interface-attribute = rpcidl-defined /
2. kw-nonextensible /
3. kw-proxy /
4. attr-oleautomation /
5. attr-dual /
6. custom-attr

**kw-nonextensible**: Specifies that an interface or [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) implements IDispatch, but includes only the properties and methods listed in the interface description. Interfaces and dispinterfaces with this attribute MUST NOT be extended with additional members at run time.

**kw-proxy**: Specifies an implementation-specific local behavior of the interface [<39>](#Appendix_A_39" \o "Product behavior note 39). This attribute has no effect across the wire.

##### Automation-Compatible Interfaces

To define an interface as an automation-compatible interface, the **oleautomation** attribute MUST be specified when defining the interface.

The following production extends the <interface\_attribute> production from [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824):

1. attr-oleautomation = kw-oleautomation

An automation-compatible interface MUST be derived from IDispatch or IUnknown, MUST have the [**oleautomation** attribute, and all of its methods MUST have only automation-compatible parameters and return types.

##### Dual Interfaces

A [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) is an [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) that allows a server to expose its functionality both to [**automation clients**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) and to regular DCOM clients. A dual interface MUST derive from IDispatch (see section [3.1.4](#Section_ac9c502bac1c42028ad4048ac98afcc9)), and be Automation compatible (that is, its methods MUST have only automation-compatible parameters and return values).

To define an interface as a dual interface, the **dual** attribute MUST be specified when defining the interface.

The following production extends the <interface\_attribute> production from [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824):

1. attr-dual = kw-dual

##### Dispinterface Interfaces

A [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) is an [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) that specifies the properties and methods (see section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862)) that the IDispatch implementation of the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) MUST implement.

A dispinterface is specified as:

1. oa-dispinterface =
2. oa-dispitf-header LWSP "{" LWSP oa-dispitf-body LWSP "}"
3. oa-dispitf-header = "[" interface-attributes "]"
4. LWSP kw-dispinterface LWSP Identifier
5. oa-dispitf-body = oa-itf-ref / oa-odl-body
6. oa-itf-ref = kw-interface LWSP Identifier LWSP ";"

A dispinterface defined using the <oa-itf-ref> production is a [**reference dispinterface**](#gt_dc320a3c-71b6-4055-bfd1-d9fa6f3f770f). The interface referenced from the <oa-itf-ref> production (its "referenced interface") MUST be an automation-compatible interface.

The **oleautomation** attribute MUST NOT be used on dispinterfaces.

#### Automation Members

Any [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) defines one or more automation members, either methods or properties. Methods and properties are specified differently, depending on whether they are defined in a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5), in a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9), or in a regular interface.

##### Interfaces Automation Members

An interface defined as "dual", or an interface referenced from the <oa-itf-ref> production, defines the automation members by extending the <op\_declarator> production from [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) with a new set of attributes:

1. operation-attribute = rpcidl-defined /
2. kw-id LWSP "(" LWSP integer-const-exp LWSP ")" /
3. kw-propget /
4. kw-propput /
5. kw-propputref /
6. kw-vararg /
7. kw-defaultcollelem /
8. kw-nonbrowsable /
9. kw-replaceable /
10. kw-restricted /
11. kw-uidefault /
12. kw-hidden /
13. oa-bindable-attr /
14. readonly-attr /
15. help-attr /
16. custom-attr

If the **id** attribute exists, it MUST represent the [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) that any client MUST pass in as the dispIdMember argument in calls to IDispatch::Invoke (see section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)) to execute the automation method identified by this value.

An operation that specifies any of the **propget**, **propput**, or **propputref** attributes MUST be a property accessor method. Two operations with the same property name MUST have the same DISPID; one of them MUST have the **propget** attribute; the other MUST have either the **propput** or **propputref** attribute.

If the attributes **propget**, **propput**, or **propputref** exist for a method, a client MUST set the values DISPATCH\_PROPERTYGET, DISPATCH\_PROPERTYPUT, or DISPATCH\_PROPERTYPUTREF, respectively, into the wFlags argument in calls to IDispatch::Invoke (see section 3.1.4.4) to disambiguate between the put and get semantics. The disambiguation is needed because the DISPID identifies only the property to be accessed, not the operation to be executed.

**kw-vararg**: Specifies that the final parameter of the method MUST be a SAFEARRAY containing VARIANTs or a pointer to a SAFEARRAY containing VARIANTs. This parameter MUST NOT be used on an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) property or on a property accessor method. For further specifications on handling "vararg" arguments, see section [3.1.4.4.3](#Section_be6e35f6932741649bdeffcd0fa0e07d).

**kw-defaultcollelem**: Specifies that a property is available for compiler-specific optimizations[<40>](#Appendix_A_40" \o "Product behavior note 40). The **[defaultcollelem]** attribute refers to the property as a whole, and MUST be applied to both get and set accessor methods, if they exist. A type SHOULD NOT contain more than one property with this attribute[<41>](#Appendix_A_41" \o "Product behavior note 41).

**kw-nonbrowsable**: Specifies that a property is not always safe to evaluate. The attribute MUST be specified only for property accessor methods or properties. Type browser clients SHOULD display the name of the property to users, but MUST NOT call the [**property's**](#gt_f930baab-25f1-4142-bced-5effc9f62d45) get accessor in order to display its contents.

**kw-replaceable**: This attribute SHOULD NOT be used[<42>](#Appendix_A_42" \o "Product behavior note 42).

**kw-restricted**: Specifies that the element is not intended to be used under all conditions, as specified in section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e).

**kw-uidefault**: Specifies that the element is intended to be used to represent its containing type to users. Type browser clients MUST display an element with this attribute whenever only one member of a type can be displayed. A type MUST NOT contain more than one element with this attribute.

##### Bindable Properties

Bindable attributes are applied only to property accessor methods and properties.

1. oa-bindable-attr = kw-bindable /
2. kw-immediatebind /
3. kw-defaultbind /
4. kw-displaybind /
5. kw-requestedit

**kw-bindable**: Specifies that the property can act as a bindable server (see section [2.2.49.1.3](#Section_2a89365308e7415a922628f8767c16ba)). The bindable attribute refers to the property as a whole, and is applied to both get and set accessor methods, if they exist.

If the server is capable of distinguishing between temporary and permanent states when the value of a property is changed (see the following entry for the immediatebind attribute), the server MUST call clients that are bound to the property, each time the value of the property is permanently changed. If the server is incapable of distinguishing between temporary and permanent states, the server MUST call clients that are bound to the property each time the value of the property is changed.

**kw-immediatebind**: Specifies that, if the server is capable of distinguishing between intermediate and final states when the value of a property is changed,[<43>](#Appendix_A_43" \o "Product behavior note 43) the server MUST call clients that are bound to the property each time the value of the property is temporarily or permanently changed. Properties that have the immediatebind attribute MUST also have the bindable attribute.

**kw-defaultbind**: Specifies that the property acts as the default bindable server for clients that bind to objects rather than to properties. Properties with the defaultbind attribute MUST also have the bindable attribute.

**kw-displaybind**: Specifies that Type browser clients MUST indicate to users that the property is bindable. Properties with the displaybind attribute MUST also have the bindable attribute.

**kw-requestedit**: Specifies that the server calls clients that are bound to the property before the property value is changed, in order to determine whether the change can be allowed. The server must not change the value of the property if any client specifies that the property cannot be changed. Properties with the requestedit attribute MUST also have the bindable attribute.

##### Dispinterfaces Automation Members

When defining a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5), the automation members can also be declared by using a split properties/methods syntax:

1. oa-odl-body =
2. kw-properties LWSP ":" LWSP \*oa-odl-prop LWSP
3. kw-methods LWSP ":" LWSP \*oa-odl-method
4. oa-odl-prop =
5. \*( operation-attributes ) LWSP oa-type-spec LWSP
6. Identifier LWSP ";"
7. oa-odl-method = op-declarator
8. readonly-attr = kw-readonly

The **readonly** attribute, if present, MUST be specified only in the context of an oa-odl-prop production. If this attribute is specified for a property, the client MUST NOT set either DISPATCH\_PROPERTYPUT or DISPATCH\_PROPERTYPUTREF into the wFlags argument in calls to IDispatch::Invoke (see section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)).

The automation properties MUST specify automation-compatible types. The automation methods MUST have return types that are automation-compatible, as well as parameters that MUST be automation-compatible.

#### Automation Parameters

The OLE Automation Protocol provides a number of attributes, specified in this section, that extend the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) syntax specified by the <param-attribute> production from [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) chapter 4. When these attributes are used, they specify how the client MUST pack the arguments when calling [IDispatch::Invoke (section 3.1.4.4)](#Section_5c2a199760d7496d8d9aed940bbb82eb).

The automation extensions to IDL expand the definition for the <param\_attribute> production in [C706]:

1. param-attribute = rpcidl-defined /
2. kw-defaultvalue LWSP "(" LWSP integer-const-exp LWSP ")" /
3. kw-optional /
4. kw-lcid /
5. kw-retval/
6. custom-attr

The **defaultvalue** attribute is valid only if the parameter is a scalar type, an enum, or a [BSTR (section 2.2.23)](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1). The expression specified with **defaultvalue** MUST be a constant, or an expression resolving to a constant, that can be represented by a VARIANT. The **defaultvalue** attribute MUST NOT be used on a parameter of a method declared with the **vararg** attribute.

The **optional** attribute is valid only if the parameter is of type VARIANT or VARIANT\*. The **optional** attribute MUST NOT be used on a parameter of a method declared with the **vararg** attribute. For information on handling **optional** arguments, see section [3.1.4.4.3](#Section_be6e35f6932741649bdeffcd0fa0e07d).

When applied to a parameter, the **lcid** attribute lets you pass a locale identifier to a function. A function MUST have at most one [lcid] parameter, which MUST be [in] only, and MUST have a type of LONG.

The **retval** attribute designates the parameter that receives the return value of an interface member that describes a method or get property. The attribute MUST appear on the last parameter of a method that has the **propget** attribute. The parameter MUST have the **[out]** attribute, and MUST be a pointer type.

The following parameter ordering (from left to right) MUST be respected when defining an automation-compatible method.

1. Required parameters (parameters that do not have the **defaultvalue** or **optional** attributes)
2. Parameters with the **defaultvalue** attribute, if any
3. Parameters with the **optional** attribute, if any
4. *lcid* parameter, if present
5. *retval* parameter, if present

#### AIDL Interfaces and ODL Dispinterfaces

An [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) is a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) whose properties and methods are specified using the two keywords: "properties" and "methods", as specified in [2.2.49.5.3](#Section_e23aaa6d3ad44886b6520203a1a50c58). All other [**automation interfaces**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) (that is, dual interfaces and reference dispinterfaces) are [**AIDL interfaces**](#gt_13017420-f6a8-4150-9549-b0d754dbb128).

This section explains how the methods defined in an AIDL interface map to properties and methods in an equivalent conceptual ODL dispinterface.

The conceptual signature of an ODL operation (a method call, or setting or retrieving a property) determines how an [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) MUST call [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) (see section 3.1.4.4): how it MUST pack the arguments in *pDispParams* and *rgVarRef*, as well as how *pVarResult* and *pExcepInfo* MUST be filled in on return (see section [3.1.4.4.2](#Section_9cf379f7fb3141fe9f9cc9a0136616e0)):

* The conceptual ODL arguments MUST be packed in *pDispParams* and *rgVarRef*.
* The conceptual return value MUST be returned in *pVarResult*.
* The actual HRESULT of an AIDL method MUST be set in pExcepInfo->scode.

##### Property Equivalence

Any [**AIDL interface**](#gt_13017420-f6a8-4150-9549-b0d754dbb128) method that has the "propget", "propput", or "propputref" method MUST map to a conceptual ODL property. If only a "propget" method exists for the property, it MUST map to a "readonly" conceptual ODL property.

* For an example illustrating this, see section [4.1](#Section_0fd123156d004d8883a40093d78156a4).
* AIDL supports the definition of indexed properties, which MUST NOT be expressed as ODL properties.

##### Method Equivalence

Any [**AIDL**](#gt_13017420-f6a8-4150-9549-b0d754dbb128) method that does not have any parameters with the **lcid** or **retval** attributes MUST map to an ODL method with the same argument types, and a void return type.

An AIDL method with a parameter with the **lcid** attribute MUST map to an ODL method that is equivalent to the AIDL method with the *lcid* parameter removed.

An AIDL method that has a parameter with the **retval** attribute MUST map to an ODL method whose return type is the type of the *retval* parameter, with one level of indirection removed. The argument types of the ODL method MUST correspond to the arguments in the AIDL method, with the *retval* parameter and any *lcid* parameter removed.

For an example illustrating this, see section [4.2](#Section_7df4503ea14b44ddb81908a381609b9d).

#### Coclass Specifications

The automation [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) extensions allow a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) to expose a rich set of attributes and behaviors, as specified below.

2. oa-coclass = "[" LWSP oa-coclass-attrs LWSP "]" LWSP
3. kw-coclass LWSP Identifier
4. LWSP "{" LWSP oa-coclass-body LWSP "}"
5. oa-coclass-attrs = oa-coclass-attr \*( LWSP ","LWSP oa-coclass-attr)
6. oa-coclass-attr = uuid-attr /
7. help-attr /
8. version-attr /
9. custom-attr /
10. kw-aggregatable /
11. kw-appobject /
12. kw-control /
13. kw-hidden /
14. kw-licensed /
15. kw-noncreatable /
16. kw-predeclid
17. oa-coclass-body = \*( oa-coclass-itf-decl )
18. oa-coclass-itf-decl = [ "[" LWSP oa-cid-attrs LWSP "]" LWSP ]
19. ( kw-interface / kw-dispinterface ) LWSP Identifier
20. LWSP ";"
21. oa-cid-attrs = oa-cid-attr \*( LWSP "," LWSP oa-cid-attr )
22. oa-cid-attr = kw-source /
23. kw-default /
24. kw-defaultvtable /
25. kw-restricted

A coclass statement MUST specify a [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) using the uuid-attr production. The other attributes in the oa-coclass-attrs production are optional.

**kw-aggregatable**: Specifies that the COM server is aggregatable (see section [2.2.49.1.1](#Section_491b9a5d4b3548a5b7ca5d6a8f3456aa)).

**kw-appobject**: Specifies that the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) is an "appobject coclass" and that its members MUST be included in the binding context of the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) (see section [3.5.4.1.1](#Section_cc88e6254b3a4cf9896626bdc913f62d)). It also specifies that ITypeInfo::CreateInstance MUST NOT create more than one instance of the coclass, and MUST return a reference to an existing instance of the coclass if one has already been created.

**kw-control**: Specifies that the COM server it describes is a visual control and is not intended to be used in a nonvisual environment. When applied to an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a), it specifies that all COM servers in the scope are visual controls. A Type browser client MAY limit the visibility of elements with this attribute[<44>](#Appendix_A_44" \o "Product behavior note 44).

**kw-hidden**: Specifies that the coclass element is not intended to be displayed to users. Type browser clients SHOULD NOT expose the functionality of elements with this attribute[<45>](#Appendix_A_45" \o "Product behavior note 45).

**kw-licensed**: Specifies that the COM server is licensed, and can be instantiated only by using a license-aware [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) creation method[<46>](#Appendix_A_46" \o "Product behavior note 46). Calls to any object creation method requesting an object of this type MUST fail if the creation method cannot validate licensing requirements at run time.

**kw-noncreatable**: Specifies that the coclass can be instantiated only by using a custom object creation method. Calls to any generic object creation method, including ITypeInfo::CreateInstance (as specified in section [3.7.4.11](#Section_b50cde468de74809ac5e884e3500b93c)), that request an object of this type MUST fail[<47>](#Appendix_A_47" \o "Product behavior note 47).

**kw-predeclid**: Specifies that ITypeInfo::CreateInstance MUST NOT create more than one instance of the coclass, and MUST return a reference to an existing instance of the coclass if one has already been created.

**kw-default**: Specifies that the interface or [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) it decorates is a default interface, and can be used in a context where it is not specified by name. Elements with the default attribute MUST NOT also have the [restricted] attribute.

An interface with the [default] attribute and without the [source] attribute is a "default nonsource interface" and defines the binding context of its coclass. Clients, such as macro languages, that can refer to only one interface per coclass MUST use this interface. A coclass MUST NOT contain more than one default nonsource interface.

An interface with both the [default] and [source] attributes behaves as specified by the [source] and [defaultvtable] sections that follow.

**kw-source**: Specifies that the interface is a [**source interface**](#gt_1036b31f-5af0-4298-a0e2-e3bf9933cd77) for the COM server described by the oa-coclass production. Such a COM server is a [**connectable server**](#gt_106828f1-1bf9-4cdb-9d12-d48b52495114) (as specified in section [2.2.49.1.2](#Section_4b96268c74984be6b20ed9220eb0057e)). The interface MUST be implemented by a client in order to receive events raised by the connectable server.

When combined with the [default] attribute, this attribute specifies that the interface is the default source interface for clients that can refer to only one source interface per coclass. A coclass MUST NOT contain more than one interface with both the [source] and [default] attributes.

**kw-defaultvtable**: Specifies that the interface is the default source [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408) for clients that can refer to only one source DCOM interface per coclass. Interfaces with this attribute MUST also have the [source] attribute. A coclass MUST NOT contain more than one interface with the [defaultvtable] attribute.

An interface that is declared with the [dual], [source], [default], and [defaultvtable] attributes MUST be both the default dispinterface and the default DCOM interface of its coclass.

**kw-restricted**: Specifies that the element is not intended to be used under all conditions, as specified in section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e).

#### Module Specifications

The module statement is provided as a means to define non-enum symbolic constants.

1. oa-module = [ "[" LWSP oa-module-attrs LWSP "]" LWSP ]
2. kw-module LWSP Identifier LWSP
3. "{" oa-module-body "}"
4. oa-module-attrs = oa-module-attr \*( LWSP "," LWSP oa-module-attr )
5. oa-module-attr = uuid-attr /
6. version-attr /
7. help-attr /
8. kw-dllname LWSP "(" LWSP string LWSP ")" /
9. kw-hidden
10. oa-module-body = \*( oa-const-stmt / oa-mmethod-stmt )
11. oa-const-stmt = [ "[" LWSP \*(help-attr) LWSP "]" LWSP ]
12. ( kw-const / kw-static ) LWSP
13. oa-base-type-spec LWSP Identifier LWSP
14. "=" LWSP const-exp LWSP ";"
15. oa-mmethod-stmt = [ "[" LWSP oa-mmethod-attrs LWSP "]" LWSP ]
16. oa-type-spec LWSP [ oa-mmethod-cc LWSP ]
17. Identifier LWSP param-declarators LWSP ";"
18. oa-mmethod-attrs = oa-mmethod-attr \*( LWSP "," LWSP oa-mmethod-attr )
19. oa-mmethod-attr =
20. kw-entry LWSP "(" LWSP oa-entry-id LWSP ")" /
21. kw-propget /
22. kw-propput /
23. kw-propputref /
24. kw-usesgetlasterror /
25. kw-vararg /
26. help-attr
27. oa-entry-id = string / integer-const-exp
28. oa-mmethod-cc = kw-cdecl /
29. kw-stdcall /
30. kw-pascal

**kw-usesgetlasterror:** Specifies that the module method supports an implementation-specific local error handling method[<48>](#Appendix_A_48" \o "Product behavior note 48).

**kw-vararg:** Specifies that the final parameter of the method MUST be of type SAFEARRAY(VARIANT) or SAFEARRAY(VARIANT)\*. This attribute MUST NOT be used on an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) property or on a property accessor method. For information on handling **vararg** arguments, see section [3.1.4.4.3](#Section_be6e35f6932741649bdeffcd0fa0e07d).

**oa-mmethod-cc:**  Specifies an implementation-specific local calling convention for the method[<49>](#Appendix_A_49" \o "Product behavior note 49).

The statements within the scope of the module statement define constant symbols with the specified type and values, and static entry points in the module are specified by the string argument of the **dllname** attribute.

All oa-mmethod-stmt productions MUST have an entry attribute specified. If oa-entry-id is a string, this is a named entry point. If oa-entry-id is an integer, the entry point is defined by an ordinal.

#### Referencing External Types

The automation [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) extensions allow elements defined inside an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) to reference types defined in an external [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e). This is done through the use of the importlib statement:

1. oa-importlib = kw-importlib LWSP "(" LWSP string LWSP ")" LWSP ";"

The string specified in the importlib statement is an implementation-specific string[<50>](#Appendix_A_50" \o "Product behavior note 50) that MUST allow an automation type library to locate the definitions of referenced types that are defined in another automation type library.

### String Handling

The following sections specify the ways in which strings are compared and stored in the [ITypeComp::Bind](#Section_476f00da080640d9bbf36059154abbb7), [ITypeComp::BindType](#Section_cf61a786b1814267bb6a0987eeb17b38), [ITypeLib::IsName](#Section_70ea09581a204d04b3d3ab4d12446c08), and [ITypeLib::FindName](#Section_8d41f5777cba48f294a4141372f59a0e) methods.

#### String Equivalence

The OLE Automation Protocol interfaces MUST treat two strings as equivalent when a comparison of their values with the CompareString method (see [[MS-UCODEREF]](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb) section 1) returns CSTR\_EQUAL.

All automation string comparisons MUST conform to the behavior of the CompareString method with the following bit flags set in its *dwCmpFlags* parameter:

* NORM\_IGNORECASE
* NORM\_IGNOREWIDTH
* NORM\_IGNOREKANATYPE

#### Globalization

The OLE Automation Protocol interfaces MUST use the WideCharToMultiByte method (as specified in [[MS-UCODEREF]](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb) section 2) to preprocess strings before they are compared or stored.

Automation string preprocessing MUST conform to the behavior of the WideCharToMultiByte method with the following parameter values:

* CodePage: CP\_ACP (zero – the current system Windows ANSI code page)
* dwFlags: zero or NULL
* lpDefaultChar: zero or NULL
* lpUsedDefaultChar: zero or NULL

### Automation Hash Values

An automation hash value is a numeric representation of a string that is used by an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) to perform implementation-specific optimizations[<51>](#Appendix_A_51" \o "Product behavior note 51) in methods that reference types and type members by name.

Implementation of the hash function is optional. An automation hash value of zero MAY be used to represent any string[<52>](#Appendix_A_52" \o "Product behavior note 52). If an automation hash value is nonzero, it MUST be computed from the specified string after it is preprocessed, as specified in section [2.2.50.2](#Section_8ba9784a29b8471aa669e493e5f28326), using a hash algorithm that MUST be consistent with the pseudocode specified in this section.

The values of [out] parameters returned by automation server methods MUST be the same for both zero and matching nonzero automation hash values.

#### ComputeHash Method

The ComputeHash algorithm requires two externally specified values:

1. A string of no more than 255 characters, converted to the ANSI code page as specified in [2.2.50.2](#Section_8ba9784a29b8471aa669e493e5f28326).
2. A [**language code identifier (LCID)**](#gt_c7f99c66-592f-4053-b62a-878c189653b6) that specifies the locale ID associated with the string.

The ComputeHash method uses the named constants defined in the [LocaleNames (section 2.2.51.4)](#Section_382297c6fcb14711924979d31e161545) and [PrimaryLookupTables (section 2.2.51.5)](#Section_0dd943a0c4c04140abb9e6e6e45f307b) sections.

Use alternative hash function if the locale ID uses DBCS-encoded strings.

1. SET PrimaryLocale to the bitwise AND of LocaleID and 0x000003FF
2. IF PrimaryLocale is LocaleChinese
3. OR PrimaryLocale is LocaleJapanese
4. OR PrimaryLocale is LocaleKorean THEN
5. CALL ComputeHashDBCS
6. WITH LocaleID, Name
7. RETURNING HashValue
8. RETURN HashValue as a 32-bit unsigned integer
9. ENDIF
10. COMMENT Set LookupTable and Encoding Mask based on Locale ID.
11. CASE LocaleID OF
12. LocaleRussian: SET LookupTable to Eur\_English\_1251
13. SET EncodingMask to 0x00300000
14. LocaleGreek: SET LookupTable to WGreek
15. SET EncodingMask to 0x00800000
16. LocaleIceland: SET LookupTable to WIceland
17. SET EncodingMask to 0x00900000
18. LocaleTurkish: SET LookupTable to WTurkish
19. SET EncodingMask to 0x00A00000
20. LocaleNorway: SET LookupTable to WNorwegian
21. SET EncodingMask to 0x00B00000
22. LocaleIreland: SET LookupTable to WEngIreland
23. SET EncodingMask to 0x00C00000
24. LocaleHebrew: SET LookupTable to WHebrew
25. SET EncodingMask to 0x00E00000
26. LocaleCzech: SET LookupTable to Eur\_English\_1250
27. SET EncodingMask to 0x00200000
28. LocaleHungary: SET LookupTable to Eur\_English\_1250
29. SET EncodingMask to 0x00200000
30. LocalePoland: SET LookupTable to Eur\_English\_1250
31. SET EncodingMask to 0x00200000
32. LocaleSlovak: SET LookupTable to Eur\_English\_1250
33. SET EncodingMask to 0x00200000
34. OTHERS: IF LocaleID is LocaleFarsi
35. OR the lower byte of LocaleID is
36. SecondaryLocaleArabic THEN
37. SET LookupTable to WArabic
38. SET EncodingMask to 0x00D00000
39. ELSE
40. SET LookupTable to US\_English\_1252
41. SET EncodingMask to 0x00100000
42. ENDIF
43. ENDCASE
44. SET HashAccumulator to 0x0DEADBEE
45. COMMENT Step through the characters in the string,
46. multiplying the accumulator by 37 at each step
47. and adding a value specified by the value of the
48. character.
49. FOR each Character in Name
50. COMPUTE HashAccumulator as HashAccumulator multiplied by 37,
51. allowing unsigned 32 bit overflows
52. COMPUTE HashAcculumator as LookupTable (Character) added to
53. HashAccumulator, allowing unsigned 32 bit overflows
54. END FOR
55. COMPUTE HashAccumulator as the remainder when HashAccumulator is
56. divided by 0x0001003F
57. COMPUTE HashAccumulator as the bitwise AND of HashAccumulator and
58. 0x0000FFFF
59. COMPUTE HashAccumulator as the bitwise OR of HashAccumulator and
60. EncodingMask
61. RETURN HashAccumulator as a 32-bit unsigned integer

#### ComputeHashDBCS Method

The DBCS implementation of the hash algorithm uses the default lookup table for all locales.

The ComputeHashDBCS method uses the named constants defined in the [LocaleNames (section 2.2.51.4)](#Section_382297c6fcb14711924979d31e161545), [PrimaryLookupTables (section 2.2.51.5)](#Section_0dd943a0c4c04140abb9e6e6e45f307b), and [DBCS Substitution Tables (section 2.2.51.6)](#Section_273681debad04d3d9f441da41d0e4f70) sections.

Set the substitution table and the character range boundaries based on the locale ID and define the character ranges that contain two-byte characters. The Korean and Chinese character tables have two-byte characters only within a single range of character values, so the second range is empty.

1. SET LookupTable to US\_English\_1252
2. SET PrimaryLocale to the Logical AND of LocaleID and 0x000003FF
3. COMMENT This method MUST only be called when PrimaryLocale
4. is LocaleJapanese, LocaleKorean, or LocaleChinese.
5. CASE PrimaryLocale OF
6. LocaleJapanese: SET SubstitutionTable to WJapanese
7. SET EncodingMask to 0x00400000
8. SET LowerRangeStart to 0x80
9. SET LowerRangeEnd to 0xA0
10. SET UpperRangeStart to 0xE0
11. SET UpperRangeEnd to 0xFF
12. LocaleKorean: SET SubstitutionTable to WKorean
13. SET EncodingMask to 0x00500000
14. SET LowerRangeStart to 0x81
15. SET LowerRangeEnd to 0xFE
16. SET UpperRangeStart to 0xFE
17. SET UpperRangeEnd to 0xFE
18. LocaleChinese: SET UpperRangeStart to 0xFE
19. SET UpperRangeEnd to 0xFE
20. SET SecondaryLocale to the lower 16 bits of LocaleID
21. COMPUTE SecondaryLocale AS SecondaryLocale divided by 1024
22. IF SecondaryLocale is SecondaryLocaleChineseTraditional
23. OR SecondaryLocale is SecondaryLocaleChineseHongKong THEN
24. SET SubstitutionTable to WChineseTraditional
25. SET EncodingMask to 0x00700000
26. SET LowerRangeStart to 0x81
27. SET LowerRangeEnd to 0xFE
28. ELSE
29. SET SubstitutionTable to WChineseSimplified
30. SET EncodingMask to 0x00600000
31. SET LowerRangeStart to 0xA1
32. SET LowerRangeEnd to 0xFE
33. ENDIF
35. ENDCASE
36. COMMENT Initialize hash accumulator with a predefined value.
37. COMMENT Initialize byte index and loop flag.
38. SET HashAccumulator to 0x0DEADBEE
39. SET ByteIndex to refer to the first byte in Name
40. SET ByteInName to the value of the byte indexed by ByteIndex
41. SET Break to False
42. COMMENT Step through the characters in the string, multiplying
43. the accumulator by 37 at each step and adding a value
44. specified by the value of the character.
45. REPEAT
47. COMMENT Store the current byte or exit the loop.
48. IF the value of ByteInName is zero THEN
49. SET Break to True
50. ELSE
51. SET TempChar to the value of ByteInName
52. ENDIF
53. COMMENT Increment the byte index. If the previous byte
54. was the first byte of a DBCS two-byte character,
55. compute the DBCS character value and increment
56. the byte index again.
57. IF Break is False THEN
58. INCREMENT ByteIndex to refer to the next byte in Name
59. SET ByteInName to
60. the value of the byte indexed by ByteIndex
61. IF (TempChar >= LowerRangeStart AND
62. TempChar <= LowerRangeEnd)
63. OR (TempChar >= UpperRangeStart AND
64. TempChar <= UpperRangeEnd) THEN
65. COMMENT If the second byte of the DBCS character
66. is zero, ignore the character and
67. exit the loop.
68. IF the value of ByteInName is zero THEN
69. SET Break to True
70. ELSE
71. COMPUTE TempChar as TempChar
72. multiplied by 256
73. COMPUTE TempChar as the value of ByteInName
74. added to TempChar
75. INCREMENT ByteIndex to refer
76. to the next byte in Name
77. SET ByteInName to the value of the byte
78. indexed by ByteIndex
79. ENDIF
80. ENDIF
81. ENDIF
82. IF Break is False THEN
83. COMMENT If the character has an upper byte, replace
84. its value with the appropriate value from a
85. Locale-specified substitution table.
86. COMMENT If the upper byte is nonzero after substitution,
87. update the hash accumulator using its value.
88. IF TempChar > 255
89. CALL MapDBChar
90. WITH TempChar, SubstitutionTable
91. RETURNING TempChar
92. SET HighByte to the upper byte of TempChar
93. IF HighByte is not 0
94. COMPUTE HashAccumulator as HashAccumulator
95. multiplied by 37, allowing unsigned 32 bit
96. overflows
97. COMPUTE HashAcculumator as LookupTable
98. (HighByte) added to HashAccumulator,
99. allowing unsigned 32 bit overflows
100. ENDIF
101. ENDIF
102. COMMENT Update the hash accumulator using the value of
103. a one-byte character or the lower byte of a
104. two-byte character.
105. SET LowByte to the lower byte of TempChar
106. COMPUTE HashAccumulator as HashAccumulator
107. multiplied by 37, allowing unsigned
108. 32 bit overflows
109. COMPUTE HashAcculumator as LookupTable
110. (LowByte) added to HashAccumulator, allowing
111. unsigned 32 bit overflows
112. ENDIF
113. UNTIL Break is True
114. COMPUTE HashAccumulator as the remainder when HashAccumulator is
115. divided by 0x0001003F
116. COMPUTE HashAccumulator as the bitwise AND of HashAccumulator and
117. 0x0000FFFF
118. COMPUTE HashAccumulator as the bitwise OR of HashAccumulator and
119. EncodingMask
120. RETURN HashAccumulator as a 32-bit unsigned integer

#### MapDBChar Method

Loop through the table, returning a substitution character if a match is found. If no match is found, return the original character.

1. FOR each Row in SubstitutionTable
2. IF Row (0) is TempChar THEN
3. RETURN Row (1)
4. END IF
5. END FOR
6. RETURN TempChar

#### Locale Names

1. SET LocaleRussian to 0x0419
2. SET LocaleGreek to 0x0408
3. SET LocaleIceland to 0x040f
4. SET LocaleTurkish to 0x041f
5. SET LocaleNorway to 0x0814
6. SET LocaleIreland to 0x1809
7. SET LocaleHebrew to 0x040d
8. SET LocaleCzech to 0x0405
9. SET LocaleHungary to 0x040e
10. SET LocalePoland to 0x0415
11. SET LocaleSlovak to 0x041b
12. SET LocaleFarsi to 0x0429
13. SET LocaleChinese to 0x04
14. SET LocaleJapanese to 0x11
15. SET LocaleKorean to 0x12
16. SET SecondaryLocaleChineseTraditional to 0x01
17. SET SecondaryLocaleChineseHongKong to 0x03
18. SET SecondaryLocaleArabic to 0x01

#### Primary Lookup Tables

1. SET US\_English\_1252 (256) to
2. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
3. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
4. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 0,
5. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
6. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
7. 80, 81, 82, 83, 84, 85, 86, 86, 88, 85, 90, 91, 92, 93, 94, 95,
8. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
9. 80, 81, 82, 83, 84, 85, 86, 86, 88, 85, 90, 123, 124, 125, 126, 127,
10. 127, 127, 130, 70, 132, 133, 134, 135, 127, 137, 83, 139, 140, 127,
11. 127, 127,
12. 127, 145, 146, 147, 148, 149, 150, 150, 152, 153, 83, 155, 140, 127,
13. 127, 85,
14. 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 65, 171, 172, 150,
15. 174, 175,
16. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 79, 187, 188, 189, 190,
17. 191,
18. 65, 65, 65, 65, 65, 65, 65, 67, 69, 69, 69, 69, 73, 73, 73, 73,
19. 68, 78, 79, 79, 79, 79, 79, 215, 79, 85, 85, 85, 85, 85, 222, 223,
20. 65, 65, 65, 65, 65, 65, 65, 67, 69, 69, 69, 69, 73, 73, 73, 73,
21. 68, 78, 79, 79, 79, 79, 79, 247, 79, 85, 85, 85, 85, 85, 222, 85
22. SET Eur\_1250 (256) to
23. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
24. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
25. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 0,
26. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
27. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
28. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
29. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
30. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
31. 127,
32. 127, 127, 130, 127, 132, 133, 134, 135, 127, 137, 83, 139, 83, 84,
33. 90, 90,
34. 127, 145, 146, 147, 148, 149, 150, 150, 127, 153, 83, 155, 83, 84,
35. 90, 90,
36. 160, 127, 162, 76, 164, 65, 166, 167, 168, 169, 83, 171, 172, 150,
37. 174, 90,
38. 176, 177, 178, 76, 180, 181, 182, 183, 184, 65, 83, 187, 76, 189,
39. 76, 90,
40. 82, 65, 65, 65, 65, 76, 67, 67, 67, 69, 69, 69, 69, 73, 73, 68,
41. 208, 78, 78, 79, 79, 79, 79, 215, 82, 85, 85, 85, 85, 89, 84, 223,
42. 82, 65, 65, 65, 65, 76, 67, 67, 67, 69, 69, 69, 69, 73, 73, 68,
43. 208, 78, 78, 79, 79, 79, 79, 247, 82, 85, 85, 85, 85, 89, 84, 255
44. SET Eur\_English\_1251 (256) to
45. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
46. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
47. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 0,
48. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
49. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
50. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
51. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
52. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
53. 127,
54. 127, 127, 130, 70, 132, 133, 134, 135, 127, 137, 83, 139, 140,
55. 127, 127, 127,
56. 127, 145, 146, 147, 148, 149, 150, 150, 152, 153, 83, 155, 140,
57. 127, 127, 89,
58. 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 65, 171, 172,
59. 150, 174, 175,
60. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 79, 187, 188, 189,
61. 190, 191,
62. 65, 65, 65, 65, 65, 65, 198, 67, 69, 69, 69, 69, 73, 73, 73, 73,
63. 208, 78, 79, 79, 79, 79, 79, 215, 79, 85, 85, 85, 85, 89, 222,
64. 223,
65. 65, 65, 65, 65, 65, 65, 198, 67, 69, 69, 69, 69, 73, 73, 73, 73,
66. 208, 78, 79, 79, 79, 79, 79, 247, 79, 85, 85, 85, 85, 89, 222, 89
67. SET WGreek (256) to
68. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
69. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
70. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
71. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
72. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
73. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
74. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
75. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
76. 0,
77. 0, 0, 130, 70, 132, 133, 134, 135, 0, 137, 0, 139, 0, 0, 0, 0,
78. 0, 145, 146, 147, 148, 149, 45, 45, 0, 153, 0, 155, 0, 0, 0, 0,
79. 9, 161, 162, 163, 164, 165, 166, 167, 168, 169, 0, 171, 172, 45,
80. 174, 45,
81. 176, 177, 50, 51, 180, 181, 182, 183, 184, 185, 186, 187, 188,
82. 189, 190, 191,
83. 186, 162, 194, 195, 196, 184, 198, 185, 200, 186, 202, 203, 204,
84. 205, 206, 188,
85. 208, 209, 0, 211, 212, 190, 214, 215, 216, 191, 186, 190, 162,
86. 184, 185, 186,
87. 190, 162, 194, 195, 196, 184, 198, 185, 200, 186, 202, 203, 204,
88. 205, 206, 188,
89. 208, 209, 211, 211, 212, 190, 214, 215, 216, 191, 186, 190, 188,
90. 190, 191, 0
91. SET WIceland (256) to
92. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
93. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
94. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
95. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
96. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
97. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
98. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
99. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
100. 0,
101. 0, 0, 130, 70, 132, 133, 134, 135, 0, 137, 83, 139, 140, 0, 0,
102. 0,
103. 0, 145, 146, 147, 148, 149, 45, 45, 152, 153, 83, 155, 140, 0,
104. 0, 89,
105. 9, 161, 162, 163, 164, 165, 166, 167, 168, 169, 65, 171, 172, 45,
106. 174, 175,
107. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 79, 187, 188, 189,
108. 190, 191,
109. 65, 193, 65, 65, 65, 65, 198, 67, 69, 201, 69, 69, 73, 205, 73,
110. 73,
111. 208, 78, 79, 211, 79, 79, 214, 215, 214, 85, 218, 85, 85, 221,
112. 222, 223,
113. 65, 193, 65, 65, 65, 65, 198, 67, 69, 201, 69, 69, 73, 205, 73,
114. 73,
115. 208, 78, 79, 211, 79, 79, 214, 247, 214, 85, 218, 85, 85, 221,
116. 222, 89
117. SET WTurkish (256) to
118. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
119. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
120. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
121. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
122. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
123. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
124. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
125. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
126. 0,
127. 0, 0, 130, 70, 132, 133, 134, 135, 0, 137, 83, 139, 140, 0, 0,
128. 0,
129. 0, 145, 146, 147, 148, 149, 45, 45, 152, 153, 83, 155, 140, 0, 0,
130. 89,
131. 9, 161, 162, 163, 164, 165, 166, 167, 168, 169, 65, 171, 172, 45,
132. 174, 175,
133. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 79, 187, 188, 189,
134. 190, 191,
135. 65, 65, 65, 65, 65, 65, 198, 199, 69, 69, 69, 69, 73, 73, 73,
136. 73,
137. 208, 78, 79, 79, 79, 79, 214, 215, 79, 85, 85, 85, 220, 221, 222,
138. 223,
139. 65, 65, 65, 65, 65, 65, 198, 199, 69, 69, 69, 69, 73, 73, 73,
140. 73,
141. 208, 78, 79, 79, 79, 79, 214, 247, 79, 85, 85, 85, 220, 221, 222,
142. 89
143. SET WNorwegian (256) to
144. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
145. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
146. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
147. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
148. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
149. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
150. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
151. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
152. 0,
153. 0, 0, 130, 70, 132, 133, 134, 135, 0, 137, 83, 139, 140, 0, 0,
154. 0,
155. 0, 145, 146, 147, 148, 149, 45, 45, 152, 153, 83, 155, 140, 0, 0,
156. 89,
157. 9, 161, 162, 163, 164, 165, 166, 167, 168, 169, 65, 171, 172, 45,
158. 174, 175,
159. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 79, 187, 188, 189,
160. 190, 191,
161. 65, 65, 65, 65, 196, 197, 196, 67, 69, 69, 69, 69, 73, 73, 73, 73,
162. 68, 78, 79, 79, 79, 79, 214, 215, 214, 85, 85, 85, 89, 89, 222,
163. 223,
164. 65, 65, 65, 65, 196, 197, 196, 67, 69, 69, 69, 69, 73, 73, 73, 73,
165. 68, 78, 79, 79, 79, 79, 214, 247, 214, 85, 85, 85, 89, 89, 222, 89
166. SET WIreland (256) to
167. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
168. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
169. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
170. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
171. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
172. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
173. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
174. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
175. 0,
176. 0, 0, 130, 70, 132, 133, 134, 135, 0, 137, 83, 139, 140, 0, 0,
177. 0,
178. 0, 145, 146, 147, 148, 149, 45, 45, 152, 153, 83, 155, 140, 0,
179. 0, 89,
180. 9, 161, 162, 163, 164, 165, 166, 167, 168, 169, 65, 171, 172, 45,
181. 174, 175,
182. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 79, 187, 188, 189,
183. 190, 191,
184. 65, 65, 65, 65, 65, 65, 198, 67, 69, 69, 69, 69, 73, 73, 73, 73,
185. 208, 78, 79, 79, 79, 79, 79, 215, 79, 85, 85, 85, 85, 89, 222,
186. 223,
187. 65, 65, 65, 65, 65, 65, 198, 67, 69, 69, 69, 69, 73, 73, 73, 73,
188. 208, 78, 79, 79, 79, 79, 79, 247, 79, 85, 85, 85, 85, 89, 222, 89
189. SET WArabic (256) to
190. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
191. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
192. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
193. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
194. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
195. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
196. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
197. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
198. 127,
199. 128, 129, 130, 70, 132, 133, 134, 135, 94, 137, 138, 139, 140,
200. 141, 142, 143,
201. 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 140,
202. 157, 0, 159,
203. 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
204. 173, 174, 175,
205. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 186, 187, 188, 189,
206. 190, 191,
207. 192, 193, 194, 193, 193, 193, 193, 199, 200, 201, 201, 203, 204,
208. 205, 206, 207,
209. 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 0,
210. 221, 222, 223,
211. 65, 225, 65, 227, 228, 229, 230, 67, 69, 69, 69, 69, 236, 236,
212. 73, 73,
213. 240, 241, 242, 243, 79, 245, 246, 247, 248, 85, 250, 85, 85, 0,
214. 0, 255
215. SET WHebrew (256) to
216. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
217. 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
218. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
219. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
220. 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
221. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
222. 96, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
223. 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 123, 124, 125, 126,
224. 127,
225. 128, 129, 130, 70, 132, 133, 134, 135, 94, 137, 138, 139, 140,
226. 141, 142, 143,
227. 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
228. 157, 158, 159,
229. 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
230. 173, 174, 175,
231. 176, 177, 50, 51, 180, 181, 182, 183, 184, 49, 186, 187, 188, 189,
232. 190, 191,
233. 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204,
234. 205, 206, 207,
235. 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
236. 221, 222, 223,
237. 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 234, 236,
238. 237, 237, 239,
239. 239, 241, 242, 243, 243, 245, 245, 247, 248, 249, 250, 251, 252,
240. 0, 0, 255

#### DBCS Substitution Tables

1. SET WJapanese (235)(2) to
2. (0x815C, 0x815C), (0x815B, 0x815B), (0x829F, 0x00A7),
3. (0x82A0, 0x00B1),
4. (0x82A1, 0x00A8), (0x82A2, 0x00B2), (0x82A3, 0x00A9),
5. (0x82A4, 0x00B3),
6. (0x82A5, 0x00AA), (0x82A6, 0x00B4), (0x82A7, 0x00AB),
7. (0x82A8, 0x00B5),
8. (0x82A9, 0x00B6), (0x82AA, 0xB6DE), (0x82AB, 0x00B7),
9. (0x82AC, 0xB7DE),
10. (0x82AD, 0x00B8), (0x82AE, 0xB8DE), (0x82AF, 0x00B9),
11. (0x82B0, 0xB9DE),
12. (0x82B1, 0x00BA), (0x82B2, 0xBADE), (0x82B3, 0x00BB),
13. (0x82B4, 0xBBDE),
14. (0x82B5, 0x00BC), (0x82B6, 0xBCDE), (0x82B7, 0x00BD),
15. (0x82B8, 0xBDDE),
16. (0x82B9, 0x00BE), (0x82BA, 0xBEDE), (0x82BB, 0x00BF),
17. (0x82BC, 0xBFDE),
18. (0x82BD, 0x00C0), (0x82BE, 0xC0DE), (0x82BF, 0x00C1),
19. (0x82C0, 0xC1DE),
20. (0x82C1, 0x00AF), (0x82C2, 0x00C2), (0x82C3, 0xC2DE),
21. (0x82C4, 0x00C3),
22. (0x82C5, 0xC3DE), (0x82C6, 0x00C4), (0x82C7, 0xC4DE),
23. (0x82C8, 0x00C5),
24. (0x82C9, 0x00C6), (0x82CA, 0x00C7), (0x82CB, 0x00C8),
25. (0x82CC, 0x00C9),
26. (0x82CD, 0x00CA), (0x82CE, 0xCADE), (0x82CF, 0xCADF),
27. (0x82D0, 0x00CB),
28. (0x82D1, 0xCBDE), (0x82D2, 0xCBDF), (0x82D3, 0x00CC),
29. (0x82D4, 0xCCDE),
30. (0x82D5, 0xCCDF), (0x82D6, 0x00CD), (0x82D7, 0xCDDE),
31. (0x82D8, 0xCDDF),
32. (0x82D9, 0x00CE), (0x82DA, 0xCEDE), (0x82DB, 0xCEDF),
33. (0x82DC, 0x00CF),
34. (0x82DD, 0x00D0), (0x82DE, 0x00D1), (0x82DF, 0x00D2),
35. (0x82E0, 0x00D3),
36. (0x82E1, 0x00AC), (0x82E2, 0x00D4), (0x82E3, 0x00AD),
37. (0x82E4, 0x00D5),
38. (0x82E5, 0x00AE), (0x82E6, 0x00D6), (0x82E7, 0x00D7),
39. (0x82E8, 0x00D8),
40. (0x82E9, 0x00D9), (0x82EA, 0x00DA), (0x82EB, 0x00DB),
41. (0x82EC, 0x838E),
42. (0x82ED, 0x00DC), (0x82EE, 0x8390), (0x82EF, 0x8391),
43. (0x82F0, 0x00A6),
44. (0x82F1, 0x00DD), (0x8340, 0x00A7), (0x8341, 0x00B1),
45. (0x8342, 0x00A8),
46. (0x8343, 0x00B2), (0x8344, 0x00A9), (0x8345, 0x00B3),
47. (0x8346, 0x00AA),
48. (0x8347, 0x00B4), (0x8348, 0x00AB), (0x8349, 0x00B5),
49. (0x834A, 0x00B6),
50. (0x834B, 0xB6DE), (0x834C, 0x00B7), (0x834D, 0xB7DE),
51. (0x834E, 0x00B8),
52. (0x834F, 0xB8DE), (0x8350, 0x00B9), (0x8351, 0xB9DE),
53. (0x8352, 0x00BA),
54. (0x8353, 0xBADE), (0x8354, 0x00BB), (0x8355, 0xBBDE),
55. (0x8356, 0x00BC),
56. (0x8357, 0xBCDE), (0x8358, 0x00BD), (0x8359, 0xBDDE),
57. (0x835A, 0x00BE),
58. (0x835B, 0xBEDE), (0x835C, 0x00BF), (0x835D, 0xBFDE),
59. (0x835E, 0x00C0),
60. (0x835F, 0xC0DE), (0x8360, 0x00C1), (0x8361, 0xC1DE),
61. (0x8362, 0x00AF),
62. (0x8363, 0x00C2), (0x8364, 0xC2DE), (0x8365, 0x00C3),
63. (0x8366, 0xC3DE),
64. (0x8367, 0x00C4), (0x8368, 0xC4DE), (0x8369, 0x00C5),
65. (0x836A, 0x00C6),
66. (0x836B, 0x00C7), (0x836C, 0x00C8), (0x836D, 0x00C9),
67. (0x836E, 0x00CA),
68. (0x836F, 0xCADE), (0x8370, 0xCADF), (0x8371, 0x00CB),
69. (0x8372, 0xCBDE),
70. (0x8373, 0xCBDF), (0x8374, 0x00CC), (0x8375, 0xCCDE),
71. (0x8376, 0xCCDF),
72. (0x8377, 0x00CD), (0x8378, 0xCDDE), (0x8379, 0xCDDF),
73. (0x837A, 0x00CE),
74. (0x837B, 0xCEDE), (0x837C, 0xCEDF), (0x837D, 0x00CF),
75. (0x837E, 0x00D0),
76. (0x8380, 0x00D1), (0x8381, 0x00D2), (0x8382, 0x00D3),
77. (0x8383, 0x00AC),
78. (0x8384, 0x00D4), (0x8385, 0x00AD), (0x8386, 0x00D5),
79. (0x8387, 0x00AE),
80. (0x8388, 0x00D6), (0x8389, 0x00D7), (0x838A, 0x00D8),
81. (0x838B, 0x00D9),
82. (0x838C, 0x00DA), (0x838D, 0x00DB), (0x838E, 0x838E),
83. (0x838F, 0x00DC),
84. (0x8390, 0x8390), (0x8391, 0x8391), (0x8392, 0x00A6),
85. (0x8393, 0x00DD),
86. (0x8394, 0xB3DE), (0x8395, 0x8395), (0x8396, 0x8396),
87. (0x824F, 0x30),
88. (0x8250, 0x31), (0x8251, 0x32), (0x8252, 0x33), (0x8253, 0x34),
89. (0x8254, 0x35), (0x8255, 0x36), (0x8256, 0x37), (0x8257, 0x38),
90. (0x8258, 0x39), (0x8260, 0x41), (0x8261, 0x42), (0x8262, 0x43),
91. (0x8263, 0x44), (0x8264, 0x45), (0x8265, 0x46), (0x8266, 0x47),
92. (0x8267, 0x48), (0x8268, 0x49), (0x8269, 0x4A), (0x826A, 0x4B),
93. (0x826B, 0x4C), (0x826C, 0x4D), (0x826D, 0x4E), (0x826E, 0x4F),
94. (0x826F, 0x50), (0x8270, 0x51), (0x8271, 0x52), (0x8272, 0x53),
95. (0x8273, 0x54), (0x8274, 0x55), (0x8275, 0x56), (0x8276, 0x57),
96. (0x8277, 0x58), (0x8278, 0x59), (0x8279, 0x5A), (0x8151, 0x5F),
97. (0x8281, 0x61), (0x8282, 0x62), (0x8283, 0x63), (0x8284, 0x64),
98. (0x8285, 0x65), (0x8286, 0x66), (0x8287, 0x67), (0x8288, 0x68),
99. (0x8289, 0x69), (0x828A, 0x6A), (0x828B, 0x6B), (0x828C, 0x6C),
100. (0x828D, 0x6D), (0x828E, 0x6E), (0x828F, 0x6F), (0x8290, 0x70),
101. (0x8291, 0x71), (0x8292, 0x72), (0x8293, 0x73), (0x8294, 0x74),
102. (0x8295, 0x75), (0x8296, 0x76), (0x8297, 0x77), (0x8298, 0x78),
103. (0x8299, 0x79), (0x829A, 0x7A)
104. SET WKorean (64)(2) to
105. (0xA3B0, 0x30), (0xA3B1, 0x31), (0xA3B2, 0x32), (0xA3B3, 0x33),
106. (0xA3B4, 0x34), (0xA3B5, 0x35), (0xA3B6, 0x36), (0xA3B7, 0x37),
107. (0xA3B8, 0x38), (0xA3B9, 0x39), (0xA3C1, 0x41), (0xA3C2, 0x42),
108. (0xA3C3, 0x43), (0xA3C4, 0x44), (0xA3C5, 0x45), (0xA3C6, 0x46),
109. (0xA3C7, 0x47), (0xA3C8, 0x48), (0xA3C9, 0x49), (0xA3CA, 0x4A),
110. (0xA3CB, 0x4B), (0xA3CC, 0x4C), (0xA3CD, 0x4D), (0xA3CE, 0x4E),
111. (0xA3CF, 0x4F), (0xA3D0, 0x50), (0xA3D1, 0x51), (0xA3D2, 0x52),
112. (0xA3D3, 0x53), (0xA3D4, 0x54), (0xA3D5, 0x55), (0xA3D6, 0x56),
113. (0xA3D7, 0x57), (0xA3D8, 0x58), (0xA3D9, 0x59), (0xA3DA, 0x5A),
114. (0xA3DF, 0x5F), (0xA3E1, 0x61), (0xA3E2, 0x62), (0xA3E3, 0x63),
115. (0xA3E4, 0x64), (0xA3E5, 0x65), (0xA3E6, 0x66), (0xA3E7, 0x67),
116. (0xA3E8, 0x68), (0xA3E9, 0x69), (0xA3EA, 0x6A), (0xA3EB, 0x6B),
117. (0xA3EC, 0x6C), (0xA3ED, 0x6D), (0xA3EE, 0x6E), (0xA3EF, 0x6F),
118. (0xA3F0, 0x70), (0xA3F1, 0x71), (0xA3F2, 0x72), (0xA3F3, 0x73),
119. (0xA3F4, 0x74), (0xA3F5, 0x75), (0xA3F6, 0x76), (0xA3F7, 0x77),
120. (0xA3F8, 0x78), (0xA3F9, 0x79), (0xA3FA, 0x7A)
122. SET WChineseTraditional (64)(2) to
123. (0xA2AF, 0x30), (0xA2B0, 0x31), (0xA2B1, 0x32), (0xA2B2, 0x33),
124. (0xA2B3, 0x34), (0xA2B4, 0x35), (0xA2B5, 0x36), (0xA2B6, 0x37),
125. (0xA2B7, 0x38), (0xA2B8, 0x39), (0xA2CF, 0x41), (0xA2D0, 0x42),
126. (0xA2D1, 0x43), (0xA2D2, 0x44), (0xA2D3, 0x45), (0xA2D4, 0x46),
127. (0xA2D5, 0x47), (0xA2D6, 0x48), (0xA2D7, 0x49), (0xA2D8, 0x4A),
128. (0xA2D9, 0x4B), (0xA2DA, 0x4C), (0xA2DB, 0x4D), (0xA2DC, 0x4E),
129. (0xA2DD, 0x4F), (0xA2DE, 0x50), (0xA2DF, 0x51), (0xA2E0, 0x52),
130. (0xA2E1, 0x53), (0xA2E2, 0x54), (0xA2E3, 0x55), (0xA2E4, 0x56),
131. (0xA2E5, 0x57), (0xA2E6, 0x58), (0xA2E7, 0x59), (0xA2E8, 0x5A),
132. (0xA1C5, 0x5F), (0xA2E9, 0x61), (0xA2EA, 0x62), (0xA2EB, 0x63),
133. (0xA2EC, 0x64), (0xA2ED, 0x65), (0xA2EE, 0x66), (0xA2EF, 0x67),
134. (0xA2F0, 0x68), (0xA2F1, 0x69), (0xA2F2, 0x6A), (0xA2F3, 0x6B),
135. (0xA2F4, 0x6C), (0xA2F5, 0x6D), (0xA2F6, 0x6E), (0xA2F7, 0x6F),
136. (0xA2F8, 0x70), (0xA2F9, 0x71), (0xA2FA, 0x72), (0xA2FB, 0x73),
137. (0xA2FC, 0x74), (0xA2FD, 0x75), (0xA2FE, 0x76), (0xA340, 0x77),
138. (0xA341, 0x78), (0xA342, 0x79), (0xA343, 0x7A)
139. SET WChineseSimplified (64)(2) to
140. (0xA3B0, 0x30), (0xA3B1, 0x31), (0xA3B2, 0x32), (0xA3B3, 0x33),
141. (0xA3B4, 0x34), (0xA3B5, 0x35), (0xA3B6, 0x36), (0xA3B7, 0x37),
142. (0xA3B8, 0x38), (0xA3B9, 0x39), (0xA3C1, 0x41), (0xA3C2, 0x42),
143. (0xA3C3, 0x43), (0xA3C4, 0x44), (0xA3C5, 0x45), (0xA3C6, 0x46),
144. (0xA3C7, 0x47), (0xA3C8, 0x48), (0xA3C9, 0x49), (0xA3CA, 0x4A),
145. (0xA3CB, 0x4B), (0xA3CC, 0x4C), (0xA3CD, 0x4D), (0xA3CE, 0x4E),
146. (0xA3CF, 0x4F), (0xA3D0, 0x50), (0xA3D1, 0x51), (0xA3D2, 0x52),
147. (0xA3D3, 0x53), (0xA3D4, 0x54), (0xA3D5, 0x55), (0xA3D6, 0x56),
148. (0xA3D7, 0x57), (0xA3D8, 0x58), (0xA3D9, 0x59), (0xA3DA, 0x5A),
149. (0xA3DF, 0x5F), (0xA3E1, 0x61), (0xA3E2, 0x62), (0xA3E3, 0x63),
150. (0xA3E4, 0x64), (0xA3E5, 0x65), (0xA3E6, 0x66), (0xA3E7, 0x67),
151. (0xA3E8, 0x68), (0xA3E9, 0x69), (0xA3EA, 0x6A), (0xA3EB, 0x6B),
152. (0xA3EC, 0x6C), (0xA3ED, 0x6D), (0xA3EE, 0x6E), (0xA3EF, 0x6F),
153. (0xA3F0, 0x70), (0xA3F1, 0x71), (0xA3F2, 0x72), (0xA3F3, 0x73),
154. (0xA3F4, 0x74), (0xA3F5, 0x75), (0xA3F6, 0x76), (0xA3F7, 0x77),
155. (0xA3F8, 0x78), (0xA3F9, 0x79), (0xA3FA, 0x7A)

# Protocol Details

The following sections specify OLE Automation Protocol details, including abstract data models, interface method syntax, and message processing rules.

## Automation Server Details

An [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) is any [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that exposes access to its functionality through an implementation of IDispatch. This [**automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced) can be either an oleautomation interface or a "dual" interface; but, for the server to qualify as an automation server, its clients need to be able to access its functionality through calls to [IDispatch::Invoke (section 3.1.4.4)](#Section_5c2a199760d7496d8d9aed940bbb82eb).

The distinguishing characteristic of IDispatch is that clients do not need to have specific type information regarding the way the automation server exposes its functionality. Rather, clients can use a looser approach in which they provide a name for the method that they need to call, and then provide the arguments to be passed to this method in a format that allows for both [**named arguments**](#gt_0d91f93b-e04c-47d2-ae86-35d7e1bf382a) and more loosely typed arguments.

### 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 explain how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with the behavior described in this specification.

The [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) needs to maintain a direct and consistent mapping for the [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) that it recognizes for specific name requests. This mapping is permanent if, and only if, the method definition in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) file is marked with the id attribute (as specified in section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862)) or if the server documents the mapping in the component documentation. If the preceding conditions are not satisfied, the automation server MAY[<53>](#Appendix_A_53" \o "Product behavior note 53) generate the mapping on the fly, but it maintains it for the extent of its own lifetime.

The automation server maintains a dispatch mapping table that contains a list of mapping entries for each supported locale ID. Automation clients calling servers that do not have their DISPIDs specified in the IDL and that also do not have their DISPIDs specified in the server documentation cannot assume that the mapping is permanent and always query for the current mapping.

Each mapping entry contains:

* A list of names that identify the method or property and the named parameters that the server supports for it.
* A corresponding list of DISPIDs.

**Note**  The preceding conceptual data can be implemented by using a variety of techniques. Any data structure that stores this conceptual data can be used in the implementation.

### Timers

None.

### Initialization

The server MUST initialize its dispatch mapping tables.

### Message Processing Events and Sequencing Rules

This is an overview of the four methods used by the IDispatch interface. The names and [**opnums**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) of each method follow, as well as a simple description of the method.

IDispatch derives from the IUnknown interface. IDispatch servers MUST implement the methods that are defined in IUnknown, in the order in which and with the opnums with which they are specified, in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) Appendix A.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [GetTypeInfoCount](#Section_d3233e5b657f4c988a6156449c96fe16) | The GetTypeInfoCount method specifies whether the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) provides type description information.  Opnum: 3 |
| [GetTypeInfo](#Section_d1791851649142898c5725967ef7b9ed) | The GetTypeInfo method provides access to the type description information that is exposed by the automation server.  Opnum: 4 |
| [GetIDsOfNames](#Section_7166d6ffb8514216bfaa34128274a242) | The GetIDsOfNames method maps a single member name (method or property name), and an optional set of argument names, to a corresponding set of integer [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474), which can be used on subsequent calls to [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb).  Opnum: 5 |
| Invoke | The Invoke method provides access to properties and methods that are exposed by the automation server.  Opnum: 6 |

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same, unless specified otherwise.

#### IDispatch::GetTypeInfoCount (Opnum 3)

The GetTypeInfoCount method specifies whether the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) provides Type description information.

1. HRESULT GetTypeInfoCount(
2. [out] UINT\* pctinfo
3. );

**pctinfo:** MUST be set to 0 if the automation server does not provide Type description information. Otherwise, it MUST be set to 1.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### IDispatch::GetTypeInfo (Opnum 4)

The GetTypeInfo method provides access to the Type description information exposed by the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

1. HRESULT GetTypeInfo(
2. [in] UINT iTInfo,
3. [in] LCID lcid,
4. [out] ITypeInfo\*\* ppTInfo
5. );

**iTInfo:** MUST be 0.

**lcid:** MUST equal the locale ID for the Type description information to be retrieved.

**ppTInfo:** MUST be set to reference an instance of an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) that corresponds to the default nonsource interface of the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) implementing IDispatch (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)). MUST refer to the [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb) if the default nonsource interface is a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9). MUST be set to NULL if the coclass does not specify a default nonsource interface.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match the value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches the value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002000B  DISP\_E\_BADINDEX | SHOULD be returned when the value of the passed in iTInfo argument was not 0. See [MS-ERREF]. |

#### IDispatch::GetIDsOfNames (Opnum 5)

The GetIDsOfNames method maps a single member (method or property) name, and an optional set of argument names, to a corresponding set of integer [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474), which can be used on subsequent calls to [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb).

1. HRESULT GetIDsOfNames(
2. [in] REFIID riid,
3. [in, size\_is(cNames)] LPOLESTR\* rgszNames,
4. [in, range(0,16384)] UINT cNames,
5. [in] LCID lcid,
6. [out, size\_is(cNames)] DISPID\* rgDispId
7. );

**riid:** MUST equal IID\_NULL (see section [1.9](#Section_58504586e4af44a3be04f1dc281b7429)).

**rgszNames:** MUST be the array of strings to be mapped. The first string in the array MUST specify the name of a method or property that is supported by the server. Any additional strings MUST contain the names of all arguments for the method or property that is specified by the value in the first string. The mapping MUST be case-insensitive.

**cNames:** MUST equal the count of names to be mapped, and MUST[<54>](#Appendix_A_54" \o "Product behavior note 54) be between 0 and 16384.

**lcid:** MUST equal the locale ID in which to interpret the names.

**rgDispId:** MUST be an array of DISPIDs that are filled in by the server. Each DISPID corresponds, by position, to one of the names passed in *rgszNames*.

**Return Values:** The method MUST return information in an HRESULT data structure, as defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x80020006  DISP\_E\_UNKNOWNNAME | One or more names were not known. The returned array of DISPIDs MUST contain at least one DISPID\_UNKNOWN, and there MUST be one DISPID\_UNKNOWN for each entry that corresponds to an unknown name. See [MS-ERREF]. |
| 0x80020001  DISP\_E\_UNKNOWNINTERFACE | The interface identifier passed in riid is not IID\_NULL. See [MS-ERREF]. |

Exceptions Thrown: No exceptions are thrown from this method except those that are thrown by the underlying RPC Protocol specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

When GetIDsOfNames is called with more than one name, the first name (rgszNames[0]) corresponds to the member name, and subsequent names correspond to the names of member parameters.

The same name can map to different DISPIDs, depending on context. For example, a name can have a DISPID when it is used as: a member name with a particular interface, a different ID as a member of a different interface, or a different mapping for each time it appears as a parameter.

The implementation of GetIDsOfNames MUST be case-insensitive.

An implementation of the OLE Automation Protocol MAY[<55>](#Appendix_A_55" \o "Product behavior note 55) choose to implement a mapping for the parameter names that map to the index of the parameter in the member parameter list.

#### IDispatch::Invoke (Opnum 6)

The Invoke method provides access to properties and methods exposed by the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

1. HRESULT Invoke(
2. [in] DISPID dispIdMember,
3. [in] REFIID riid,
4. [in] LCID lcid,
5. [in] DWORD dwFlags,
6. [in] DISPPARAMS\* pDispParams,
7. [out] VARIANT\* pVarResult,
8. [out] EXCEPINFO\* pExcepInfo,
9. [out] UINT\* pArgErr,
10. [in] UINT cVarRef,
11. [in, size\_is(cVarRef)] UINT\* rgVarRefIdx,
12. [in, out, size\_is(cVarRef)] VARIANT\* rgVarRef
13. );

**dispIdMember:** MUST equal the [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) of the method or property to call.

**riid:** MUST equal IID\_NULL (see section [1.9](#Section_58504586e4af44a3be04f1dc281b7429)).

**lcid:** MUST equal a locale ID supported by the automation server. This value SHOULD be used by the automation server if any of the arguments are strings whose meaning is dependent on a specific locale ID. If no such strings are present in the arguments the server SHOULD ignore this value.

**dwFlags:**  MUST be a combination of the bit flags specified in the following table.

**Note**  The value MUST specify one and only one of the following bit flags: DISPATCH\_METHOD, DISPATCH\_PROPERTYGET, DISPATCH\_PROPERTYPUT, or DISPATCH\_PROPERTYPUTREF.

| Value | Meaning |
| --- | --- |
| DISPATCH\_METHOD  0x00000001 | The member is invoked as a method. |
| DISPATCH\_PROPERTYGET  0x00000002 | The member is retrieved as a property or data member. |
| DISPATCH\_PROPERTYPUT  0x00000004 | The member is changed as a property or data member. |
| DISPATCH\_PROPERTYPUTREF  0x00000008 | The member is changed by a reference assignment, rather than by a value assignment. This flag is valid only when the property accepts a reference to an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). |
| DISPATCH\_zeroVarResult  0x00020000 | MUST specify that the client is not interested in the actual pVarResult [out] argument. On return the *pVarResult* argument MUST point to a VT\_EMPTY variant, with all reserved fields set to 0. |
| DISPATCH\_zeroExcepInfo  0x00040000 | MUST specify that the client is not interested in the actual pExcepInfo [out] argument. On return *pExcepInfo* MUST point to an EXCEPINFO structure, with all scalar fields set to 0 and all BSTR fields set to NULL. |
| DISPATCH\_zeroArgErr  0x00080000 | MUST specify that the client is not interested in the actual pArgErr [out] argument. On return, *pArgErr* MUST be set to 0. |

**pDispParams:**  MUST point to a [DISPPARAMS](#Section_144b00dd4c2f4b35a28fc17f591b990c) structure that defines the arguments passed to the method. Arguments MUST be stored in pDispParams->rgvarg in reverse order, so that the first argument is the one with the highest index in the array. [**Byref arguments**](#gt_969e6685-c90a-4da6-99be-1a25efb6d1cd) MUST be marked in this array as VT\_EMPTY entries, and stored in *rgVarRef* instead. For more information, see section 2.2.33.

**pVarResult:** MUST point to a VARIANT that will be filled with the result of the method or property call.

**pExcepInfo:** If this value is not null and the return value is DISP\_E\_EXCEPTION, this structure MUST be filled by the automation server. Otherwise, it MUST specify a 0 value for the **scode** and **wCode** fields, and it MUST be ignored on receipt.

**pArgErr:** If this value is not null and the return value is DISP\_E\_TYPEMISMATCH or DISP\_E\_PARAMNOTFOUND, this argument MUST equal the index (within pDispParams->rgvarg) of the first argument that has an error. Otherwise, it MUST be ignored on receipt.

**cVarRef:** MUST equal the number of byref arguments passed in *pDispParams*.

**rgVarRefIdx:** MUST contain an array of *cVarRef* unsigned integers that holds the indices of the byref arguments marked as VT\_EMPTY entries in pDispParams->rgvarg.

**rgVarRef:** MUST contain the byref arguments as set by the client at the time of the call, and by the server on successful return from the call. Arguments in this array MUST also be stored in reverse order, so that the first byref argument has the highest index in the array.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x80020009  DISP\_E\_EXCEPTION | The application needs to raise an exception. In this case, the structure passed in *pExcepInfo* MUST be filled in with a nonzero error code. See [MS-ERREF]. |
| 0x80020004  DISP\_E\_PARAMNOTFOUND | One of the parameter DISPIDs does not correspond to a parameter on the method. In this case, pArgErr MUST be set to the first argument that contains the error. See [MS-ERREF]. |
| 0x80020005  DISP\_E\_TYPEMISMATCH | One or more of the arguments could not be coerced into the type of the corresponding formal parameter. The index within rgvarg of the first parameter with the incorrect type MUST be returned in the pArgErr parameter. For more information, see section [3.1.4.4.4](#Section_5c01ab3cf71944ccbb45d36850cf4c5b) and [MS-ERREF]. |
| 0x8002000E  DISP\_E\_BADPARAMCOUNT | The number of elements provided to DISPPARAMS is different from the number of arguments accepted by the method or property. See [MS-ERREF]. |
| 0x80020008  DISP\_E\_BADVARTYPE | One of the arguments in rgvarg is not a valid variant type. See [MS-ERREF]. |
| 0x80020003  DISP\_E\_MEMBERNOTFOUND | The requested member does not exist, or the call to Invoke tried to set the value of a read-only property. See [MS-ERREF]. |
| 0x80020007  DISP\_E\_NONAMEDARGS | At least one named argument was provided for methods with a vararg parameter (see section [3.1.4.4.3](#Section_be6e35f6932741649bdeffcd0fa0e07d)), for which named arguments are illegal. See [MS-ERREF]. |
| 0x8002000A  DISP\_E\_OVERFLOW | One of the arguments in rgvarg could not be coerced to the type of its corresponding formal parameter. See [MS-ERREF]. |
| 0x80020001  DISP\_E\_UNKNOWNINTERFACE | The interface identifier passed in riid is not IID\_NULL. See [MS-ERREF]. |
| 0x8002000F  DISP\_E\_PARAMNOTOPTIONAL | A required parameter was omitted. See [MS-ERREF]. |

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC Protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

##### Invoke Consistency Checks

The following conditions MUST hold:

* If pDispParams->cNamedArgs is nonzero, rgdispidNamedArgs MUST contain the [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) corresponding to the [**named arguments**](#gt_0d91f93b-e04c-47d2-ae86-35d7e1bf382a) that are recognized by the server in the context of the current method or property (identified by the dispIdMember argument passed to Invoke).
* The last cArgs-cNamedArgs VARIANT values from rgvargs correspond to the positional arguments passed to the method.
* There are no VT\_BYREF variant values in rgvargs.
* All variant values in rgVarRef have the VT\_BYREF bit flag set.
* Each VARIANT contained within the pDispParams->rgvarg array, whose index corresponds to an entry in *rgVarRefIdx*, is set to VT\_EMPTY.

##### Invoke Argument-Parameter Mapping

The two arrays of VARIANTs that appear in this method, pDispParams->rgvarg and *rgVarRef*, are two halves of the same whole.

The first array, pDispParams->rgvarg, MUST contain only the [in] argument values, whose updates do not need to be reflected on the client side. The second array, rgVarRef, MUST contains all the [in, out] or [out] arguments, passed by reference, which need to update client-side state upon return. The elements in this array MUST be mapped to positional or named arguments through *rgVarRefIdx*.

In addition, any parameters that have the lcid or retval attributes MUST NOT be packed in the pDispParams->rgvarg or *rgVarRef*. Instead the "lcid" argument MUST be used to specify the third argument (lcid) to [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb). Also, the "retval" argument MUST be set from the sixth argument (pVarResult) to IDispatch::Invoke.

##### Handling Default Value and Optional Arguments

For any arguments that the [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) does not semantically need to specify that correspond to *defaultvalue* parameters, the automation client MUST[<56>](#Appendix_A_56" \o "Product behavior note 56) use the value specified in the **defaultvalue** attribute.

For any arguments that the automation client does not semantically need to specify, that correspond to optional parameters, the automation client MUST use the "optional argument marker". The "optional argument marker" is a [VARIANT](#Section_a6a540af38ac48bebd4092e2c01e9aa6) value with the **discriminant** field set to VT\_ERROR and the **scode** field set to DISP\_E\_PARAMNOTFOUND.

A method with the **vararg** attribute does not specify optional arguments by position. To pass arguments that do not correspond to positional arguments, the automation client MUST create a one-dimensional [SAFEARRAY](#Section_04e72b3f573145089bb4de29fbd0f781) of VARIANTs whose length is the number of additional arguments, package the values of the arguments as VARIANTs in the SAFEARRAY, and pass the SAFEARRAY as the first argument in *pDispParams*. The automation client MUST pass an empty SAFEARRAY as the final argument of a **vararg** method if it does not need to specify any optional arguments.

##### Argument Coercion

The [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) MAY[<57>](#Appendix_A_57" \o "Product behavior note 57) attempt to convert the arguments passed in pDispParams to the expected types of the formal parameters of the method or property that is called. If no possible coercion exists, the server MUST return DISP\_E\_TYPEMISMATCH.

### Timer Events

None.

### Other Local Events

None.

## Automation Client Details

### 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 explain how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with the behavior described in this specification.

For every locale ID that an [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) is using, it MUST maintain a dispatch mapping table that contains a list of mapping entries.

Each mapping entry MUST contain:

* The list of names that identify the method or property, and the named parameters that the client intends to use for calls to those operations.
* A corresponding list of [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474).

**Note**  The preceding conceptual data can be implemented by using a variety of techniques. Any data structure that stores this conceptual data can be used in the implementation.

### Timers

None.

### Initialization

If the client caches the mapping between the [**automation server's**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) member names and the corresponding [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474), the client MUST initialize the cache at this time.

### Message Processing and Sequencing Rules

If the client does not have prior knowledge of the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) [**DISPIDs**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474), it MUST call [IDispatch::GetIDsOfNames (section 3.1.4.3)](#Section_7166d6ffb8514216bfaa34128274a242) before calling [IDispatch::Invoke (section 3.1.4.4)](#Section_5c2a199760d7496d8d9aed940bbb82eb). The client MUST call GetIDsOfNames at least once for each automation member (see section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862) ) that it needs to call.

The sequence of events in this case MUST be as follows:

* The [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) calls IDispatch::GetIDsOfNames to obtain the mapping between the automation member name and any of its [**named arguments**](#gt_0d91f93b-e04c-47d2-ae86-35d7e1bf382a), and their corresponding DISPIDs (see section [2.2.32](#Section_b0b43e39b0804edda26d7134075c75cd)).
* The automation client potentially caches this mapping in the dispatch mapping table that corresponds to the locale ID value it specified.
* The automation client invokes the operation by using IDispatch::Invoke one or more times.

If the client uses the DISPIDs retrieved at compile time (from the documentation of the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa), or from the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) that describes the [**Automation interface**](#gt_db2ca497-c239-403c-81a7-78b0cecb5ced)), there are no sequencing rules.

### Timer Events

None.

### Other Local Events

None.

## IEnumVARIANT Server Details

By implementing the IEnumVARIANT interface, a server provides a method for enumerating a collection of variants, including heterogeneous collections of objects. This interface SHOULD be implemented by COM servers that expose collections of objects.

An [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) SHOULD expose this functionality by implementing a \_NewEnum method with a [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) of DISPID\_NEWENUM, as specified in section [2.2.32.1](#Section_cb9d0131c6bd463d9c407264856a10c5).

IEnumVARIANT MUST NOT impose restrictions on the semantics associated with the collection it manages.

### Abstract Data Model

An [IEnumVARIANT server](#Section_716d04d1cd1640659b191b8808b3df31) MUST maintain a sequence of elements and the current position in the enumeration:

* Static servers: MUST maintain a static sequence of elements throughout its lifetime, and the current position in the sequence.
* Semi-static servers: MUST maintain a static sequence of elements, and the current position in the sequence. The sequence MUST be updated upon the call to Reset.
* Dynamic servers: MUST maintain the current position in the sequence and server-specific state that allows it to dynamically retrieve the next elements requested by a client.

### Timers

None.

### Initialization

The server MUST initialize the current position. A static and semi-static server MUST initialize the sequence of elements it manages.

### Message Processing and Sequencing Rules

This is an overview of the four methods that are used by the [IEnumVARIANT](#Section_716d04d1cd1640659b191b8808b3df31) interface. The names and [**opnums**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) of each method are given as follows, as well as a simple description of the method.

The IEnumVARIANT derives from the IUnknown interface. IEnumVARIANT servers MUST implement the methods defined in IUnknown in the order and with the opnums with which they are specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) Appendix A.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [Next](#Section_ba5dded9503a4c29884138174de2295d) | The IEnumVARIANT::Next method returns the requested items that are next in the enumeration sequence.  Opnum: 3 |
| [Skip](#Section_cc090461ef064bef9b4cc51a11b29aea) | The IEnumVARIANT::Skip method skips over the requested number of elements in the enumeration sequence.  Opnum: 4 |
| [Reset](#Section_fc80414cad314b0ebac8c874ccc11725) | The IEnumVARIANT::Reset method resets the enumeration sequence to the beginning.  Opnum: 5 |
| [Clone](#Section_94aaabbd5e8941d08acf1dd200b39288) | The IEnumVARIANT::Clone method creates a copy of the current state of the enumeration.  Opnum: 6 |

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same unless specified otherwise.

#### IEnumVARIANT::Next (Opnum 3)

The IEnumVARIANT::Next method returns up to the number of requested items that occur next in the enumeration sequence.

1. HRESULT Next(
2. [in] ULONG celt,
3. [out, size\_is(celt), length\_is(\*pCeltFetched)]
4. VARIANT\* rgVar,
5. [out] ULONG\* pCeltFetched
6. );

**celt:** MUST be set to the maximum number of elements to return. The value MUST NOT be 0.

**rgVar:** MUST be set to an array of elements that are returned from the enumeration sequence.

**pCeltFetched:** MUST be set to the number of elements successfully returned. This number MUST be equal to the value in *celt*, unless the end of the enumeration sequence is encountered.

**Return Values:** The method MUST return the information in an **HRESULT** data structure, which is defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

A static or semistatic IEnumVARIANT server MUST retrieve the next *celt* elements from the sequence and fill in *rgVar* up to the *celt* elements or up to the remaining number of elements that are not yet enumerated.

A dynamic server MUST use its server-specific state to retrieve the next elements.

In all cases, the server MUST:

* Set *pCeltFetched* with the number retrieved.
* Update the current position in the sequence.
* Return a status of 1 (S\_FALSE) if *pCeltFetched* is not equal to *celt*.

#### IEnumVARIANT::Skip (Opnum 4)

The IEnumVARIANT::Skip method skips over the requested number of elements in the enumeration sequence.

1. HRESULT Skip(
2. [in] ULONG celt
3. );

**celt:** MUST be set to the maximum number of elements to skip over.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

A static or semi-static [IEnumVARIANT server](#Section_716d04d1cd1640659b191b8808b3df31) MUST update the current position in the sequence by either celt elements, or the number of elements remaining, whichever is smaller.

A dynamic server MUST use its server-specific state to affect the dynamic collection it manages, and MUST update the current position in the sequence by either celt elements, or the number of elements remaining, whichever is smaller.

In all cases, the server MUST return 1 (S\_FALSE), if the current position was updated by less than celt elements.

#### IEnumVARIANT::Reset (Opnum 5)

The IEnumVARIANT::Reset method resets the enumeration sequence to the beginning.

1. HRESULT Reset();

This method has no parameters.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

A static server MUST reset its current position in the sequence.

A semi-static or dynamic server MUST update the sequence of elements it maintains and MUST reset the current position in the sequence to the beginning.

#### IEnumVARIANT::Clone (Opnum 6)

The IEnumVARIANT::Clone method creates a copy of the current state of the enumeration.

1. HRESULT Clone(
2. [out] IEnumVARIANT\*\* ppEnum
3. );

**ppEnum:** MUST be set to an instance of the enumeration.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

The server MUST create a new [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that implements IEnumVARIANT. The state of the newly created IEnumVARIANT server MUST be set according to the type of the current server:

* If the original server is a static or semi-static server, the state of the new COM server MUST be based on the current state of the enumeration.
* If the original server is a dynamic server, the state of the new COM server MUST be based on the current state of the underlying collection, and the current position in the sequence.

### Timer Events

None.

### Other Local Events

None.

## IEnumVARIANT Client Details

An IEnumVARIANT client iterates over the elements of a collection managed by an [IEnumVARIANT server](#Section_716d04d1cd1640659b191b8808b3df31).

### Abstract Data Model

None.

### Timers

None.

### Initialization

None.

### Message Processing and Sequencing Rules

An [IEnumVARIANT client](#Section_079100caf62f47028861a1f64e66e9e2) MUST retrieve an instance of the server by:

* Some DCOM server-specific method.
* Calling \_NewEnum (see section [2.2.32.1](#Section_cb9d0131c6bd463d9c407264856a10c5)) on an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

The client MUST call [IEnumVARIANT::Next](#Section_ba5dded9503a4c29884138174de2295d) to enumerate over the elements of the collection.

The client MUST call [IEnumVARIANT::Skip](#Section_cc090461ef064bef9b4cc51a11b29aea) if it needs to omit collection elements from the enumeration.

The client MUST call [IEnumVARIANT::Reset](#Section_fc80414cad314b0ebac8c874ccc11725) if it needs to restart the enumeration.

The client MUST call [IEnumVARIANT::Clone](#Section_94aaabbd5e8941d08acf1dd200b39288) if it needs to save the current state of the enumeration.

### Timer Events

None.

### Other Local Events

None.

## ITypeComp Server Details

The ITypeComp interface defines methods that map names to types and type members.

### 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 specification does not mandate that implementations adhere to this model so long as their external behavior is consistent with the behavior described in this specification.

An [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) MUST maintain a "binding server" reference to either an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b) or an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa). The binding context of the ITypeComp server (see section [3.5.4.1.1](#Section_cc88e6254b3a4cf9896626bdc913f62d)) MUST be the same as the binding context of the binding server.

The reference to an ITypeLib server or ITypeInfo server MUST be permanent over the lifetime of the ITypeComp server.

### Timers

None.

### Initialization

The server MUST initialize its [ITypeLib](#Section_5daecf67bc6e4e17bcf8797bdba1748b) or [ITypeInfo](#Section_99504cf916d8401ea87383b85d1ee4aa) reference and maintain the reference over its lifetime. The ITypeLib or ITypeInfo instance referenced by the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) MUST be initialized as specified in section [3.11.3](#Section_34d15e8d68494e7d818c2478ff25da06) or section [3.7.3](#Section_752034952f384211835a7e76c48d14b8).

### Message Processing Events and Sequencing Rules

This is an overview of the two methods that are used by the [ITypeComp](#Section_7894019fde1e455eb2aa3b899c2e50f6) interface. The names and [**opnums**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) of each method are given as follows, in addition to simple descriptions of the methods.

ITypeComp derives from the IUnknown interface. ITypeComp servers MUST implement the methods that are defined in IUnknown, in the order in which and with the opnums with which they are specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) Appendix A.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [Bind](#Section_476f00da080640d9bbf36059154abbb7) | The Bind method retrieves a type member whose name corresponds to a specified string.  Opnum: 3 |
| [BindType](#Section_cf61a786b1814267bb6a0987eeb17b38) | The BindType method retrieves a reference to an [**Automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) whose name corresponds to a specified string.  Opnum: 4 |

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same, unless specified otherwise.

#### ITypeComp::Bind (Opnum 3)

The Bind method retrieves [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) server instances and related structures that provide access to a method, property or data member whose name corresponds to a specified string.

1. HRESULT Bind(
2. [in] LPOLESTR szName,
3. [in] ULONG lHashVal,
4. [in] WORD wFlags,
5. [out] ITypeInfo\*\* ppTInfo,
6. [out] DESCKIND\* pDescKind,
7. [out] LPFUNCDESC\* ppFuncDesc,
8. [out] LPVARDESC\* ppVarDesc,
9. [out] ITypeComp\*\* ppTypeComp,
10. [out] DWORD\* pReserved
11. );

**szName:** MUST be set to a string.

**lHashVal:** MUST be set to the hash value that corresponds to the value of *szName* (as specified in section [2.2.51](#Section_7a8ed4c314a4433cbaa5b6ec88135352)) or 0. Whether 0 or a computed hash value is specified for this argument, the server MUST return the same values in *ppTInfo*, *pDescKind*, *ppFuncDesc*, *ppVarDesc*, and *ppTypeComp*.

**wFlags:** MUST be set to a value of the [INVOKEKIND](#Section_a0d3598da3ee440187fd17a7031b0b9a) enumeration, as specified in section 2.2.14, or 0.

**ppTInfo:** MUST be set to a reference to the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) instance that corresponds to the element whose name matches the value of *szName* or NULL, as specified in sections [3.5.4.1.2](#Section_3ca3ffe31f9547159b54574a300ffd88) and [2.2.50](#Section_ef05bc9b062d467bad880f19e3e545f6). MUST be set to NULL if *szName* does not match the name of any element in the binding context (see section [3.5.4.1.1](#Section_cc88e6254b3a4cf9896626bdc913f62d)).

**pDescKind:** MUST be set to one of the following values of the [DESCKIND enumeration (section 2.2.22)](#Section_a7d2404b99344fcca69bd396fb51141a):

* MUST be set to DESCKIND\_FUNCDESC if the values of *szName* and *wFlags* specify a method or property accessor method in the binding context of the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6).
* MUST be set to DESCKIND\_VARDESC if the values of *szName* and *wFlags* specify a data member of an enumeration, module, or [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) in the binding context of the ITypeComp server, or if the binding server is an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b) and *szName* specifies the name of an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) in the binding context of the ITypeComp server.
* MUST be set to DESCKIND\_TYPECOMP if the value of *szName* specifies an enumeration or module in the binding context of the ITypeComp server.
* MUST be set to DESCKIND\_IMPLICITAPPOBJ if the binding server is an ITypeLib server, the value of *szName* specifies an element in the binding context of an appobject coclass, and the appobject coclass is in the binding context of the ITypeComp server.
* Otherwise, MUST be set to DESCKIND\_NONE.

**ppFuncDesc:** MUST be set to a FUNCDESC that describes the method or property whose name matches the value of *szName*, if *pDescKind* is DESCKIND\_FUNCDESC. Otherwise, MUST be set to NULL.

**ppVarDesc:** MUST be set to a value that is specified by the value of *pDescKind*.

* DESCKIND\_VARDESC: MUST be set to a VARDESC that describes a data member of an enumeration, module, or ODL dispinterface if the name of the data member matches *szName*.
* DESCKIND\_IMPLICITAPPOBJ: MUST be set to a VARDESC that describes the appobject coclass whose binding context contains a member whose name matches *szName*.
* DESCKIND\_FUNCDESC, DESCKIND\_TYPECOMP, DESCKIND\_NONE: MUST be set to NULL.

**ppTypeComp:** MUST be set to a reference to an ITypeComp server instance that provides access to the binding context of the bound element, if *pDescKind* is DESCKIND\_TYPECOMP or DESCKIND\_IMPLICITAPPOBJ. Otherwise, MUST be set to NULL.

**pReserved:** MUST be set to 0.

**Return Values:** The method MUST return information in an HRESULT data structure, which is defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802C  TYPE\_E\_AMBIGUOUSNAME | The values of *szName* and *wFlags* match more than one element in the binding context. See [MS-ERREF]. |
| 0x80028CA0  TYPE\_E\_TYPEMISMATCH | The value of *szName* matched a single element in the binding context, but the **INVOKEKIND** value that is associated with that element did not match the value of *wFlags*. See [MS-ERREF]. |

##### Binding Context

The binding context of [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) is determined by its associated [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) or [**Automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) server. The binding context defines the set of members and members of inherited types that can be referenced by name.

###### Automation Type Library Binding Context

The binding context of an [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) MUST consist of the following elements:

* The names of all enum types of the module, if present, and of all appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) types.
* The data members of each enumeration member of its type information table.
* The data and method members of the module member of its type information table, if there is one.
* The binding context of each coclass defined in the [**Automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a), if the coclass was declared with the **[appobject]** attribute (an appobject coclass).

###### Automation Type Description Binding Context

The binding context of an [**Automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) MUST be specified by its associated TYPEKIND value.

| TYPEKIND value | Binding context |
| --- | --- |
| TKIND\_ENUM, TKIND\_RECORD, TKIND\_UNION | The binding context MUST consist of the members of the data member table. |
| TKIND\_MODULE | The binding context MUST consist of the members of the method and data member tables. |
| TKIND\_COCLASS | The binding context MUST consist of the binding context of the default nonsource interface in the interface table (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)). A [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) whose interface table does not include an interface declared with the [default] attribute and without the [source] attribute (that is, a "default nonsource interface") MUST have an empty binding context. |
| TKIND\_DISPATCH | The binding context MUST consist of the members of the dispatch method and data member tables. The binding context of an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2) also includes the members of the method tables of IDispatch and IUnknown. |
| TKIND\_INTERFACE | The binding context MUST consist of the members of the method table, plus the members of the binding context specified by the entry in the interface table (if present). |
| TKIND\_ALIAS | The binding context MUST be empty. |

##### Types Returned with Bound Elements

The following rules specify the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) instance that is returned with each element in the binding context whose name matches a specified string according to the string-matching criteria specified in section [2.2.50](#Section_ef05bc9b062d467bad880f19e3e545f6).

###### Types Returned with ITypeLib Members

The [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) returned with a member of an [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) MUST be specified by the kind of element whose name was matched.

* MUST be the ITypeInfo server instance that represents an enumeration, if the matched element was a member of the enumeration's data member table.
* MUST be the ITypeInfo server instance that represents the module declared with the automation type library, if the matched element was a member of the module's method or data member tables.
* MUST be the ITypeInfo server instance that represents the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc), if the matched element was an appobject coclass with a default nonsource interface or a member of its default nonsource interface.
* MUST be NULL, if the matched element was an enumeration or the module declared with the automation type library.

###### Types Returned with ITypeInfo Members

The [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) reference returned with a member of an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) MUST be specified by its associated TYPEKIND value.

| TYPEKIND value | Returned type |
| --- | --- |
| TKIND\_ENUM, TKIND\_RECORD, TKIND\_UNION, TKIND\_MODULE: | MUST be the ITypeInfo server instance whose corresponding Bind method was called. |
| TKIND\_COCLASS | MUST be the ITypeInfo server instance that would be returned by the Bind method associated with its default nonsource interface. |
| TKIND\_DISPATCH | MUST be the ITypeInfo server instance that represents IDispatch, if the ITypeInfo server instance whose corresponding Bind method was called represents an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2), and the bound element is a method member of IDispatch or IUnknown.  MUST be the ITypeInfo server instance whose corresponding Bind method was called in all other cases. |
| TKIND\_INTERFACE | MUST be the ITypeInfo server whose method table contains the matching element. |
| TKIND\_ALIAS: | MUST be NULL. |

#### ITypeComp::BindType (Opnum 4)

The BindType method retrieves a reference to an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) whose name corresponds to a specified string.

1. HRESULT BindType(
2. [in] LPOLESTR szName,
3. [in] ULONG lHashVal,
4. [out] ITypeInfo\*\* ppTInfo
5. );

**szName:** MUST be set to a string.

**lHashVal:** MUST be the hash value associated with the value of *szName* as specified in section [2.2.51](#Section_7a8ed4c314a4433cbaa5b6ec88135352), or 0. The server MUST return the same values in *ppTInfo* in either case.

**ppTInfo:** MUST be set to a reference to an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) instance, or NULL.

If the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) is associated with an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b), *ppTInfo* MUST specify a type defined in its [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) whose name matches the value of *szName* according to the string-matching criteria specified in section [2.2.50](#Section_ef05bc9b062d467bad880f19e3e545f6), or be set to NULL if no such type exists.

MUST be set to NULL if the ITypeComp server is associated with an ITypeInfo server.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

### Timer Events

None.

### Other Local Events

None.

## ITypeComp Client Details

### Abstract Data Model

None.

### Timers

None.

### Initialization

None.

### Message Processing Events and Sequencing Rules

To retrieve a reference to an [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6), a client MUST call [ITypeLib::GetTypeComp (section 3.11.4.6)](#Section_636b2c755fd643b2bfca5894da8623eb), [ITypeInfo::GetTypeComp (section 3.7.4.2)](#Section_2d345f2a4eb3452bb1f6fca33cfe16e6), or [ITypeComp::Bind (section 3.5.4.1)](#Section_476f00da080640d9bbf36059154abbb7).

The protocol specifies no additional sequencing rules.

### Timer Events

None.

### Other Local Events

None.

## ITypeInfo Server Details

An [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) server is a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that provides access to an automation type through an implementation of ITypeInfo.

### 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 explain how the protocol behaves. This specification does not mandate that implementations adhere to this model so long as their external behavior is consistent with that described in this specification.

An [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) server provides a concrete representation of a type element that has been defined or referenced in an [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a). An automation type description can describe the following type elements: an enumeration, a data-only structure, a union, a typedef alias, an interface, a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5), a module, or a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c).

#### Common Automation Type Description Elements

Every [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) server maintains the following data elements:

* The name of the type.
* A [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33) value that specifies the type that the automation type description describes, as specified in section 2.2.17.
* A value or structure that specifies the attributes declared with the type in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824), as specified in section [2.2.49](#Section_0c5a99a3dc4b4f7ea109c695b6702b85).
* A value that specifies the size, in bytes, of an instance of the type.
* A reference to the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) that contains the automation type description in its type information table.
* A reference to an [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) whose binding context (see section [3.5.4.1.1](#Section_cc88e6254b3a4cf9896626bdc913f62d)) is specified by the members of the automation type description.
* A set of implementation-specific documentation values.[<58>](#Appendix_A_58" \o "Product behavior note 58)

An automation type description can include the following tables of member elements, as specified in the remainder of this topic.

A method table specifies the methods that are defined by an interface. Each element of a method table includes the following:

* The name of the method.
* The [MEMBERID](#Section_ace8758fee2b4cb68645973994d12530) of the method.
* A value or structure that specifies the attributes declared with the method in the IDL (as specified in section 2.2.49) and other method information, as specified in section [2.2.42](#Section_d3349d25e11d4095ba86de3fda178c4e).
* An [ELEMDESC](#Section_e14ff3cf034a4884a498fc7586f7160c) that specifies the return type of the method.
* A parameter table (see below) that specifies the parameters declared with the method in the IDL, as specified in section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862).

A data member table describes data-only members of the type. Each element of a data member table includes the following:

* The name of the data member.
* The MEMBERID of the data member.
* A value or structure that specifies the attributes declared with the data member in the IDL (as specified in section 2.2.49.5) and other data member information, as specified in section [2.2.43](#Section_ae7791d243994dffb7c6b0d4f3dce982).
* An ELEMDESC that specifies the type of the data member.

An interface table contains references to interfaces and [**dispinterfaces**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5). Each element of an interface table includes the following:

* The HREFTYPE of the referenced interface or dispinterface.
* A reference to the automation type description of the interface or dispinterface.
* A value or structure that specifies the attributes declared with the interface reference in the IDL (as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)) and other information, as specified in section [2.2.13](#Section_2743c240436242bf9482a910ac1857f7).

A parameter table contains references to method parameters. The order of the elements in the parameter table follows the parameter ordering criteria specified in section [2.2.49.6](#Section_ff2bd74bcb4d48b3ab896bb32cda3833). Each element of a parameter table includes the following:

* The name of the parameter.
* An ELEMDESC that specifies the type of the parameter.
* A value or structure that specifies the attributes declared with the parameter in the IDL (as specified in section 2.2.49.6) and other data member information, as specified in section [2.2.40](#Section_a965ce8e6c064d7cab302f14b1d8488a).

#### TYPEKIND Dependent Automation Type Description Elements

Every [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) maintains the following data elements specified by its [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33) value.

| TYPEKIND value | Type description elements |
| --- | --- |
| TKIND\_ENUM | A data member table that specifies the values defined by the enumeration. |
| TKIND\_RECORD and TKIND\_UNION | A data member table that specifies the fields of the structure or union. |
| TKIND\_INTERFACE | A method table that specifies the method members of the interface.  An interface table that specifies the interfaces inherited by the described interface.  A reference to the [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb) if the described type is a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) (see section [2.2.49.4.2](#Section_bd30db0d6c384d549c4467c0e9d25551)). |
| TKIND\_DISPATCH | A method table that specifies the method members defined by the [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5). If the type is an [**ODL dispinterface**](#gt_544446d5-79ab-4b08-85c4-214f1b64fdf2), the method table includes the elements defined with the "methods" keyword. If the type is a partner dispinterface, the method table includes the methods defined by the dual interface. If the type is a [**reference dispinterface**](#gt_dc320a3c-71b6-4055-bfd1-d9fa6f3f770f), the method table is empty.  A "dispatch" method table that specifies the method members available through [ITypeComp::Bind](#Section_476f00da080640d9bbf36059154abbb7). If the type is an ODL dispinterface, the dispatch method table includes the elements defined with the "methods" keyword. If the type is a reference dispinterface, the dispatch method table includes all members defined by its referenced [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408) and its base interfaces (including IUnknown), and includes separate entries for property accessor methods with the same [MEMBERID](#Section_ace8758fee2b4cb68645973994d12530) (see 2.2.35). If the type is a partner dispinterface, the dispatch method table includes all members defined by the dual interface or its base interfaces (including IUnknown), and includes separate entries for property accessor methods with the same MEMBERID (see section 2.2.35).  A data member table that specifies the data members defined by the dispinterface and available through [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb). If the type is an ODL dispinterface, the data member table includes the elements defined with the properties keyword. Otherwise, it is empty.  An interface table that specifies a single interface. If the type is an ODL dispinterface or a partner dispinterface, the specified interface is IDispatch. If the type is a reference dispinterface, the specified interface is its referenced DCOM interface (see section [2.2.49.4.3](#Section_07829751cb564eec88ef476f8a09dd43).)  A reference to the [**partner interface**](#gt_3610e61d-bd5c-454f-992a-0a020995e66b) if the described type is a dual interface (see section 2.2.49.4.2). |
| TKIND\_COCLASS | An interface table that specifies the interfaces and dispinterfaces defined or referenced by the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc). |
| TKIND\_MODULE | A method table that specifies the local-only methods defined by the module.  A data member table that specifies the symbolic constants and static variables defined by the module (see section [2.2.49.9](#Section_82f9465bae46474e87ffe65e9751affb)). |
| TKIND\_ALIAS | A reference to the [TYPEDESC](#Section_95bb92a7f783477facbcc947d754fa8b) that specifies the predefined type of the alias. |

### Timers

None.

### Initialization

The server MUST initialize its name, [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33), [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) reference, and [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) reference, and specify the size of an instance of the type in bytes. These values MUST NOT change over the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) lifetime.

The server MUST initialize the representation of the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) attributes declared with the type and its implementation-specific documentation values. The attribute representation and documentation values MUST NOT change over the server's lifetime.

The server MUST initialize the following elements according to its TYPEKIND value, as specified in the first paragraph of this section:

* Data member table
* Method tables (including parameter tables)
* Interface table
* Partner interface reference or [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb) reference
* Alias type reference

These tables and references MUST NOT change over the server's lifetime.

### Message Processing Events and Sequencing Rules

The ITypeInfo interface derives from the IUnknown interface. [ITypeInfo servers](#Section_99504cf916d8401ea87383b85d1ee4aa) MUST implement the methods that are defined in IUnknown in the order in which, and with the opnums with which, they are specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) Appendix A.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [GetTypeAttr](#Section_b96292a8c06e4b9c905e129b95697ee4) | The GetTypeAttr method retrieves a **TYPEATTR** structure that contains information about the type described by the ITypeInfo server.  Opnum: 3 |
| [GetTypeComp](#Section_2d345f2a4eb3452bb1f6fca33cfe16e6) | The GetTypeComp method retrieves a reference to the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) instance associated with the ITypeInfo server.  Opnum: 4 |
| [GetFuncDesc](#Section_d54aca0905654fa8b5e4cf87723a89ed) | The GetFuncDesc method retrieves a [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e) structure that contains information about a member of the ITypeInfo server's method or dispatch method table.  Opnum: 5 |
| [GetVarDesc](#Section_a6b5857a38e446cda2e4bfbc7e21c787) | The GetVarDesc method retrieves a VARDESC structure that contains information about a member of the ITypeInfo server’s data member table.  Opnum: 6 |
| [GetNames](#Section_78533af1f18640f78b3dc65ba9c6ee3c) | The GetNames method retrieves the data member name, or the method and parameter names associated with a specified MEMBERID.  Opnum: 7 |
| [GetRefTypeOfImplType](#Section_7225fbad7ad0458bb1492e854364fbfd) | The GetRefTypeOfImplType method retrieves the HREFTYPE corresponding to the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) of an interface that is inherited, implemented, or referenced by the ITypeInfo server.  Opnum: 8 |
| [GetImplTypeFlags](#Section_4232aaa957a844bfb0d0a02dbdb3e9bc) | The GetImplTypeFlags method retrieves the IMPLTYPEFLAGS values associated with an interface member of a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc).  Opnum: 9 |
| Opnum10NotUsedOnWire | Reserved for local use.  Opnum: 10 |
| Opnum11NotUsedOnWire | Reserved for local use.  Opnum: 11 |
| [GetDocumentation](#Section_2ea2f705bc334cecbbc7613d6ae0f0c6) | The GetDocumentation method retrieves the documentation resources associated with a type member.  Opnum: 12 |
| [GetDllEntry](#Section_d82eb39db2184484a1587b582ab65e5c) | The GetDllEntry method retrieves values associated with a local-only method defined in a module.  Opnum: 13 |
| [GetRefTypeInfo](#Section_8428b8f844574fab99fe38a8c6ab217b) | The GetRefTypeInfo method retrieves an automation type description that describes a type that is inherited or referenced by the ITypeInfo server.  Opnum: 14 |
| Opnum15NotUsedOnWire | Reserved for local use.  Opnum: 15 |
| [CreateInstance](#Section_b50cde468de74809ac5e884e3500b93c) | The CreateInstance method creates a new instance of a type that describes a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) (coclass).  Opnum: 16 |
| [GetMops](#Section_2dbee2076c704c7bb460c0488a13256d) | The GetMops method has no effect when called across the wire.  Opnum: 17 |
| [GetContainingTypeLib](#Section_1a843f04e55a47dd85026ba3cd94b161) | The GetContainingTypeLib method retrieves the [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b) instance whose type information table contains the ITypeInfo instance, and the index of the ITypeInfo instance within the type information table.  Opnum: 18 |
| Opnum19NotUsedOnWire | Reserved for local use.  Opnum: 19 |
| Opnum20NotUsedOnWire | Reserved for local use.  Opnum: 20 |
| Opnum21NotUsedOnWire | Reserved for local use.  Opnum: 21 |

In the preceding table, the term "Reserved for local use" means that the client MUST NOT send the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7), and the server behavior is undefined because it does not affect interoperability.

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same, unless specified otherwise.

#### ITypeInfo::GetTypeAttr (Opnum 3)

The GetTypeAttr method retrieves a [TYPEATTR](#Section_0ca10d0861d2405991097bbaf545715e) structure that contains information about the type described by the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa).

1. HRESULT GetTypeAttr(
2. [out] LPTYPEATTR\* ppTypeAttr,
3. [out] DWORD\* pReserved
4. );

**ppTypeAttr:** MUST be set to a TYPEATTR structure whose data values describe the type associated with the ITypeInfo server, as specified in section 2.2.44.

**pReserved:** MUST be set to 0 and ignored on receipt.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo::GetTypeComp (Opnum 4)

The GetTypeComp method retrieves a reference to the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) instance associated with the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa).

1. HRESULT GetTypeComp(
2. [out] ITypeComp\*\* ppTComp
3. );

**ppTComp:** MUST be set to a reference to the ITypeComp server instance associated with the ITypeInfo server (see section 3.5).

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the sevierty bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo::GetFuncDesc (Opnum 5)

The GetFuncDesc method retrieves a [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e) structure that contains information about a member of the [ITypeInfo server's](#Section_99504cf916d8401ea87383b85d1ee4aa) method or dispatch method table.

1. HRESULT GetFuncDesc(
2. [in] UINT index,
3. [out] LPFUNCDESC\* ppFuncDesc,
4. [out] DWORD\* pReserved
5. );

**index:** MUST equal the ordinal position in the method table (if the type describes a [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408) or module) or the dispatch method table (if the type describes a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5)) of the method whose description is to be returned. The value of index MUST be less than the value of the cFuncs field in the [TYPEATTR (section 2.2.44)](#Section_0ca10d0861d2405991097bbaf545715e) structure returned by the [GetTypeAttr (section 3.7.4.1)](#Section_b96292a8c06e4b9c905e129b95697ee4) method.

**ppFuncDesc:** MUST be set to a FUNCDESC structure (see section 2.2.42) that contains the data values associated with the specified member of the method or dispatch method table, or NULL if no such member exists.

**pReserved:** MUST be set to 0 and ignored on receipt.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the method table. |

#### ITypeInfo::GetVarDesc (Opnum 6)

The GetVarDesc method retrieves a [VARDESC](#Section_ae7791d243994dffb7c6b0d4f3dce982) structure that contains information about a member of the [ITypeInfo server's](#Section_99504cf916d8401ea87383b85d1ee4aa) data member table.

1. HRESULT GetVarDesc(
2. [in] UINT index,
3. [out] LPVARDESC\* ppVarDesc,
4. [out] DWORD\* pReserved
5. );

**index:** MUST equal the data member table index value of the data member whose description is to be returned. The value of index MUST be less than the value of the cVars field in the [TYPEATTR](#Section_0ca10d0861d2405991097bbaf545715e) structure returned by the GetTypeAttr method, as specified in [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4) and 2.2.44.

**ppVarDesc:** MUST be set to a VARDESC structure (see section 2.2.43) that contains the data values associated with the specified member of the data member table, or NULL if no such member exists.

**pReserved:** MUST be set to 0 and ignored on receipt.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* and *invKind* did not specify the ordinal position of an element in the method table. See [MS-ERREF]. |

#### ITypeInfo::GetNames (Opnum 7)

The GetNames method retrieves the data member name or the method and parameter names associated with a specified [MEMBERID](#Section_ace8758fee2b4cb68645973994d12530).

1. HRESULT GetNames(
2. [in] MEMBERID memid,
3. [out, size\_is(cMaxNames), length\_is(\*pcNames)]
4. BSTR\* rgBstrNames,
5. [in] UINT cMaxNames,
6. [out] UINT\* pcNames
7. );

**memid:** MUST be a MEMBERID (section 2.2.35).

**rgBstrNames:** MUST be set to an array of [BSTR](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1). If *pcNames* is 0, *rgBstrNames* MUST be NULL.

**cMaxNames:** MUST specify the maximum length of the *rgBstrNames* array that the client can accept.

**pcNames:** MUST be set to the length of the *rgBstrNames* array.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of memid did not specify the MEMBERID of a member of the type. See [MS-ERREF]. |

If the *memid* field corresponds to multiple property accessor methods, the contents of *rgBstrNames* MUST correspond to the [propget] property accessor.

If the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) represents an appobject [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) (see section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c)) and *memid* is MEMBERID\_DEFAULTINST, the first element of *rgBstrNames* MUST be set to the name of the coclass.

In all other cases, the first element of *rgBstrNames* MUST be set to the name of the method or data element in the binding context of the ITypeInfo server that corresponds to the value of *memid*.

If *memid* specifies a method or property accessor method, the remaining elements of *rgBstrNames* MUST be set to the names of entries in its parameter table, in the order in which they are stored in the parameter table.

If *memid* specifies a put or putref property, the *rgBstrNames* array MUST NOT include the name of the [retval] parameter. If *memid* specifies a member of a [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5), the *rgBstrNames* array MUST NOT include the names of [lcid] or [retval] parameters (see section [3.1.4.4](#Section_5c2a199760d7496d8d9aed940bbb82eb)). In all other cases, the *rgBstrNames* array MUST include all members of the parameter table.

#### ITypeInfo::GetRefTypeOfImplType (Opnum 8)

The GetRefTypeOfImplType method retrieves the [HREFTYPE](#Section_ed6620b16b234fa199e6781832999f93) corresponding to the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) of an interface that is inherited, implemented, or referenced by the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa).

1. HRESULT GetRefTypeOfImplType(
2. [in] UINT index,
3. [out] HREFTYPE\* pRefType
4. );

**index:** MUST be a nonnegative integer, or -1.

If the ITypeInfo server describes a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) (see sections [2.2.49.4.2](#Section_bd30db0d6c384d549c4467c0e9d25551) and [3.7.1](#Section_1e542e10fe4d475f96156b6d956b7073)), the value of index MUST be 0 or -1.

If the ITypeInfo server describes a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc), the value of index MUST be nonnegative and less than the value of the *cImplTypes* field in the [TYPEATTR (section 2.2.44)](#Section_0ca10d0861d2405991097bbaf545715e) structure returned by the [GetTypeAttr (section 3.7.4.1)](#Section_b96292a8c06e4b9c905e129b95697ee4) method.

For all other ITypeInfo servers, the value of index MUST be 0.

**pRefType:** MUST be set to one of the following values, if index is -1 or specifies an interface table entry.

If the ITypeInfo server describes a dual interface and index is -1, *pRefType* is specified by the [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33) value associated with the ITypeInfo server (see section 2.2.44):

TKIND\_DISPATCH: MUST be set to the HREFTYPE of the [**partner interface**](#gt_3610e61d-bd5c-454f-992a-0a020995e66b).

TKIND\_INTERFACE: MUST be set to the HREFTYPE of the [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb).

In all other cases, *pRefType* is specified by the interface table member whose ordinal position is specified by index:

If the interface table member is a dual interface and the ITypeInfo server describes a [**DCOM interface**](#gt_4b20db64-5f0c-4df0-9ecf-91cdde2c2408) or partner interface, *pRefType* MUST be the HREFTYPE of the partner interface of the interface table member.

**Note**  This is the only case where an OLE Automation Protocol interface method returns a partner interface by default.

If the interface table member is a dual interface and the ITypeInfo server describes a coclass, *pRefType* MUST be the HREFTYPE of the partner dispinterface of the interface table member.

MUST be set to the HREFTYPE of the interface table member in all other cases.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of index did not specify the ordinal position of an element in the interface table, or the value of index was -1 and the type was not a dual interface. See [MS-ERREF]. |

#### ITypeInfo::GetImplTypeFlags (Opnum 9)

The GetImplTypeFlags method retrieves the [IMPLTYPEFLAGS values](#Section_2743c240436242bf9482a910ac1857f7) associated with an interface member of a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc).

1. HRESULT GetImplTypeFlags(
2. [in] UINT index,
3. [out] INT\* pImplTypeFlags
4. );

**index:** MUST be the ordinal position in the coclass interface table of the interface whose associated IMPLTYPEFLAGS values are to be returned.

**pImplTypeFlags:** MUST be set to either a combination of the IMPLTYPEFLAGS feature constants specified in section 2.2.13, or 0.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of index did not specify the ordinal position of an element in the interface table. See [MS-ERREF]. |

#### ITypeInfo::GetDocumentation (Opnum 12)

The GetDocumentation method retrieves the documentation resources associated with a type member.

1. HRESULT GetDocumentation(
2. [in] MEMBERID memid,
3. [in] DWORD refPtrFlags,
4. [out] BSTR\* pBstrName,
5. [out] BSTR\* pBstrDocString,
6. [out] DWORD\* pdwHelpContext,
7. [out] BSTR\* pBstrHelpFile
8. );

**memid:** MUST be either the [MEMBERID](#Section_ace8758fee2b4cb68645973994d12530) of a method or data member in the binding context of the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) (see section [3.5.4.1.1](#Section_cc88e6254b3a4cf9896626bdc913f62d)), or MEMBERID\_NIL (see section [2.2.35.1](#Section_5fbb485125f645ef9f83e9dd633e1e00)).

**refPtrFlags:** MUST be a combination of the bit flags specified in the following table, or 0.

| Value | Meaning |
| --- | --- |
| TYPEINFO\_NameArg  0x00000001 | MUST specify that the client is interested in the actual *pBstrName* [out] argument. |
| TYPEINFO\_DocStringArg  0x00000002 | MUST specify that the client is interested in the actual *pBstrDocString* [out] argument. |
| TYPEINFO\_HelpContextArg  0x00000004 | MUST specify that the client is interested in the actual *pdwHelpContext* [out] argument. |
| TYPEINFO\_HelpFileArg  0x00000008 | MUST specify that the client is interested in the actual *pBstrHelpFile* [out] argument. |

**pBstrName:** If the TYPEINFO\_NameArg bit flag is set in *refPtrFlags*,*pBstrName* MUST be set to a [BSTR](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1) that contains the name of the type or specified type member. Othernwise, *pBstrName* MUST be set to a NULL BSTR.

**pBstrDocString:** MUST be set to the documentation string that was associated with the type or specified type member using the [helpstring] attribute (see section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)), if the TYPEINFO\_DocStringArg bit flag is set in *refPtrFlags*. MAY be set to an implementation-specific string[<59>](#Appendix_A_59" \o "Product behavior note 59) if no [helpstring] attribute is specified. MUST be set to a NULL BSTR otherwise.

**pdwHelpContext:** MUST be set to the value that was associated with the type or specified type member using the [helpcontext] attribute (see section 2.2.49.2), if the TYPEINFO\_HelpContextArg bit flag is set in *refPtrFlags*. MUST be set to 0 otherwise.

**pBstrHelpFile:** MUST be set to the documentation string that was associated with the type or specified type member using the [helpfile] attribute (see section 2.2.49.2), if the TYPEINFO\_HelpFileArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

If *memid* is MEMBERID\_NIL, the values of *pBstrName*, *pBstrDocString*, *pdwHelpContext*, and *pBstrHelpFile* MUST correspond to the attributes declared with the type, as specified in section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e). Otherwise, they MUST correspond to the attributes declared with the specified member of the type.

#### ITypeInfo::GetDllEntry (Opnum 13)

The GetDllEntry method retrieves values associated with a local-only method defined in a module.

1. HRESULT GetDllEntry(
2. [in] MEMBERID memid,
3. [in] INVOKEKIND invKind,
4. [in] DWORD refPtrFlags,
5. [out] BSTR\* pBstrDllName,
6. [out] BSTR\* pBstrName,
7. [out] WORD\* pwOrdinal
8. );

**memid:** MUST be the [MEMBERID](#Section_ace8758fee2b4cb68645973994d12530) of a method member of the module defined in the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

**invKind:** MUST be a value of the [INVOKEKIND (section 2.2.14)](#Section_a0d3598da3ee440187fd17a7031b0b9a) enumeration that specifies a single property accessor method, if *memid* corresponds to a property with multiple accessors.

**refPtrFlags:** MUST be a combination of the bit flags specified in the following table, or 0.

| Value | Meaning |
| --- | --- |
| TYPEINFO\_DLLNameArg  0x00000001 | MUST specify that the client is interested in the actual *pBstrDllName* [out] argument. |
| TYPEINFO\_NameArg  0x00000002 | MUST specify that the client is interested in the actual *pBstrName* [out] argument. |
| TYPEINFO\_OrdinalArg  0x00000004 | MUST specify that the client is interested in the actual *pwOrdinal* [out] argument. |

**pBstrDllName:** MUST be set to the value associated with the method using the [dllname] attribute (see section [2.2.49.9](#Section_82f9465bae46474e87ffe65e9751affb)) if the TYPEINFO\_DllNameArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**pBstrName:** MUST be set to the value associated with the method using the [entry] attribute (see section 2.2.49.9), if the associated value is a string and the TYPEINFO\_NameArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**pwOrdinal:** MUST be set to the value associated with the method using the [entry] attribute (see section 2.2.49.9), if the associated value is an integer and the TYPEINFO\_OrdinalArg bit flag is set in *refPtrFlags*. MUST be set to 0 otherwise.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x800288BD  TYPE\_E\_BADMODULEKIND | The type is not a module. See [MS-ERREF]. |
| 0x8002802C  TYPE\_E\_AMBIGUOUSNAME | The values of *memid* and *invKind* match more than one element in the binding context. See [MS-ERREF]. |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *memid* and *invKind* did not specify the ordinal position of an element in the interface table, or the type is not a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc). See [MS-ERREF]. |

#### ITypeInfo::GetRefTypeInfo (Opnum 14)

The GetRefTypeInfo method retrieves an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) that describes a type that is inherited, implemented, or referenced by the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa).

1. HRESULT GetRefTypeInfo(
2. [in] HREFTYPE hRefType,
3. [out] ITypeInfo\*\* ppTInfo
4. );

**hRefType:** MUST be an [HREFTYPE (section 2.2.36)](#Section_ed6620b16b234fa199e6781832999f93) that has been provided by the ITypeInfo server instance whose GetRefTypeInfo method is being called.

**ppTInfo:** MUST be set to a reference to an ITypeInfo server instance that provides an automation type description of the inherited or referenced interface, or NULL if *hRefType* does not specify an available interface.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo::CreateInstance (Opnum 16)

The CreateInstance method creates a new instance of a type that describes a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) ([**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc)).

1. HRESULT CreateInstance(
2. [in] REFIID riid,
3. [out, iid\_is(riid)] IUnknown\*\* ppvObj
4. );

**riid:** MUST be an [**IID**](#gt_76ad3105-3f05-479d-a40c-c9c8fa2ebd83).

**ppvObj:**  MUST be set to reference an existing instance of the coclass described by the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa), if the coclass was declared with the [predeclid] or [appobject] attributes and an instance of the coclass exists. MUST be set to NULL if the coclass was declared with the [noncreatable] attribute. Otherwise, MUST be set to a new instance of the coclass described by the ITypeInfo server or NULL if the class cannot be instantiated.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x80000004  TYPE\_E\_NOINTERFACE | The value of riid did not specify a known type. See [MS-ERREF]. |
| 0x800288BD  TYPE\_E\_BADMODULEKIND | The ITypeInfo server specified a non-coclass type. See [MS-ERREF]. |

#### ITypeInfo::GetMops (Opnum 17)

The GetMops method has no effect when called across the wire.

1. HRESULT GetMops(
2. [in] MEMBERID memid,
3. [out] BSTR\* pBstrMops
4. );

**memid:** MUST be a nonzero [MEMBERID](#Section_ace8758fee2b4cb68645973994d12530).

**pBstrMops:** MUST be set to a NULL BSTR.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo::GetContainingTypeLib (Opnum 18)

The GetContainingTypeLib method retrieves the [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b) instance whose type information table contains the ITypeInfo instance, and the index of the ITypeInfo instance within the type information table.

1. HRESULT GetContainingTypeLib(
2. [out] ITypeLib\*\* ppTLib,
3. [out] UINT\* pIndex
4. );

**ppTLib:** MUST be set to a reference to an ITypeLib server instance (see section 3.11).

**pIndex:** MUST be set to the index value of the current [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) within the type information table (see section [3.11.1](#Section_87fd9a39606742a7b8e613637df3bd0d)).

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

### Timer Events

None.

### Other Local Events

None.

## ITypeInfo Client Details

### Abstract Data Model

None.

### Timers

None.

### Initialization

None.

### Message Processing Events and Sequencing Rules

To retrieve a reference to an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa), a client MUST do one of the following:

* If the client holds a reference to an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) IDispatch implementation, it can call [IDispatch::GetTypeInfo](#Section_d1791851649142898c5725967ef7b9ed) (see section 3.1.4.2)
* If the client holds a reference to an [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6), it can call [ITypeComp::Bind](#Section_476f00da080640d9bbf36059154abbb7) (see section 3.5.4.1) or [ITypeComp::BindType](#Section_cf61a786b1814267bb6a0987eeb17b38) (see section 3.5.4.2)
* If the client holds a reference to another ITypeInfo server, it can call [ITypeInfo::GetRefTypeInfo](#Section_8428b8f844574fab99fe38a8c6ab217b) (see section 3.7.4.10)
* If the client holds a reference to an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b), it can call [ITypeLib::GetTypeInfo](#Section_ee27cb4791624e5caa667557421a88de) (see section 3.11.4.2), [ITypeLib::GetTypeInfoOfGuid](#Section_01dd3fef481b4957b540baa469cbc3a7) (see section 3.11.4.4), or [ITypeLib::FindName](#Section_8d41f5777cba48f294a4141372f59a0e) (see section 3.11.4.9)

A client MUST call [ITypeInfo::GetTypeAttr (section 3.7.4.1)](#Section_b96292a8c06e4b9c905e129b95697ee4) to retrieve the essential characteristics of the type. To iterate over the data and method members of the type, the client MUST use the data retrieved in [TYPEATTR](#Section_0ca10d0861d2405991097bbaf545715e) and then call [ITypeInfo::GetVarDesc (section 3.7.4.4)](#Section_a6b5857a38e446cda2e4bfbc7e21c787) and [ITypeInfo::GetFuncDesc (section 3.7.4.3)](#Section_d54aca0905654fa8b5e4cf87723a89ed).

To discover inheritance relationships for interfaces or the set of nonsource and source interfaces supported by a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc), a client MUST enumerate the referenced types of an ITypeInfo server using [ITypeInfo::GetRefTypeOfImplType (section 3.7.4.6)](#Section_7225fbad7ad0458bb1492e854364fbfd) and ITypeInfo::GetRefTypeInfo (section 3.7.4.10).

To retrieve string information related to the ITypeInfo server, the client MUST call [ITypeInfo::GetDocumentation (section 3.7.4.8)](#Section_2ea2f705bc334cecbbc7613d6ae0f0c6) or [ITypeInfo::GetNames (section 3.7.4.5)](#Section_78533af1f18640f78b3dc65ba9c6ee3c).

### Timer Events

None.

### Other Local Events

None.

## ITypeInfo2 Server Details

An ITypeInfo2 server MUST extend the functionality of [ITypeInfo](#Section_99504cf916d8401ea87383b85d1ee4aa). The main extension present in ITypeInfo2 is the support it provides for custom attributes as well as support for reverse mapping between the MEMBERID of one of its members and its index in the corresponding member table.

### Abstract Data Model

An [ITypeInfo2 server (section 3.9)](#Section_2d6024dad2294d78bbb0b9d5bf6459b7) MUST implement the data model of an [ITypeInfo server (section 3.7)](#Section_99504cf916d8401ea87383b85d1ee4aa).

The [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) reference maintained by an ITypeInfo2 server MUST refer to a server that implements [ITypeLib2](#Section_4bb9bc733cf540a185c7aafaff4874cc).

An ITypeInfo2 server MUST maintain a collection of custom data items for the type, as well as a collection of custom data items for each element in its method table, data member table, interface table, and parameter tables.

Each entry in a custom data item collection corresponds to a custom data item that was declared with the named element in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824), as specified by the **custom-attr** production throughout section [2.2.49](#Section_0c5a99a3dc4b4f7ea109c695b6702b85). For each custom data item, the server MUST maintain the mapping between the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) and the value of the custom data item.

The data value of all custom data items MUST be a value that can be stored in a [\_wireVARIANT (section 2.2.29.1)](#Section_4e2e9bff2ac54bab83081806b256833e).

The custom data item mappings maintained by an ITypeInfo2 server MUST be permanent.

### Timers

None.

### Initialization

The server MUST initialize all data objects that are required of an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa), as specified in section [3.7.3](#Section_752034952f384211835a7e76c48d14b8).

The server MUST initialize its collection of custom data items. The collection MUST NOT change over the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) lifetime.

### Message Processing Events and Sequencing Rules

The [ITypeInfo2](#Section_2d6024dad2294d78bbb0b9d5bf6459b7) interface derives from the [ITypeInfo](#Section_6ca989bf8b69467d96be9634a30155cb) interface. ITypeInfo2 servers MUST implement the methods defined in ITypeInfo in the order in which, and with the opnums with which, they are specified in section 3.7.4.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [GetTypeKind](#Section_6a03300ebd2d45e4b15e2a4c121554e5) | The GetTypeKind method returns the TYPEKIND value that is associated with the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860).  Opnum: 22 |
| [GetTypeFlags](#Section_5e6482f38bb044038dd3fce5c1071c30) | The GetTypeFlags method returns the TYPEFLAGS value that is associated with the automation type description.  Opnum: 23 |
| [GetFuncIndexOfMemId](#Section_9b417eae849f460889f060e3ea04d8a6) | The GetFuncIndexOfMemId method retrieves the location of an element in the data member table, given the associated MEMBERID of the element.  Opnum: 24 |
| [GetVarIndexOfMemId](#Section_34c839261130446ebe590eba8c32aae1) | The GetVarIndexOfMemId method retrieves the location of an element in the data member table, given the associated MEMBERID of the element.  Opnum: 25 |
| [GetCustData](#Section_cd7b30efb5b04d7e976774af8854798d) | The GetCustData method retrieves the value of a custom data item that is associated with the type.  Opnum: 26 |
| [GetFuncCustData](#Section_0e83075806f54526bd6f75dd4dd3bc7a) | The GetFuncCustData method retrieves the value of a custom data item that is associated with the specified method.  Opnum: 27 |
| [GetParamCustData](#Section_ed33bcd0542243f59b657723a6e10280) | The GetParamCustData method retrieves the value of a custom data item that is associated with the specified method parameter.  Opnum: 28 |
| [GetVarCustData](#Section_f80e601f5a46432fa8cb8d682b6f0162) | The GetVarCustData method retrieves the value of a custom data item that is associated with the specified data member.  Opnum: 29 |
| [GetImplTypeCustData](#Section_beed4f3625084de2a16f1a7d01652e23) | The GetImplTypeCustData method retrieves the value of a custom data item that is associated with the specified interface.  Opnum: 30 |
| [GetDocumentation2](#Section_541262a3d8704c8ebe311eb6ab1d9259) | The GetDocumentation2 method retrieves the values that are associated with a type member.  Opnum: 31 |
| [GetAllCustData](#Section_e255f542ccb94eaea2b72cfdab0636eb) | The GetAllCustData method retrieves all the custom data items that are associated with the automation type description.  Opnum: 32 |
| [GetAllFuncCustData](#Section_d11e36add3c14c5f9c55c542cce41c2c) | The GetAllFuncCustData method retrieves all the custom data items that are associated with the specified method.  Opnum: 33 |
| [GetAllParamCustData](#Section_9390aaaa3e254f1480b6ac4cf95ba9a8) | The GetAllParamCustData method retrieves all the custom data items that are associated with the specified method parameter.  Opnum: 34 |
| [GetAllVarCustData](#Section_2b38e4d7189b4f1fa2d2dd95d561bfe6) | The GetAllVarCustData method retrieves all the custom data items that are associated with the specified data member.  Opnum: 35 |
| [GetAllImplTypeCustData](#Section_1a93cba3831444389025fc2a61fc23e7) | The GetAllImplTypeCustData method retrieves all the custom data items that are associated with the specified interface.  Opnum: 36 |

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same, unless specified otherwise.

#### ITypeInfo2::GetTypeKind (Opnum 22)

The GetTypeKind method returns the TYPEKIND value associated with the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860).

1. HRESULT GetTypeKind(
2. [out] TYPEKIND\* pTypeKind
3. );

**pTypeKind:** MUST be set to the TYPEKIND value associated with the automation type description, as specified in section [2.2.17](#Section_78ccbd1cd8ff43019afcdf562372fb33).

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo2::GetTypeFlags (Opnum 23)

The GetTypeFlags method returns the TYPEFLAGS value associated with the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860).

1. HRESULT GetTypeFlags(
2. [out] ULONG\* pTypeFlags
3. );

**pTypeFlags:** MUST be set either to a combination of the TYPEFLAGS type feature constants specified in section [2.2.16](#Section_155c66e2ffe14f18b849f827ca989aa7), or to 0.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo2::GetFuncIndexOfMemId (Opnum 24)

The GetFuncIndexOfMemId method retrieves the location of an element in the data member table, given its associated MEMBERID.

1. HRESULT GetFuncIndexOfMemId(
2. [in] MEMBERID memid,
3. [in] INVOKEKIND invKind,
4. [out] UINT\* pFuncIndex
5. );

**memid:** MUST be a MEMBERID, as specified in section [2.2.35](#Section_ace8758fee2b4cb68645973994d12530).

**invKind:** MUST be set to one of the values of the INVOKEKIND enumeration (as specified in section [2.2.14](#Section_a0d3598da3ee440187fd17a7031b0b9a)) or to 0.

**pFuncIndex:** MUST be set to the ordinal position in the method table of the element specified by the values of memid and invKind as described below, or to –1 if no such element exists.

If invKind is not 0, the specified element is the one whose MEMBERID matches the value of memid, and whose associated INVOKEKIND constant (see [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e)) matches the value of invKind.

If invKind is 0, the specified element is the one with the lowest ordinal position in the method table whose MEMBERID matches the value of memid.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The values of *memid* and *invKind* did not specify a member of the type. See [MS-ERREF]. |

#### ITypeInfo2::GetVarIndexOfMemId (Opnum 25)

The GetVarIndexOfMemId method retrieves the location of an element in the data member table by using the associated MEMBERID of the element.

The method is received by the server in an RPC\_REQUEST packet.

1. HRESULT GetVarIndexOfMemId(
2. [in] MEMBERID memid,
3. [out] UINT\* pVarIndex
4. );

**memid:** MUST be a MEMBERID, as specified in section [2.2.35](#Section_ace8758fee2b4cb68645973994d12530). MUST NOT be MEMBERID\_NIL.

**pVarIndex:** MUST be set to the ordinal position in the data member table of the element whose MEMBERID is specified by memid, if such an element exists. If the method returns a failure code, the value MUST be ignored on receipt.

**Return Values:** The method MUST return information in an HRESULT data structure that is defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *memid* did not specify a member of the type. See [MS-ERREF]. |

#### ITypeInfo2::GetCustData (Opnum 26)

The GetCustData method retrieves the value of a custom data item associated with the type.

1. HRESULT GetCustData(
2. [in] REFGUID guid,
3. [out] VARIANT\* pVarVal
4. );

**guid:** MUST be a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item.

**pVarVal:** MUST be set to the value associated with the GUID using the [custom] attribute (as specified in section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e)), or VT\_EMPTY if the type does not have a value associated with the GUID.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo2::GetFuncCustData (Opnum 27)

The GetFuncCustData method retrieves the value of a custom data item associated with the specified method.

1. HRESULT GetFuncCustData(
2. [in] UINT index,
3. [in] REFGUID guid,
4. [out] VARIANT\* pVarVal
5. );

**index:** MUST specify an ordinal position in the method table and MUST be less than the value of the **cFuncs** field in the [TYPEATTR](#Section_0ca10d0861d2405991097bbaf545715e) structure of the associated type, as specified in sections 2.2.44 and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**guid:** MUST be the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item using the [custom] attribute, as specified in section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f).

**pVarVal:** MUST be set to the value of the custom data item, or VT\_EMPTY if *index* and *guid* do not specify a custom data item.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the method table. See [MS-ERREF]. |

#### ITypeInfo2::GetParamCustData (Opnum 28)

The GetParamCustData method retrieves the value of a custom data item associated with the specified method parameter.

1. HRESULT GetParamCustData(
2. [in] UINT indexFunc,
3. [in] UINT indexParam,
4. [in] REFGUID guid,
5. [out] VARIANT\* pVarVal
6. );

**indexFunc:** MUST specify an ordinal position in the method table and MUST be less than the value of the **cFuncs** field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**indexParam:** MUST specify an ordinal position in the parameter table of the method specified by *indexFunc*. The value of *indexParam* MUST be less than the value of the **cParams** field in the [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e) structure of the associated method, as specified in sections 2.2.42 and [3.7.4.3](#Section_d54aca0905654fa8b5e4cf87723a89ed).

**guid:** MUST be the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item using the [custom] attribute, as specified in section [2.2.49.6](#Section_ff2bd74bcb4d48b3ab896bb32cda3833).

**pVarVal:** MUST be set to the value of the custom data item, or to VT\_EMPTY if the parameter does not have an associated custom data item.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *indexFunc* did not specify the ordinal position of an element in the interface table, or the value of *indexParam* did not specify the ordinal position of an element in the method's parameter table. See [MS-ERREF]. |

#### ITypeInfo2::GetVarCustData (Opnum 29)

The GetVarCustData method retrieves the value of a custom data item associated with the specified data member.

1. HRESULT GetVarCustData(
2. [in] UINT index,
3. [in] REFGUID guid,
4. [out] VARIANT\* pVarVal
5. );

**index:** MUST specify an ordinal position in the data member table and MUST be less than the value of the **cVars** field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**guid:** MUST be the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item using the [custom] attribute, as specified in section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862).

**pVarVal:** MUST be set to the value of the custom data item, or to VT\_EMPTY if the type does not have a value associated with the GUID.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the data member table. See [MS-ERREF]. |

#### ITypeInfo2::GetImplTypeCustData (Opnum 30)

The GetImplTypeCustData method retrieves the value of a custom data item associated with the specified interface member of a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc).

1. HRESULT GetImplTypeCustData(
2. [in] UINT index,
3. [in] REFGUID guid,
4. [out] VARIANT\* pVarVal
5. );

**index:** MUST specify an ordinal position in the interface table and MUST be less than the value of the cImplTypes field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**guid:** MUST be the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item using the [custom] attribute, as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c).

**pVarVal:** MUST be set to the value of the custom data item, or to VT\_EMPTY if the type does not have a value associated with the GUID.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the interface table, or the type is not a coclass. See [MS-ERREF]. |

#### ITypeInfo2::GetDocumentation2 (Opnum 31)

The GetDocumentation2 method retrieves values associated with a type member.

1. HRESULT GetDocumentation2(
2. [in] MEMBERID memid,
3. [in] LCID lcid,
4. [in] DWORD refPtrFlags,
5. [out] BSTR\* pbstrHelpString,
6. [out] DWORD\* pdwHelpStringContext,
7. [out] BSTR\* pbstrHelpStringDll
8. );

**memid:** MUST be the MEMBERID of a member of the type (as specified in section [2.2.35](#Section_ace8758fee2b4cb68645973994d12530)), or MEMBERID\_NIL.

If *memid* is MEMBERID\_NIL, the values of *pBstrHelpString*, *pdwHelpStringContext*, and *pBstrHelpStringDll* MUST correspond to the attributes declared with the type as specified in [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e). Otherwise, they MUST correspond to the attributes declared with the specified member of the type.

**lcid:** MUST be the Locale ID associated with the specified type member.

**refPtrFlags:** MUST be 0, or a combination of the bit flags specified in the following table.

| Value | Meaning |
| --- | --- |
| TYPEINFO\_HelpStringArg  0x00000001 | MUST specify that the client is interested in the actual *pBstrHelpString* [out] argument. |
| TYPEINFO\_HelpStringContextArg  0x00000002 | MUST specify that the client is interested in the actual *pdwHelpStringContext* [out] argument. |
| TYPEINFO\_HelpStringDllArg  0x00000004 | MUST specify that the client is interested in the actual *pBstrHelpStringDll* [out] argument. |

**pbstrHelpString:** If the TYPEINFO\_HelpStringContextArg and TYPEINFO\_HelpStringDllArg bit flags are set in *refPtrFlags*, *pbstrHelpString* MUST be set to an implementation-specific BSTR[<60>](#Appendix_A_60" \o "Product behavior note 60) . Otherwise, MUST be set to a NULL BSTR.

**pdwHelpStringContext:** MUST be set to the value that was associated with the specified type or type member using the [helpstringcontext] attribute (see [IDL Automation Scope](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)) if the TYPEINFO\_HelpStringContextArg bit flag is set in *refPtrFlags*. MUST be set to 0 otherwise.

**pbstrHelpStringDll:** MUST be set to the documentation string that was associated with the specified type or type member using the [helpstringdll] attribute (see IDL Automation Scope) if the TYPEINFO\_HelpStringDllArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo2::GetAllCustData (Opnum 32)

The GetAllCustData method retrieves all the custom data items associated with the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860).

1. HRESULT GetAllCustData(
2. [out] CUSTDATA\* pCustData
3. );

**pCustData:** MUST be set to a CUSTDATA structure that contains an array of custom data items, as specified in section [2.2.47](#Section_b74500e231534cc6bebf9e11320f7bed). The structure's **cCustData** field MUST be set to 0 and its **prgCustData** field set to NULL, if there are no custom data items associated with the automation type description.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeInfo2::GetAllFuncCustData (Opnum 33)

The GetAllFuncCustData method retrieves all of the custom data items associated with the specified method.

1. HRESULT GetAllFuncCustData(
2. [in] UINT index,
3. [out] CUSTDATA\* pCustData
4. );

**index:** MUST specify an ordinal position in the method table and MUST be less than the value of the cFuncs field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**pCustData:** MUST be set to a CUSTDATA structure that contains an array of custom data items, as specified in section [2.2.49.5.1](#Section_232d5f124b8843e3a63360fc157b1a5f). The structure's **cCustData** field MUST be set to 0 and its **prgCustData** field set to NULL, if there are no custom data items associated with the method.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the data member table. See [MS-ERREF]. |

#### ITypeInfo2::GetAllParamCustData (Opnum 34)

The GetAllParamCustData method retrieves all of the custom data items associated with the specified parameter.

1. HRESULT GetAllParamCustData(
2. [in] UINT indexFunc,
3. [in] UINT indexParam,
4. [out] CUSTDATA\* pCustData
5. );

**indexFunc:** MUST specify an ordinal position in the method table and MUST be less than the value of the **cFuncs** field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**indexParam:** MUST specify an ordinal position in the parameter table of the method specified by *indexFunc*. The value of *indexParam* MUST be less than the value of the **cParams** field in the FUNCDESC structure of the associated method, as specified in sections [2.2.42](#Section_d3349d25e11d4095ba86de3fda178c4e) and [3.7.4.3](#Section_d54aca0905654fa8b5e4cf87723a89ed).

**pCustData:** MUST be set to a CUSTDATA structure that contains an array of custom data items, as specified in section [2.2.49.6](#Section_ff2bd74bcb4d48b3ab896bb32cda3833). The structure's **cCustData** field MUST be set to 0 and its **prgCustData** field set to NULL, if there are no custom data items associated with the parameter.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *indexFunc* did not specify the ordinal position of an element in the method table, or the value of *indexParam* did not specify the ordinal position of an element in the parameter table. |

#### ITypeInfo2::GetAllVarCustData (Opnum 35)

The GetAllVarCustData method retrieves all of the custom data items associated with the specified data member.

1. HRESULT GetAllVarCustData(
2. [in] UINT index,
3. [out] CUSTDATA\* pCustData
4. );

**index:** MUST specify an ordinal position in the data member table and MUST be less than the value of the **cVars** field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**pCustData:** MUST be set to a CUSTDATA structure that contains an array of custom data items, as specified in section [2.2.49.5](#Section_da55c4194395453582c4bac998dae862). The structure's **cCustData** field MUST be set to 0 and its **prgCustData** field set to NULL, if there are no custom data items associated with the data member.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the data member table. See [MS-ERREF]. |

#### ITypeInfo2::GetAllImplTypeCustData (Opnum 36)

The GetAllImplTypeCustData method retrieves all of the custom data items associated with the specified data member.

1. HRESULT GetAllImplTypeCustData(
2. [in] UINT index,
3. [out] CUSTDATA\* pCustData
4. );

**index:** MUST specify an ordinal position in the interface table and MUST be less than the value of the **cImplTypes** field in the TYPEATTR structure of the associated type, as specified in sections [2.2.44](#Section_0ca10d0861d2405991097bbaf545715e) and [3.7.4.1](#Section_b96292a8c06e4b9c905e129b95697ee4).

**pCustData:** MUST be set to a CUSTDATA structure that contains an array of custom data items, as specified in section [2.2.49.8](#Section_e86fe771d83647b6b846846de41d592c). The structure's **cCustData** field MUST be set to 0 and its **prgCustData** field set to NULL if there are no custom data items associated with the interface.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire HRESULT DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire HRESULT DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of*index* did not specify the ordinal position of an element in the interface table, or the type is not a [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc). See [MS-ERREF]. |

### Timer Events

None.

### Other Local Events

None.

## ITypeInfo2 Client Details

### Abstract Data Model

None.

### Timers

None.

### Initialization

None.

### Message Processing Events and Sequencing Rules

To retrieve a reference to an [ITypeInfo2 server](#Section_2d6024dad2294d78bbb0b9d5bf6459b7), the client MUST first retrieve a reference to an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) (as specified in section [3.7.4](#Section_6ca989bf8b69467d96be9634a30155cb)), and then call IUnknown::QueryInterface and request IID\_ITypeInfo2.

The protocol specifies no additional sequencing rules.

### Timer Events

None.

### Other Local Events

None.

## ITypeLib Server Details

An [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) server is a [**COM server**](#gt_afe9a4a6-56fc-4162-8c9b-ca7c7e44449c) that provides access to a collection of automation type descriptions through an implementation of ITypeLib.

The types described in an automation type library are defined or referenced in a single [IDL Automation scope (section 2.2.49.2)](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4).

### 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 explain how the protocol behaves. This specification does not mandate that implementations adhere to this model, as long as their external behavior is consistent with the behavior described in this specification.

An [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) server provides a concrete representation of an [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a).

Every automation type library server MUST maintain the following data elements:

* A string that MUST specify the name of the type library.
* A value or structure that MUST specify the attributes declared with the type library in the IDL, as specified in section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4).
* A type information table (see below) that contains a list of [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) mapping entries for types that are defined in the automation scope.
* A type reference table (see below) that contains a list of mappings between HREFTYPE values and automation type descriptions.
* A library reference table (see below) that contains a list of mappings between strings, as specified by importlib statements (see section [2.2.49.10](#Section_ca898dc2450e4e2e983e1eb1f0a68a45)) and automation type libraries.
* A reference to an [ITypeComp](#Section_7894019fde1e455eb2aa3b899c2e50f6) server whose binding context (see section [3.5.4.1.1](#Section_cc88e6254b3a4cf9896626bdc913f62d)) is specified by the members of the automation type library.
* A system pointer size value that specifies the size of a pointer within the data structures used by the automation type descriptions in its type information table. This value is determined when the server is initialized, and is not specified as part of the IDL automation scope.

Each entry in the type information table MUST correspond to a type that is defined or referenced in the automation scope. The entry that corresponds to a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) MUST be its [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb). Each entry MUST contain:

* An ordinal value. The ordinal position of automation type descriptions within the type information table MUST be permanent.
* A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). The GUID MUST be the GUID specified in the automation scope defining the type, or IID\_NULL if no GUID is specified. The mapping between a GUID and an automation type description MUST be permanent and MUST be consistent among all servers that represent automation scopes within the same [**automation scope family**](#gt_79854f42-4476-4d1f-9dad-944a7c81e4fb).
* A string. The mapping between a type name and an automation type description MUST be permanent and MUST be consistent among all servers that represent automation scopes with the same LCID in the same automation scope family.

The type reference table MUST contain one entry for each type defined or referenced in the automation scope.

Each element of the type reference table MUST include the following:

* An HREFTYPE.
* A reference to the automation type description [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) corresponding to the HREFTYPE.
* A reference to an entry in the library reference table (see below) if the automation type description corresponds to a type that is defined by another automation type library.

The library reference table MUST contain one entry for each external automation type library that defines types that are referenced by members of the type reference table.

Each element of the library reference table MUST include the following:

* A string specified by an importlib statement (see section 2.2.49.10).
* A reference to the automation type library instance that corresponds to the specified string.

### Timers

None.

### Initialization

The server MUST initialize its name and [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) reference, and specify the system pointer size value. These values MUST NOT change over the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) lifetime.

The server MUST initialize its type information table, type reference table, and library reference table. The tables MUST NOT change over the server's lifetime.

The server MUST initialize the representation of the attributes declared with its [IDL Automation scope](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4). The attribute representation MUST NOT change over the server's lifetime.

### Message Processing Events and Sequencing Rules

ITypeLib derives from the IUnknown interface. ITypeLib servers MUST implement the methods that are defined in IUnknown, in the order in which, and with the opnums with which, they are specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) Appendix A.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [GetTypeInfoCount](#Section_430a2456664943df966d9d18a8bc2efa) | The GetTypeInfoCount method provides the number of [**automation type descriptions**](#gt_fb6a1829-c102-482c-902f-51c197b22860) in the type information table.  Opnum: 3 |
| [GetTypeInfo](#Section_ee27cb4791624e5caa667557421a88de) | The GetTypeInfo method retrieves the automation type description that has the specified ordinal position within the type information table.  Opnum: 4 |
| [GetTypeInfoType](#Section_1d132d304f9a4d51bdbcf1e92920e23d) | The GetTypeInfoType method retrieves the [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33) value that is associated with an automation type description.  Opnum: 5 |
| [GetTypeInfoOfGuid](#Section_01dd3fef481b4957b540baa469cbc3a7) | The GetTypeInfoOfGuid method retrieves the automation type description with the specified [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) from the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) type information table.  Opnum: 6 |
| [GetLibAttr](#Section_d2941c4ae6684722a50ff22610c809d4) | The GetLibAttr method retrieves a structure that contains the attributes that are declared with the type library.  Opnum: 7 |
| [GetTypeComp](#Section_636b2c755fd643b2bfca5894da8623eb) | The GetTypeComp method retrieves a reference to the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) instance that is associated with the [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b).  Opnum: 8 |
| [GetDocumentation](#Section_ceb2d9eb975a47019a793bb9e6ad419b) | The GetDocumentation method retrieves the documentation resources that are associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).  Opnum: 9 |
| [IsName](#Section_70ea09581a204d04b3d3ab4d12446c08) | The IsName method indicates whether the specified string matches the name of a type or type member that is contained in the automation type library or its binding context.  Opnum: 10 |
| [FindName](#Section_8d41f5777cba48f294a4141372f59a0e) | The FindName method retrieves references to types or type members that are contained in the automation type library and whose names match a specified string.  Opnum: 11 |
| Opnum12NotUsedOnWire | This method is reserved for local use.  Opnum: 12 |

In the preceding table, the term "Reserved for local use" means that the client MUST NOT send the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7), and the server behavior is undefined because it does not affect interoperability.

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same, unless specified otherwise.

#### ITypeLib::GetTypeInfoCount (Opnum 3)

The GetTypeInfoCount method provides the number of [**automation type descriptions**](#gt_fb6a1829-c102-482c-902f-51c197b22860) in the type information table.

1. HRESULT GetTypeInfoCount(
2. [out] UINT\* pcTInfo
3. );

**pcTInfo:** MUST be set to the number of automation type descriptions contained in the type information table of the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib::GetTypeInfo (Opnum 4)

The GetTypeInfo method retrieves the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) that has the specified ordinal position within the type information table.

1. HRESULT GetTypeInfo(
2. [in] UINT index,
3. [out] ITypeInfo\*\* ppTInfo
4. );

**index:** MUST equal the ordinal position of the specified automation type description within the type information table.

**ppTInfo:** MUST be set to a reference to the [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) instance (see section 3.7) with the specified position in the type information table, or to NULL if the value of index is greater than or equal to the number of automation type descriptions in the type information table.

MUST refer to the [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb) if index specifies a [dual interface](#Section_bd30db0d6c384d549c4467c0e9d25551). To retrieve the ITypeInfo server that corresponds to the [**partner interface**](#gt_3610e61d-bd5c-454f-992a-0a020995e66b), the client MUST call the [GetRefTypeOfImplType](#Section_7225fbad7ad0458bb1492e854364fbfd) and [GetRefTypeInfo](#Section_8428b8f844574fab99fe38a8c6ab217b) methods of the ITypeInfo instance that correspond to the partner dispinterface, as specified in sections 3.7.4.6 and 3.7.4.10.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the type information table. See [MS-ERREF]. |

#### ITypeLib::GetTypeInfoType (Opnum 5)

The GetTypeInfoType method retrieves the [TYPEKIND](#Section_78ccbd1cd8ff43019afcdf562372fb33) value associated with an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860).

1. HRESULT GetTypeInfoType(
2. [in] UINT index,
3. [out] TYPEKIND\* pTKind
4. );

**index:** MUST equal the ordinal position of the specified automation type description within the type information table.

**pTKind:** MUST be set to the TYPEKIND value associated with the type description, as specified in 2.2.17.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* did not specify the ordinal position of an element in the type information table. See [MS-ERREF]. |

#### ITypeLib::GetTypeInfoOfGuid (Opnum 6)

The GetTypeInfoOfGuid method retrieves the [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) with the specified [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) from the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) type information table.

1. HRESULT GetTypeInfoOfGuid(
2. [in] REFGUID guid,
3. [out] ITypeInfo\*\* ppTInfo
4. );

**guid:** MUST be a GUID.

**ppTInfo:** MUST be set to an [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) instance that represents the automation type description associated with the specified GUID in the type information table (see section 3.7) or to NULL. MUST be NULL if the value of *guid* is IID\_NULL, or is not associated with an automation type description.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *guid* did not correspond to any entry in the Type information table, or the value of *guid* was IID\_NULL. See [MS-ERREF]. |

#### ITypeLib::GetLibAttr (Opnum 7)

The GetLibAttr method retrieves a structure that contains the attributes declared with the Type library.

1. HRESULT GetLibAttr(
2. [out] LPTLIBATTR\* ppTLibAttr,
3. [out] DWORD\* pReserved
4. );

**ppTLibAttr:** MUST be set to a [TLIBATTR (section 2.2.45)](#Section_b568f4be95e5431bbb3b08dc56e9b224) structure that specifies the attributes declared with the Type library.

**pReserved:** MUST be set to 0 and ignored on receipt.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib::GetTypeComp (Opnum 8)

The GetTypeComp method retrieves a reference to the [ITypeComp server](#Section_7894019fde1e455eb2aa3b899c2e50f6) instance associated with the [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b).

1. HRESULT GetTypeComp(
2. [out] ITypeComp\*\* ppTComp
3. );

**ppTComp:** MUST be set to a reference to the ITypeComp server instance associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e), or to NULL if the automation type library does not have an associated ITypeComp server instance.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib::GetDocumentation (Opnum 9)

The GetDocumentation method retrieves the documentation resources associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

1. HRESULT GetDocumentation(
2. [in] INT index,
3. [in] DWORD refPtrFlags,
4. [out] BSTR\* pBstrName,
5. [out] BSTR\* pBstrDocString,
6. [out] DWORD\* pdwHelpContext,
7. [out] BSTR\* pBstrHelpFile
8. );

**index:** MUST equal the ordinal position of an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) in the type information table, or –1. If index is –1, the values of *pBstrName*, *pBstrDocString*, *pdwHelpContext*, and *pBstrHelpFile* MUST correspond to the attributes declared with the Type library, as specified in section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4). Otherwise, they MUST correspond to the attributes declared with the specified type.

**refPtrFlags:** MUST be a combination of 0, or the bit flags specified in the following table.

| Value | Meaning |
| --- | --- |
| TYPELIB\_NameArg  0x00000001 | MUST specify that the client is interested in the actual *pBstrName* [out] argument. |
| TYPELIB\_DocStringArg  0x00000002 | MUST specify that the client is interested in the actual *pBstrDocString* [out] argument. |
| TYPELIB\_HelpContextArg  0x00000004 | MUST specify that the client is interested in the actual *pdwHelpContext* [out] argument. |
| TYPELIB\_HelpFileArg  0x00000008 | MUST specify that the client is interested in the actual *pBstrHelpFile* [out] argument. |

**pBstrName:** MUST be set to a BSTR that contains the name of the specified type or Type library if the TYPELIB\_NameArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**pBstrDocString:** MUST be set to the documentation string that was associated with the specified type or Type library using the [helpstring] attribute (see section 2.2.49.2), if the TYPELIB\_DocStringArg bit flag is set in *refPtrFlags*. MAY be set to an implementation-specific string[<61>](#Appendix_A_61" \o "Product behavior note 61) if no [helpstring] attribute is specified. MUST be set to a NULL BSTR otherwise.

**pdwHelpContext:** MUST be set to the value that was associated with the specified type or Type library using the [helpcontext] attribute (see section 2.2.49.2), if the TYPELIB\_HelpContextArg bit flag is set in *refPtrFlags*. MUST be set to 0 otherwise.

**pBstrHelpFile:** MUST be set to the documentation string that was associated with the specified type or Type library using the [helpfile] attribute (see section 2.2.49.2), if the TYPELIB\_HelpFileArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**Return Values:** The method MUST return information in an HRESULT data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD does not match a value in the following table, a fatal failure occurred.
* If the severity bit is set to 1 and the entire **HRESULT** DWORD matches a value in the following table, a failure occurred.

| Return value/code | Description |
| --- | --- |
| 0x8002802B  TYPE\_E\_ELEMENTNOTFOUND | The value of *index* was not –1 and did not specify the ordinal position of an element in the type information table. See [MS-ERREF]. |

#### ITypeLib::IsName (Opnum 10)

The IsName method indicates whether the specified string matches the name of a type or type member that is contained in the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) or its binding context.

1. HRESULT IsName(
2. [in] LPOLESTR szNameBuf,
3. [in] ULONG lHashVal,
4. [out] BOOL\* pfName,
5. [out] BSTR\* pBstrNameInLibrary
6. );

**szNameBuf:** MUST be set to a string to be tested if it matches the name of a type or type member.

**lHashVal:** MUST be either the hash value that corresponds to the value of *szNameBuf* (as specified in section [2.2.51](#Section_7a8ed4c314a4433cbaa5b6ec88135352)) or 0.

**pfName:** MUST be set to TRUE if the specified string matches the name of a type or member that is contained in the automation type library (see section [3.11.4.9](#Section_8d41f5777cba48f294a4141372f59a0e)) or its binding context (see section [3.5.4.1.1.1](#Section_693e9d4fc27448868e4f07899ba023c7)) according to the string-matching criteria, as specified in section [2.2.50](#Section_ef05bc9b062d467bad880f19e3e545f6). Otherwise, MUST be set to FALSE.

**pBstrNameInLibrary:** MUST be set to a string whose value matches the value of *szNameBuf* according to the string-matching rules (as specified in section 2.2.50), if *pfName* is TRUE. MUST be set to a NULL BSTR if *pfName* is FALSE.

**Return Values:** The method MUST return the information in an **HRESULT** data structure, which is defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib::FindName (Opnum 11)

The FindName method retrieves references to types, or type members, contained in the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e) whose names match a specified string.

1. HRESULT FindName(
2. [in] LPOLESTR szNameBuf,
3. [in] ULONG lHashVal,
4. [out, size\_is(\*pcFound), length\_is(\*pcFound)]
5. ITypeInfo\*\* ppTInfo,
6. [out, size\_is(\*pcFound), length\_is(\*pcFound)]
7. MEMBERID\* rgMemId,
8. [in, out] USHORT\* pcFound,
9. [out] BSTR\* pBstrNameInLibrary
10. );

**szNameBuf:** MUST be a string.

**lHashVal:** MUST be either the hash value corresponding to the value of *szNameBuf* (as specified in section [2.2.51](#Section_7a8ed4c314a4433cbaa5b6ec88135352)), or 0.

**ppTInfo:** MUST be set to an array of references to ITypeInfo server instances (see section [3.7.4](#Section_6ca989bf8b69467d96be9634a30155cb)). Each entry of *ppTInfo* MUST correspond to a type whose name matches the value of *szNameBuf* according to the string matching criteria (as specified in section [2.2.50](#Section_ef05bc9b062d467bad880f19e3e545f6)) or that contains a member whose name matches the value of *szNameBuf*.

The array MUST be empty if there are no types or method or data members of types defined in the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) whose names match the value of *szNameBuf*. Otherwise, the array MUST contain one entry for each named nonparameter element defined in the automation scope whose name matches *szNameBuf*. The array MAY contain entries for matching types or type members that are referenced, but not defined in the automation scope. [<62>](#Appendix_A_62" \o "Product behavior note 62)

If *szNameBuf* matches the name of a [**dual interface**](#gt_3bb740fd-35c1-4082-a912-2bde177753b9) or one of its members, the corresponding entry in *ppTInfo* MUST refer to the [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb) and MUST NOT refer to the [**partner interface**](#gt_3610e61d-bd5c-454f-992a-0a020995e66b).

The length of the array MUST NOT be greater than the value of *pcFound* set by the client.

**rgMemId:** MUST be set to an array of [MEMBERIDs](#Section_ace8758fee2b4cb68645973994d12530) (see section 2.2.35) corresponding to the ITypeInfo instances in the *ppTInfo* array. For each entry in the *ppTInfo* array, the corresponding value in the *rgMemId* array MUST specify the MEMBERID of the type member whose name matches the value of *szNameBuf*, or MEMBERID\_NIL to specify that the name of the type matches the value of *szNameBuf*.

**pcFound:** The client MUST set *pcFound* to the maximum number of matches that can be returned. The server MUST set *pcFound* to the number of elements in the *ppTInfo* and *rgMemId* arrays.

**pBstrNameInLibrary:** MUST be set to a string whose value matches the value of *szNameBuf* according to the string-matching rules (as specified in section 2.2.50), if the *ppTInfo* array is not empty. MUST be set to a NULL BSTR otherwise.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

### Timer Events

None.

### Other Local Events

None.

## ITypeLib Client Details

### Abstract Data Model

None.

### Timers

None.

### Initialization

None.

### Message Processing Events and Sequencing Rules

A client MUST retrieve a reference to an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b) by calling [ITypeInfo::GetContainingTypeLib](#Section_1a843f04e55a47dd85026ba3cd94b161).

To iterate over the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) collection of [ITypeInfo server](#Section_99504cf916d8401ea87383b85d1ee4aa) references, a client MUST first call [ITypeLib::GetTypeInfoCount](#Section_430a2456664943df966d9d18a8bc2efa) to retrieve the count, and then call [ITypeLib::GetTypeInfo,](#Section_ee27cb4791624e5caa667557421a88de) repeatedly for each index value between 0 and one less than the count previously retrieved.

To retrieve an ITypeInfo reference, given the [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) associated with the type, the client MUST call [ITypeLib::GetTypeInfoOfGuid](#Section_01dd3fef481b4957b540baa469cbc3a7).

To retrieve all the ITypeInfo references associated with a string, the client MUST call [ITypeLib::FindName](#Section_8d41f5777cba48f294a4141372f59a0e).

### Timer Events

None.

### Other Local Events

None.

## ITypeLib2 Server Details

An ITypeLib2 server MUST extend the functionality of [ITypeLib](#Section_5daecf67bc6e4e17bcf8797bdba1748b). The main extension present in ITypeLib2 is the support it provides for custom attributes.

### Abstract Data Model

An [ITypeLib2 server](#Section_4bb9bc733cf540a185c7aafaff4874cc) MUST implement the data model of an [ITypeLib](#Section_87fd9a39606742a7b8e613637df3bd0d) server, as specified in [3.11](#Section_5daecf67bc6e4e17bcf8797bdba1748b).

An ITypeLib2 server MUST maintain a collection of the custom data items that were declared with the library keyword in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824), as specified in section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4). For each custom data item, the server MUST maintain the following mapping:

* The mapping between the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) and the value of the custom data item.

The data value of a custom data item MUST be a value that can be stored in a \_wireVARIANT, as specified in [2.2.29.1](#Section_4e2e9bff2ac54bab83081806b256833e).

The custom data item mappings maintained by an ITypeLib2 server MUST be permanent.

### Timers

None.

### Initialization

The server MUST initialize all data objects that are required of an [ITypeLib](#Section_5daecf67bc6e4e17bcf8797bdba1748b) server, as specified in section [3.11.3](#Section_34d15e8d68494e7d818c2478ff25da06).

The server MUST initialize its collection of custom data items. The collection MUST NOT change over the [**server's**](#gt_434b0234-e970-4e8c-bdfa-e16a30d96703) lifetime.

### Message Processing Events and Sequencing Rules

[ITypeLib2](#Section_4bb9bc733cf540a185c7aafaff4874cc) derives from the [ITypeLib](#Section_5daecf67bc6e4e17bcf8797bdba1748b) interface. ITypeLib2 servers MUST implement the opnums with which they are specified in section [3.11.4](#Section_a1436b20e676495ab4f39e9251a40e7b).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [GetCustData](#Section_7428511033e740c7840a1caab8381609) | The GetCustData method retrieves the value of a custom data item that is associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).  Opnum: 13 |
| [GetLibStatistics](#Section_5ebc64439c0f43e0a5ca54387a97c743) | The GetLibStatistics method returns statistics about the unique names in the automation type library.  Opnum: 14 |
| [GetDocumentation2](#Section_137253c0736e4616833c527ef2dc1618) | The GetDocumentation2 method retrieves the values that are associated with the automation type library.  Opnum: 15 |
| [GetAllCustData](#Section_16112a9f4f20452b9c018b27752c6e72) | The GetAllCustData method retrieves the values of all custom data items that are associated with the automation type library.  Opnum: 16 |

All methods MUST NOT throw exceptions. All return values use the NTSTATUS numbering space; in particular, a value of 0x00000000 indicates success, and any other return value indicates an error. All error values are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) and MUST be treated the same, unless specified otherwise.

#### ITypeLib2::GetCustData (Opnum 13)

The GetCustData method retrieves the value of a custom data item associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

1. HRESULT GetCustData(
2. [in] REFGUID guid,
3. [out] VARIANT\* pVarVal
4. );

**guid:** MUST be the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) associated with the custom data item using the [custom] attribute, as specified in section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4).

**pVarVal:** MUST be set to the value of the custom data item, or [VT\_EMPTY](#Section_3fe7db9f58034dc49d145425d3f5461f) if there is no custom data item associated with the specified GUID.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib2::GetLibStatistics (Opnum 14)

The GetLibStatistics method returns statistics about the unique names in the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

1. HRESULT GetLibStatistics(
2. [out] ULONG\* pcUniqueNames,
3. [out] ULONG\* pcchUniqueNames
4. );

**pcUniqueNames:** MUST be set to the number of unique names in the Type library.

**pcchUniqueNames:** MUST be set to the total length, in characters, of the unique names in the library.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib2::GetDocumentation2 (Opnum 15)

The GetDocumentation2 method retrieves values associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

1. HRESULT GetDocumentation2(
2. [in] INT index,
3. [in] LCID lcid,
4. [in] DWORD refPtrFlags,
5. [out] BSTR\* pbstrHelpString,
6. [out] DWORD\* pdwHelpStringContext,
7. [out] BSTR\* pbstrHelpStringDll
8. );

**index:** MUST be equal to the index of an [**automation type description**](#gt_fb6a1829-c102-482c-902f-51c197b22860) or to –1. If index is –1, the values of *pBstrHelpString*, *pdwHelpStringContext*, and *pBstrHelpStringDll* MUST correspond to the attributes declared with the Type library as specified in section [2.2.49.3](#Section_7b5fa59bd8f64a479695630d3c10363e). Otherwise, they MUST correspond to the attributes declared with the specified type.

**lcid:** MUST be the locale ID of the specified type or type library.

**refPtrFlags:** MUST be 0, or a combination of the bit flags specified in the following table.

| Value | Meaning |
| --- | --- |
| TYPELIB\_HelpStringArg  0x00000001 | MUST specify that the client is interested in the actual *pBstrHelpString* [out] argument. |
| TYPELIB\_HelpContextArg  0x00000002 | MUST specify that the client is interested in the actual *pdwHelpStringContext* [out] argument. |
| TYPELIB\_HelpFileArg  0x00000004 | MUST specify that the client is interested in the actual *pBstrHelpStringDll* [out] argument. |

**pbstrHelpString:** MUST be set to an implementation-specific [BSTR](#Section_9c5a5ce4ff5b45ceb915ada381b34ac1) type[<63>](#Appendix_A_63" \o "Product behavior note 63) if the TYPELIB\_HelpStringArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**pdwHelpStringContext:** MUST be set to the value that was associated with the specified type or type library using the [helpstringcontext] attribute (see section [2.2.49.2](#Section_ae4d27fafaaa4d5fb0f915bfbaaae9b4)) if the TYPELIB\_HelpContextArg bit flag is set in *refPtrFlags*. MUST be set to 0 otherwise.

**pbstrHelpStringDll:** MUST be set to the documentation string that was associated with the specified type or type library using the [helpstringdll] attribute (see section 2.2.49.2) if the TYPELIB\_HelpFileArg bit flag is set in *refPtrFlags*. MUST be set to a NULL BSTR otherwise.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

#### ITypeLib2::GetAllCustData (Opnum 16)

The GetAllCustData method retrieves the values of all custom data items associated with the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

1. HRESULT GetAllCustData(
2. [out] CUSTDATA\* pCustData
3. );

**pCustData:** MUST be set to a [CUSTDATA](#Section_b74500e231534cc6bebf9e11320f7bed) structure that contains an array of custom data items, as specified in section 2.2.47. The structure's **cCustData** field MUST be set to 0 and its **prgCustData** field set to NULL if there are no custom data items associated with the automation type library.

**Return Values:** The method MUST return information in an **HRESULT** data structure, defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.1. The severity bit in the structure identifies the following conditions:

* If the severity bit is set to 0, the method completed successfully.
* If the severity bit is set to 1, the method failed and encountered a fatal error.

### Timer Events

None.

### Other Local Events

None.

## ITypeLib2 Client Details

### Abstract Data Model

None.

### Timers

None.

### Initialization

None.

### Message Processing Events and Sequencing Rules

To retrieve a reference to an [ITypeLib2 server](#Section_4bb9bc733cf540a185c7aafaff4874cc), the client MUST first retrieve a reference to an [ITypeLib server](#Section_5daecf67bc6e4e17bcf8797bdba1748b) (as specified in section [3.12.4](#Section_42f7fe088f65412989924c67d4ecf825)), and then call IUnknown::QueryInterface requesting IID\_ITypeLib2.

The protocol specifies no additional sequencing rules.

### Timer Events

None.

### Other Local Events

None.

# Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the OLE Automation Protocol.

## AIDL-ODL Property Equivalence

This example shows how to map [**AIDL**](#gt_13017420-f6a8-4150-9549-b0d754dbb128) operations to their conceptual ODL equivalents.

For example, a property specified in AIDL as follows:

1. [id(1), propget] HRESULT prop1 ( [out, retval] BSTR\* bstr );

... is equivalent to:

1. [id(1), readonly] BSTR prop1;

Or, the following two methods specified in AIDL as:

1. [id(2), propget] HRESULT prop2 ( [out, retval] BSTR\* bstr );
2. [id(2), propput] HRESULT prop2 ( [in] BSTR bstr );

... are equivalent to a:

1. [id(2)] BSTR prop2;

## AIDL-ODL Method Equivalence

This example shows how to map an [**AIDL**](#gt_13017420-f6a8-4150-9549-b0d754dbb128) method to its conceptual ODL equivalent.

For example, a method specified in AIDL as follows:

1. [id(3)] HRESULT func1 ( [in] int n, [in] BSTR ticker,
2. [lcid] DWORD lcid,
3. [out, retval] CURRENCY\* cy);

... is equivalent to:

1. [id(3)] CURRENCY func1 ( [in] int n, [in] BSTR ticker );

## Invoke Argument Parameter Mapping

This example shows how automation method arguments map to [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) arguments.

Consider a method that conceptually takes the following arguments.

1. test(vPos0, vPosByRef1, vPos2, vNamed3, vNamedByRef4);

The call to the server must be structured as in the following figure.



Figure 5: Call to server

## Getting the Value of a Property

This example shows a sequence of messages between an [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) and a server to get the value of a property.

1. The client calls the [GetIDsOfNames](#Section_7166d6ffb8514216bfaa34128274a242) method on the server for a property named test. The server returns the [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) (see section [2.2.32](#Section_b0b43e39b0804edda26d7134075c75cd)) of the property in the pointer passed by the client.
2. The client calls the [Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) method on the server by using the DISPID returned from the server. Other parameters can be filled as shown in the following diagram. The property value is returned by the server in pVarResult. If an exception occurs during execution, or if there is an error in the arguments passed to Invoke, it is indicated in pExcepInfo or in pArgErr, accordingly (see section 3.1.4.4).

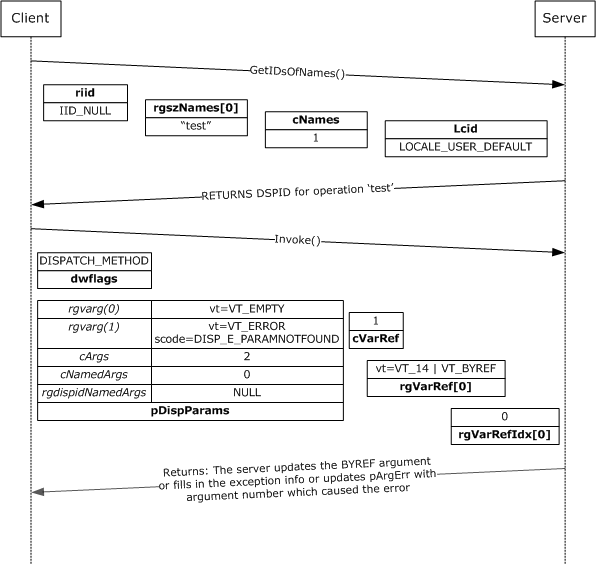


Figure 6: Getting the value of a property

**Note**  For brevity, the **Invoke** method's *dispid*, *riid*, and *lcid* parameters are not shown.

## Setting the Value of a Property

This example shows a sequence of messages between an [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) and a server to set the value of a property.

1. The client calls the [GetIDsOfName](#Section_7166d6ffb8514216bfaa34128274a242) method on the server for a property named test. The server returns the [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474) (see section [2.2.32](#Section_b0b43e39b0804edda26d7134075c75cd)) of the property in the pointer passed by the client.
2. The client calls the [Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) method on the server by using the DISPID returned from the server. The client passes the value of the property in the first VARIANT in rgVarg that is a member of pDispParams (see section [2.2.33](#Section_144b00dd4c2f4b35a28fc17f591b990c)). The remaining members of pDispParams must be filled as shown in the following diagram. On the server side, if an exception occurs during execution, or if there is an error in the arguments passed to Invoke, it is indicated in pExcepInfo or in pArgErr, accordingly (see section 3.1.4.4).

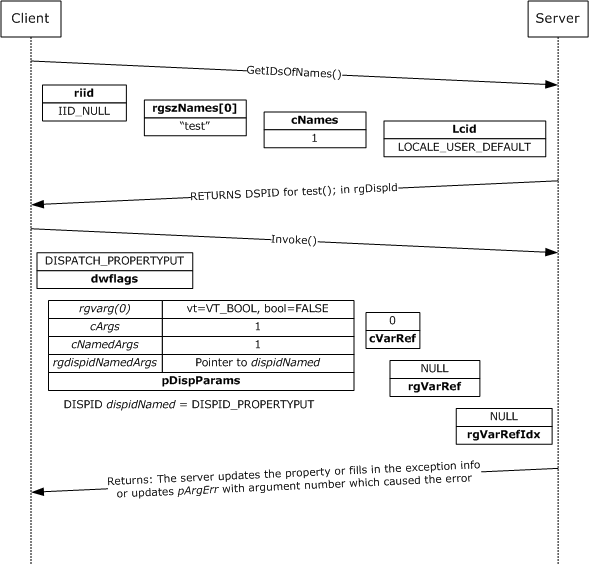


Figure 7: Setting the value of a property

**Note**  For brevity, the **Invoke** method's *dispid*, *riid*, and *lcid* parameters are not shown.

## Calling a Method with Byref and Optional Arguments

This example shows a sequence of messages between an [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) and a server to call a method with a byref and an optional argument. The signature of the function is:

1. HRESULT test ([in, optional] VARIANT A, [in, out, optional] VARIANT \*B);
2. The client calls the **GetIDsOfName** method on the server for the method named test. The server returns the [DISPID (section 2.2.32)](#Section_b0b43e39b0804edda26d7134075c75cd) in the pointer passed by the client.
3. The client calls the [Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) method, filling the parameters as shown in the following figure. In the following example, the client is not passing any value for the first optional argument; thus, the VARIANT in rgVarg[1] has to have the field **vt** set to VT\_ERROR and scode set to DISP\_E\_PARAMNOTFOUND. rgVarg[0] has the **vt** field set to VT\_EMPTY. rgVarRef[0] is a VARIANT with the VT\_BYREF bit flag set.
4. On return from Invoke, rgVarRef[0] with the server-updated value is passed back to the client.

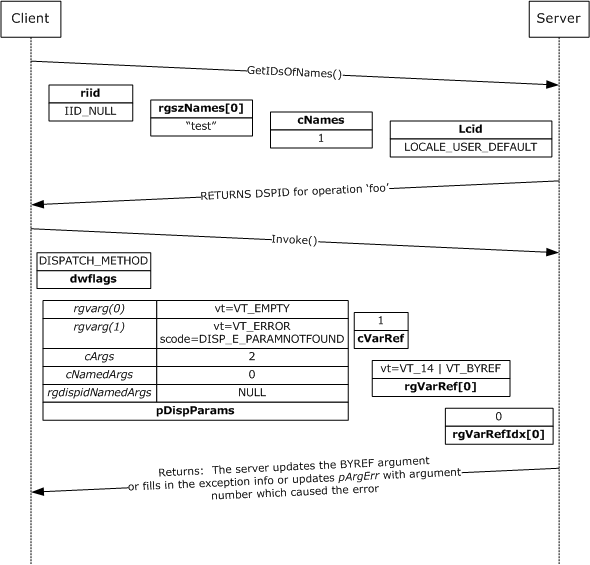


Figure 8: Calling a method with Byref and optional arguments

**Note**  For brevity, the **Invoke** method's *dispid*, *riid*, and *lcid* parameters are not shown.

## IEnumVARIANT Example

An application implementer can choose to implement IEnumVARIANT to expose a collection of homogeneous or heterogeneous data. Depending on the requirements of the application and the nature of the data exposed, the implementer can choose to implement the collection either as a static, semi-static, or dynamic server.

If the application exposes a rarely changing collection (such as the set of students that are enrolled in a specific class), it can do so by exposing a static [IEnumVARIANT server](#Section_716d04d1cd1640659b191b8808b3df31).

If the application exposes data that might change more frequently, but that is meant to be read-only (such as statistics on a set of currently running processes), it can do so by exposing a semi-static IEnumVARIANT server.

If the application exposes data that changes frequently and that is to be as current as possible (such as the set of files from a folder), it can do so by exposing a dynamic IEnumVARIANT server.

In all the examples that follow, the client can use either the \_NewEnum method or the QueryInterface method on the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to get the IEnumVARIANT implementation.

In the examples, an array is maintained as a collection by the server, and the variable current shows the first position in the enumeration that has not yet been returned to the client (see section [3.3.1](#Section_4bc7ca2a61c54fea9d157ec8473350a0)).

### IEnumVARIANT Next() Example

The following diagram illustrates a call to IEnumVARIANT::Next for a server that manages a collection of seven elements. Before the call, the current position is 2. The call to Next(), requesting two elements, causes the current position to be updated to 4, and results in the return by the server to the client of elements with indices 2 and 3. The server also indicates that it filled two elements by setting \*pCeltFetched to 2, and returning 0 as the HRESULT.

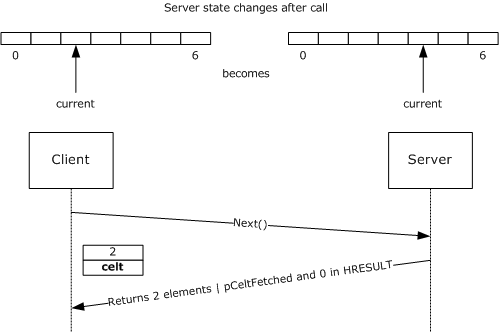


Figure 9: Call to IEnumVARIANT::Next

The following diagram illustrates a call to IEnumVARIANT::Next for a server that manages a collection of seven elements. Before the call, the current position is 3. The call to Next(), requesting seven elements, causes the current position to be updated to 7, and results in the return by the server to the client of elements with indices 3, 4, 5, and 6. The server also indicates that it filled only four elements by setting \*pCeltFetched to 4, and returning 1 as the HRESULT.

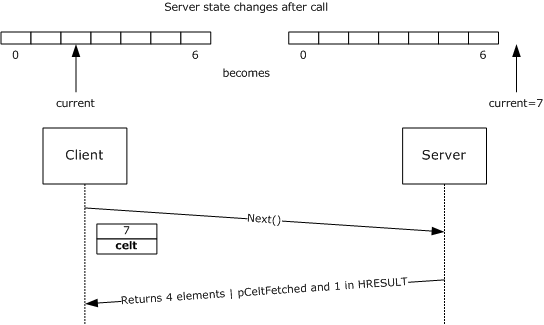


Figure 10: Call to IEnumVARIANT::Next

### IEnumVARIANT Skip() Example

The following diagram illustrates a call to IEnumVARIANT::Skip for a server that manages a collection of seven elements. Before the call, the current position is 2. The call to Skip(), requesting that two elements be skipped, causes the current position to be updated to 4.

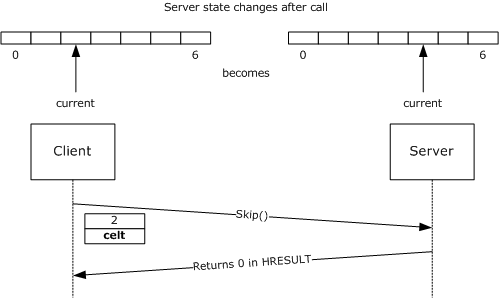


Figure 11: Call to IEnumVARIANT::Skip

### IEnumVARIANT Reset() Example

The following diagram illustrates a call to IEnumVARIANT::Reset for a server that manages a collection of seven elements. Before the call, the current position is 2. The call to Reset() causes the current position to be updated to 0.

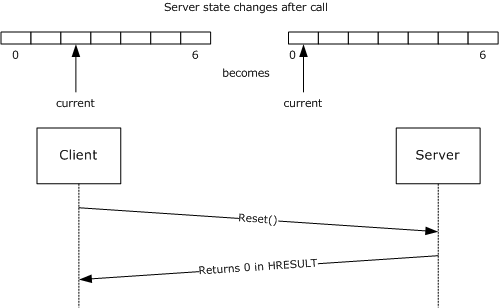


Figure 12: Call to IEnumVARIANT::Reset

### IEnumVARIANT Clone() Example

The following diagram illustrates a call to IEnumVARIANT::Clone for a server that manages a collection of seven elements. Before the call, the current position is 2. The call to Clone() causes a new IEnumVARIANT server to be created. The new server manages a copy of the collection of seven elements, and its current position is set to 2. An **object reference** to the new IEnumVARIANT server is returned to the client.

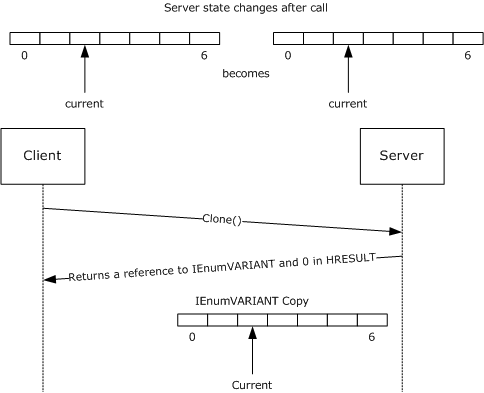


Figure 13: Call to IEnumVARIANT::Clone

## Reading Type Information

The type information exposed by an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) can be read by an [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) using the ITypeLib and ITypeInfo interfaces. The examples that follow show common scenarios.

### Getting ITypeLib Implementations from Automation Server

Assuming that the [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) already has an IDispatch pointer from the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa), the following pseudocode shows how to get the ITypeLib implementation.

1. INPUT: IDispatch pointer from the Automation Server
2. CALL IDispatch::GetTypeInfoCount and OBTAIN pcTInfo
3. COMMENT see Section 3.1.4.1 for information on pcTInfo
4. IF pcTInfo = 0 THEN
5. PRINT Automation Server does not support type information for this object
6. ELSE
7. CALL IDispatch::GetTypeInfo with correct LocaleID and OBTAIN ITypeInfo pointer
8. CALL ITypeInfo::GetContainingTypeLib and OBTAIN ITypeLib pointer
9. ENDIF

### Enumerating on All Types in a Type Library

Building on the [previous example](#Section_fea1146ad7d24c1aa1cd2de196fcbc9a), it is assumed that the [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) has the ITypeLib implementation of the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa). The following pseudocode illustrates how to get type information for all types defined in the type library.

1. INPUT: Reference to the ITypeLib server corresponding to the Automation Server
2. CALL ITypeLib::GetTypeInfoCount and OBTAIN pcTInfo
3. COMMENT see Section 3.11.4.1 for information on pcTInfo
4. FOR X = 0 to pctInfo -1
5. CALL ITypeLib::GetTypeInfo with X and OBTAIN ITypeInfo pointer
6. END FOR

### Enumerating on All Enumerations in a Type Library

This example illustrates how to get all enumerations in a type library, and the value represented by each member within the enumeration.

1. INPUT: Reference to the ITypeLib server corresponding to the Automation Server
2. CALL ITypeLib::GetTypeInfoCount and OBTAIN pcTInfo
3. COMMENT see Section 3.11.4.1 for information on pcTInfo
4. FOR X = 0 to pctInfo -1
5. CALL ITypeLib::GetTypeInfoType with X and OBTAIN pTKind
6. COMMENT see Section 3.11.4.3 for more information on pTKind
7. IF ptKind = TYPEKIND::TKIND\_ENUM THEN
8. CALL ITypeLib::GetTypeInfo with X and OBTAIN ITypeInfo pointer
9. CALL ITypeInfo::GetDocumentation(MEMBERID\_NIL, 1, &BstrName, NULL, NULL, NULL)
10. PRINT Name of the Enumeration is BstrName
11. CALL ITypeInfo::GetTypeAttr and OBTAIN TYPEATTR pointer
12. FOR Y = 0 to TYPEATTR::cVars -1
13. ITypeInfo::GetVarDesc with Y and OBTAIN VARDESC pointer
14. CALL ITypeInfo::GetDocumentation(VARDESC::memid, 1, &BstrName, NULL, NULL,
15. NULL)
16. COMMENT BstrName will contain the name of the enumeration member
17. PRINT BstrName =
18. SET Z to the constant value from VARDESC::lpvarValue
19. COMMENT On most platforms the constant value for enumerations would be in
20. VARDESC::lpvarValue::intVal
21. PRINT Z
22. END FOR
23. END IF
24. END FOR

### Enumerating All Nonsource Interfaces in a Coclass

This example illustrates the identification of all [**coclasses**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) in an [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) type library, and the interfaces implemented by the coclasses.

1. INPUT: Reference to the ITypeLib server corresponding to the Automation Server
2. COMMENT See example in Section 4.8.1 for getting ITypeLib pointer
3. CALL ITypeLib::GetTypeInfoCount and OBTAIN pcTInfo
4. COMMENT see Section 3.11.4.1 for information on pcTInfo
5. FOR X = 0 to pctInfo -1
6. CALL ITypeLib::GetTypeInfoType with X and OBTAIN pTKind
7. COMMENT See Section 3.11.4.3 for more information on pTKind
8. IF pTKind = TYPEKIND::TKIND\_COCLASS THEN
9. CALL ITypeLib::GetDocumentation(X, 1, &BstrName, NULL, NULL, NULL)
10. PRINT Name of the CoClass is BstrName
11. CALL ITypeLib::GetTypeInfo with X and OBTAIN ITypeInfo pointer
12. CALL ITypeInfo::GetTypeAttr and OBTAIN TYPEATTR pointer
13. FOR Y = 0 to TYPEATTR::cImplTypes -1
14. ITypeInfo::GetRefTypeOfImplType with Y and OBTAIN HREFTYPE
15. COMMENT HREFTYPE is a handle to the implemented interface
16. ITypeInfo::GetRefTypeInfo with HREFTYPE and OBTAIN pInterfaceTypeInfo
17. COMMENT pInterfaceTypeInfo is a ITypeInfo pointer for interface implemented
18. by this CoClass
19. CALL pInterfaceTypeInfo::GetDocumentation(MEMBERID\_NIL, 1, &BstrName, NULL,
20. NULL, NULL)
21. PRINT Interface implemented is BstrName
22. END FOR
23. END IF
24. END FOR

### Enumerating All Methods in an Interface

This example pseudocode shows how to enumerate on all the methods declared in an interface. It assumes that the [**automation client**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) has already obtained the IDispatch pointer from the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa).

1. INPUT: IDispatch pointer from the automation server
2. CALL IDispatch::GetTypeInfoCount and OBTAIN pcTInfo
3. COMMENT see Section 3.1.4.1 for information on pcTInfo
4. IF pcTInfo = 0 THEN
5. PRINT Automation Server does not support type information for this object
6. ELSE
7. CALL IDispatch::GetTypeInfo with correct LocaleID and OBTAIN ITypeInfo pointer
8. CALL ITypeInfo::GetDocumentation(MEMBERID\_NIL, 1, &BstrName, NULL,
9. NULL, NULL)
10. PRINT Name of the Interface is BstrName
11. CALL ITypeInfo::GetTypeAttr and OBTAIN TYPEATTR pointer
13. FOR X = 0 to TYPEATTR:: cFuncs -1
14. CALL ITypeInfo::GetFuncDesc with X and OBTAIN FUNCDESC pointer
15. CALL ITypeInfo::GetNames with FUNCDESC::memid and appropriate values for
16. rgBstrNames, cMaxNames and pcNames
17. COMMENT see Section 3.7.4.5 for more information regarding the parameters
18. to ITypeinfo::GetNames
19. IF pcNames > 0 THEN
20. PRINT Name of the method is rgBstrNames[0]
21. PRINT Parameters to above method are following
22. FOR Y = 1 to pcNames -1
23. PRINT rgBstrNames[Y]
24. END FOR
25. END IF
26. END FOR
27. ENDIF

### Retrieving Type Information

This example shows how to retrieve type information for method parameters or members of a struct, a union, or an enumeration given a [TYPEDESC (section 2.2.37)](#Section_95bb92a7f783477facbcc947d754fa8b) structure. A **TYPEDESC** can be obtained from an [ELEMDESC](#Section_e14ff3cf034a4884a498fc7586f7160c) structure, which is a member of [VARDESC](#Section_ae7791d243994dffb7c6b0d4f3dce982) or [FUNCDESC](#Section_d3349d25e11d4095ba86de3fda178c4e).

1. COMMENT This is a recursive procedure and is called PrintTypeDesc.
3. INPUT: TYPEDESC pointer and reference to ITypeInfo server in the binding context
4. OUTPUT: Prints type described by the TYPEDESC
5. CASE TYPEDESC::vt OF
6. VT\_PTR:
7. CALL PrintTypeDesc with TYPEDESC::lptdesc and ITypeInfo pointer
8. PRINT \*
9. VT\_SAFEARRAY:
10. PRINT SAFEARRAY OF
11. CALL PrintTypeDesc with TYPEDESC::lptdesc and ITypeInfo pointer
12. VT\_CARRAY:
13. CALL PrintTypeDesc with TYPEDESC::lpadesc::tdescElem and ITypeInfo pointer
14. COMMENT see Section 2.2.31 for more information on TYPEDESC::lpadesc
15. FOR X = 0 to TYPEDESC::lpadesc::cDims -1
16. PRINT [
17. PRINT TYPEDESC::lpadesc::rgbounds[X].lLbound
18. PRINT &
19. SET Y to TYPEDESC::lpadesc::rgbounds[X].lLbound +
20. TYPEDESC::lpadesc::rgbounds[X].cElements -1
21. PRINT Y
22. PRINT ]
23. END FOR
24. VT\_USERDEFINED:
25. CALL ITypeInfo::GetRefTypeInfo with TYPEDESC::hreftype and OBTAIN
26. pCustomTypeInfo which of type ITypeInfo pointer
27. CALL pCustomTypeInfo::GetDocumentation(MEMBERID\_NIL, 1, &BstrName, NULL,
28. NULL, NULL)
29. PRINT BstrName
30. VT\_I2: PRINT short
31. VT\_I4: PRINT int
32. VT\_R4: PRINT float
33. VT\_R8: PRINT double
34. VT\_CY: PRINT CY
35. VT\_DATE: PRINT DATE
36. VT\_BSTR: PRINT BSTR
37. VT\_DECIMAL: PRINT DECIMAL
38. VT\_DISPATCH: PRINT IDispatch
39. VT\_ERROR: PRINT SCODE
40. VT\_BOOL: PRINT VARIANT\_BOOL
41. VT\_VARIANT: PRINT VARIANT
42. VT\_UNKNOWN: PRINT IUnknown
43. VT\_UI1: PRINT BYTE
44. VT\_I1: PRINT char
45. VT\_UI2: PRINT unsigned short
46. VT\_UI4: PRINT unsigned long
47. VT\_I8: PRINT \_\_int64
48. VT\_UI8: PRINT unsigned \_\_int64
49. VT\_INT: PRINT int
50. VT\_UINT: PRINT unsigned int
51. VT\_HRESULT: PRINT HRESULT
52. VT\_VOID: PRINT void
53. VT\_LPSTR: PRINT char \*
54. VT\_LPWSTR: PRINT wchar \*
55. OTHERS: PRINT Error
56. ENDCASE

### Binding to a Member of a Default Nonsource Interface of an Appobject Coclass

This example shows how to bind to a member of a default nonsource interface of an appobject class.

The sample first tries to bind against the name by using the ITypeComp that corresponds to the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e). The sample then uses the ITypeInfo reference thus retrieved, corresponding to the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) definition, to bind against the name again, retrieving the ITypeInfo server that contains the name in its binding context and the element description (a FUNCDESC or a VARDESC) that corresponds to the name passed in.

1. INPUT: A reference to an ITypeLib server and the name of the
2. member to bind against
3. OUTPUT: A reference to the ITypeInfo server corresponding to the
4. interface containing the member
5. CALL ITypeLib::GetTypeComp with pTypeLib and OBTAIN ITypeComp
6. pointer pTLComp
7. CALL ITypeComp::Bind with pTLComp and name and OBTAIN ITypeInfo
8. pointer pTIAppObj and DESCKIND value dk1
9. COMMENT If the name was a member of a default nonsource interface
10. on an appobject coclass the Bind operation will return
11. the coclass ITypeInfo, DESCKIND\_IMPLICITAPPOBJ, and a
12. VAR\_STATIC VARDESC
13. IF dk1 = DESCKIND\_IMPLICITAPPOBJ THEN
14. CALL ITypeInfo::GetTypeComp with pTIAppObj
15. and OBTAIN ITypeComp pointer pAppObjComp
16. COMMENT The Bind operation below will return the default
17. interface ITypeInfo, DESCKIND\_VARDESC or DESCKIND\_FUNCDESC,
18. and a corresponding VARDESC or FUNCDESC
19. CALL ITypeComp::Bind with pAppObjComp and name
20. and OBTAIN ITypeInfo pointer pTIDefItf and DESCKIND value dk2
21. END IF

### Binding to a Member of a Partner Interface

This example shows how to bind to a member of a [**partner dispinterface**](#gt_603521de-c0d9-4732-ad40-2ece61f8c7eb) given a reference to an ITypeInfo server corresponding to a partner dispinterface.

The sample first retrieves the ITypeInfo reference corresponding to the [**partner interface**](#gt_3610e61d-bd5c-454f-992a-0a020995e66b) by using the special value –1 as an argument to [ITypeInfo::GetRefTypeOfImpleType](#Section_7225fbad7ad0458bb1492e854364fbfd). Next, the sample uses the ITypeInfo reference to complete a bind operation on the specified member name.

1. INPUT: A reference to an ITypeInfo server, and a name of a member
2. OUTPUT: A reference to the corresponding partner interface ITypeInfo server and the FUNCDESC that describes the named member
3. COMMENT pass -1 to GetRefTypeOfImplType to retrieve the HREFTYPE
4. Corresponding to the partner interface
5. CALL ITypeInfo::GetRefTypeOfImplType with pTIDispPartner and -1
6. and OBTAIN HREFTYPE value hrefItfPartner
7. CALL ITypeInfo::GetRefTypeInfo with pTIDispPartner and hrefItfPartner
8. and OBTAIN ITypeInfo pointer pTIItfPartner
9. CALL ITypeInfo::GetTypeComp with pTIItfPartner
10. and OBTAIN ITypeComp pointer pItfPartnerComp
11. COMMENT The Bind operation below will return the ITypeInfo pointer
12. corresponding to the interface in the inheritance hierarchy
13. that defines "name"
14. CALL ITypeComp::Bind with pItfPartnerComp and name
15. and OBTAIN ITypeInfo pointer pTIBindRes and FUNCDESC struct fd

# Security

The following sections specify security considerations for implementers of the OLE Automation Protocol.

## Security Considerations for Implementer

There are no security considerations for this protocol.

## Index of Security Parameters

There are no protocol-specific security parameters.

# Appendix A: Full IDL

For ease of implementation, the full [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) is provided where "ms-dcom.idl" is the IDL specified in [[MS-DCOM]](%5bMS-DCOM%5d.pdf#Section_4a893f3dbd2948cd9f43d9777a4415b0) Appendix A.

1. import "ms-dtyp.idl";
2. import "ms-dcom.idl";
3. // Begin: Extra definitions
4. // Use the next line in any IDL file that imports ms-oaut.idl
5. // that makes use of SAFEARRAYs.
6. // It is needed to bypass special assumptions MIDL compiler makes
7. // about the SAFEARRAY type in windows environment.
8. // #define SAFEARRAY(type) SAFEARRAY
9. // basic type aliases
10. typedef byte BYTE;
11. typedef LONG SCODE;
12. typedef IID \*REFIID;
13. typedef GUID \*REFGUID;
14. typedef [string] wchar\_t \*LPOLESTR;
15. typedef [string] const wchar\_t \*LPCOLESTR;
16. typedef [public] unsigned \_\_int3264 ULONG\_PTR, \*PULONG\_PTR;
17. typedef void \*PVOID, \*LPVOID;
18. // forward declarations
19. interface IDispatch;
20. interface ITypeLib;
21. interface ITypeInfo;
22. typedef [unique] SAFEARRAY \* PSAFEARRAY, \*LPSAFEARRAY;
23. typedef [unique] struct \_wireVARIANT \* VARIANT;
24. // End: Extra definitions
25. typedef enum tagVARENUM
26. {
27. VT\_EMPTY = 0x0000,
28. VT\_NULL = 0x0001,
29. VT\_I2 = 0x0002,
30. VT\_I4 = 0x0003,
31. VT\_R4 = 0x0004,
32. VT\_R8 = 0x0005,
33. VT\_CY = 0x0006,
34. VT\_DATE = 0x0007,
35. VT\_BSTR = 0x0008,
36. VT\_DISPATCH = 0x0009,
37. VT\_ERROR = 0x000A,
38. VT\_BOOL = 0x000B,
39. VT\_VARIANT = 0x000C,
40. VT\_UNKNOWN = 0x000D,
41. VT\_DECIMAL = 0x000E,
42. VT\_I1 = 0x0010,
43. VT\_UI1 = 0x0011,
44. VT\_UI2 = 0x0012,
45. VT\_UI4 = 0x0013,
46. VT\_I8 = 0x0014,
47. VT\_UI8 = 0x0015,
48. VT\_INT = 0x0016,
49. VT\_UINT = 0x0017,
50. VT\_VOID = 0x0018,
51. VT\_HRESULT = 0x0019,
52. VT\_PTR = 0x001A,
53. VT\_SAFEARRAY = 0x001B,
54. VT\_CARRAY = 0x001C,
55. VT\_USERDEFINED = 0x001D,
56. VT\_LPSTR = 0x001E,
57. VT\_LPWSTR = 0x001F,
58. VT\_RECORD = 0x0024,
59. VT\_INT\_PTR = 0x0025,
60. VT\_UINT\_PTR = 0x0026,
61. VT\_ARRAY = 0x2000,
62. VT\_BYREF = 0x4000
63. } VARENUM;
64. typedef enum tagADVFEATUREFLAGS
65. {
66. FADF\_AUTO = 0x0001,
67. FADF\_STATIC = 0x0002,
68. FADF\_EMBEDDED = 0x0004,
69. FADF\_FIXEDSIZE = 0x0010,
70. FADF\_RECORD = 0x0020,
71. FADF\_HAVEIID = 0x0040,
72. FADF\_HAVEVARTYPE = 0x0080,
73. FADF\_BSTR = 0x0100,
74. FADF\_UNKNOWN = 0x0200,
75. FADF\_DISPATCH = 0x0400,
76. FADF\_VARIANT = 0x0800
77. } ADVFEATUREFLAGS;
78. typedef [v1\_enum] enum tagSF\_TYPE {
79. SF\_ERROR = VT\_ERROR,
80. SF\_I1 = VT\_I1,
81. SF\_I2 = VT\_I2,
82. SF\_I4 = VT\_I4,
83. SF\_I8 = VT\_I8,
84. SF\_BSTR = VT\_BSTR,
85. SF\_UNKNOWN = VT\_UNKNOWN,
86. SF\_DISPATCH = VT\_DISPATCH,
87. SF\_VARIANT = VT\_VARIANT,
88. SF\_RECORD = VT\_RECORD,
89. SF\_HAVEIID = VT\_UNKNOWN|0x8000
90. } SF\_TYPE;
91. typedef [v1\_enum] enum tagCALLCONV {
92. CC\_CDECL = 1,
93. CC\_PASCAL = 2,
94. CC\_STDCALL = 4
95. } CALLCONV;
96. typedef enum tagFUNCFLAGS {
97. FUNCFLAG\_FRESTRICTED = 1,
98. FUNCFLAG\_FSOURCE = 0x2,
99. FUNCFLAG\_FBINDABLE = 0x4,
100. FUNCFLAG\_FREQUESTEDIT = 0x8,
101. FUNCFLAG\_FDISPLAYBIND = 0x10,
102. FUNCFLAG\_FDEFAULTBIND = 0x20,
103. FUNCFLAG\_FHIDDEN = 0x40,
104. FUNCFLAG\_FUSESGETLASTERROR = 0x80,
105. FUNCFLAG\_FDEFAULTCOLLELEM = 0x100,
106. FUNCFLAG\_FUIDEFAULT = 0x200,
107. FUNCFLAG\_FNONBROWSABLE = 0x400,
108. FUNCFLAG\_FREPLACEABLE = 0x800,
109. FUNCFLAG\_FIMMEDIATEBIND = 0x1000
110. } FUNCFLAGS;
111. typedef [v1\_enum] enum tagFUNCKIND {
112. FUNC\_PUREVIRTUAL = 1,
113. FUNC\_STATIC = 3,
114. FUNC\_DISPATCH = 4
115. } FUNCKIND;
116. typedef enum tagIMPLTYPEFLAGS {
117. IMPLTYPEFLAG\_FDEFAULT = 0x1,
118. IMPLTYPEFLAG\_FSOURCE = 0x2,
119. IMPLTYPEFLAG\_FRESTRICTED = 0x4,
120. IMPLTYPEFLAG\_FDEFAULTVTABLE = 0x8
121. } IMPLTYPEFLAGS;
122. typedef [v1\_enum] enum tagINVOKEKIND {
123. INVOKE\_FUNC = 0x1,
124. INVOKE\_PROPERTYGET = 0x2,
125. INVOKE\_PROPERTYPUT = 0x4,
126. INVOKE\_PROPERTYPUTREF = 0x8
127. } INVOKEKIND;
128. typedef enum tagPARAMFLAGS {
129. PARAMFLAG\_NONE = 0,
130. PARAMFLAG\_FIN = 0x1,
131. PARAMFLAG\_FOUT = 0x2,
132. PARAMFLAG\_FLCID = 0x4,
133. PARAMFLAG\_FRETVAL = 0x8,
134. PARAMFLAG\_FOPT = 0x10,
135. PARAMFLAG\_FHASDEFAULT = 0x20,
136. PARAMFLAG\_FHASCUSTDATA = 0x40
137. } PARAMFLAGS;
138. typedef enum tagTYPEFLAGS {
139. TYPEFLAG\_FAPPOBJECT = 0x1,
140. TYPEFLAG\_FCANCREATE = 0x2,
141. TYPEFLAG\_FLICENSED = 0x4,
142. TYPEFLAG\_FPREDECLID = 0x8,
143. TYPEFLAG\_FHIDDEN = 0x10,
144. TYPEFLAG\_FCONTROL = 0x20,
145. TYPEFLAG\_FDUAL = 0x40,
146. TYPEFLAG\_FNONEXTENSIBLE = 0x80,
147. TYPEFLAG\_FOLEAUTOMATION = 0x100,
148. TYPEFLAG\_FRESTRICTED = 0x200,
149. TYPEFLAG\_FAGGREGATABLE = 0x400,
150. TYPEFLAG\_FREPLACEABLE = 0x800,
151. TYPEFLAG\_FDISPATCHABLE = 0x1000,
152. TYPEFLAG\_FPROXY = 0x4000
153. } TYPEFLAGS;
154. typedef [v1\_enum] enum tagTYPEKIND {
155. TKIND\_ENUM = 0x0,
156. TKIND\_RECORD = 0x1,
157. TKIND\_MODULE = 0x2,
158. TKIND\_INTERFACE = 0x3,
159. TKIND\_DISPATCH = 0x4,
160. TKIND\_COCLASS = 0x5,
161. TKIND\_ALIAS = 0x6,
162. TKIND\_UNION = 0x7
163. } TYPEKIND;
164. typedef enum tagVARFLAGS {
165. VARFLAG\_FREADONLY = 0x1,
166. VARFLAG\_FSOURCE = 0x2,
167. VARFLAG\_FBINDABLE = 0x4,
168. VARFLAG\_FREQUESTEDIT = 0x8,
169. VARFLAG\_FDISPLAYBIND = 0x10,
170. VARFLAG\_FDEFAULTBIND = 0x20,
171. VARFLAG\_FHIDDEN = 0x40,
172. VARFLAG\_FRESTRICTED = 0x80,
173. VARFLAG\_FDEFAULTCOLLELEM = 0x100,
174. VARFLAG\_FUIDEFAULT = 0x200,
175. VARFLAG\_FNONBROWSABLE = 0x400,
176. VARFLAG\_FREPLACEABLE = 0x800,
177. VARFLAG\_FIMMEDIATEBIND = 0x1000
178. } VARFLAGS;
179. typedef [v1\_enum] enum tagVARKIND {
180. VAR\_PERINSTANCE = 0,
181. VAR\_STATIC = ( VAR\_PERINSTANCE + 1 ),
182. VAR\_CONST = ( VAR\_STATIC + 1 ),
183. VAR\_DISPATCH = ( VAR\_CONST + 1 )
184. } VARKIND;
185. typedef [v1\_enum] enum tagLIBFLAGS {
186. LIBFLAG\_FRESTRICTED = 0x01,
187. LIBFLAG\_FCONTROL = 0x02,
188. LIBFLAG\_FHIDDEN = 0x04,
189. LIBFLAG\_FHASDISKIMAGE = 0x08
190. } LIBFLAGS;
191. typedef [v1\_enum] enum tagSYSKIND {
192. SYS\_WIN32 = 1,
193. SYS\_WIN64 = 3
194. } SYSKIND;
195. typedef [v1\_enum] enum tagDESCKIND {
196. DESCKIND\_NONE = 0,
197. DESCKIND\_FUNCDESC = 1,
198. DESCKIND\_VARDESC = 2,
199. DESCKIND\_TYPECOMP = 3,
200. DESCKIND\_IMPLICITAPPOBJ = 4
201. } DESCKIND;
202. typedef struct \_FLAGGED\_WORD\_BLOB {
203. unsigned long cBytes;
204. unsigned long clSize;
205. [size\_is(clSize)] unsigned short asData[];
206. } FLAGGED\_WORD\_BLOB;
207. typedef [unique] FLAGGED\_WORD\_BLOB\* BSTR;
208. typedef struct tagCY {
209. \_\_int64 int64;
210. } CURRENCY;
211. typedef double DATE;
212. typedef struct tagDEC {
213. WORD wReserved;
214. BYTE scale;
215. BYTE sign;
216. ULONG Hi32;
217. ULONGLONG Lo64;
218. } DECIMAL;
219. typedef short VARIANT\_BOOL;
220. #define VARIANT\_TRUE ((VARIANT\_BOOL)0xffff)
221. #define VARIANT\_FALSE ((VARIANT\_BOOL)0)
222. typedef struct \_wireBRECORD {
223. ULONG fFlags;
224. ULONG clSize;
225. MInterfacePointer \* pRecInfo;
226. [size\_is(clSize)] byte\* pRecord;
227. } wireBRECORDStr;
228. typedef [unique] struct \_wireBRECORD\* BRECORD;
229. typedef struct \_wireVARIANT {
230. DWORD clSize;
231. DWORD rpcReserved;
232. USHORT vt;
233. USHORT wReserved1;
234. USHORT wReserved2;
235. USHORT wReserved3;
236. [switch\_type(ULONG), switch\_is(vt)]
237. union {
238. [case(VT\_I8)]
239. LONGLONG llVal;
240. [case(VT\_I4)]
241. LONG lVal;
242. [case(VT\_UI1)]
243. BYTE bVal;
244. [case(VT\_I2)]
245. SHORT iVal;
246. [case(VT\_R4)]
247. FLOAT fltVal;
248. [case(VT\_R8)]
249. DOUBLE dblVal;
250. [case(VT\_BOOL)]
251. VARIANT\_BOOL boolVal;
252. [case(VT\_ERROR)]
253. HRESULT scode;
254. [case(VT\_CY)]
255. CURRENCY cyVal;
256. [case(VT\_DATE)]
257. DATE date;
258. [case(VT\_BSTR)]
259. BSTR bstrVal;
260. [case(VT\_UNKNOWN)]
261. IUnknown\* punkVal;
262. [case(VT\_DISPATCH)]
263. IDispatch\* pdispVal;
264. [case(VT\_ARRAY)]
265. PSAFEARRAY parray;
266. [case(VT\_RECORD, VT\_RECORD|VT\_BYREF)]
267. BRECORD brecVal;
268. [case(VT\_UI1|VT\_BYREF)]
269. BYTE\* pbVal;
270. [case(VT\_I2|VT\_BYREF)]
271. SHORT\* piVal;
272. [case(VT\_I4|VT\_BYREF)]
273. LONG\* plVal;
274. [case(VT\_I8|VT\_BYREF)]
275. LONGLONG\* pllVal;
276. [case(VT\_R4|VT\_BYREF)]
277. FLOAT\* pfltVal;
278. [case(VT\_R8|VT\_BYREF)]
279. DOUBLE\* pdblVal;
280. [case(VT\_BOOL|VT\_BYREF)]
281. VARIANT\_BOOL\* pboolVal;
282. [case(VT\_ERROR|VT\_BYREF)]
283. HRESULT\* pscode;
284. [case(VT\_CY|VT\_BYREF)]
285. CURRENCY\* pcyVal;
286. [case(VT\_DATE|VT\_BYREF)]
287. DATE\* pdate;
288. [case(VT\_BSTR|VT\_BYREF)]
289. BSTR\* pbstrVal;
290. [case(VT\_UNKNOWN|VT\_BYREF)]
291. IUnknown\*\* ppunkVal;
292. [case(VT\_DISPATCH|VT\_BYREF)]
293. IDispatch\*\* ppdispVal;
294. [case(VT\_ARRAY|VT\_BYREF)]
295. PSAFEARRAY\* pparray;
296. [case(VT\_VARIANT|VT\_BYREF)]
297. VARIANT\* pvarVal;
298. [case(VT\_I1)]
299. CHAR cVal;
300. [case(VT\_UI2)]
301. USHORT uiVal;
302. [case(VT\_UI4)]
303. ULONG ulVal;
304. [case(VT\_UI8)]
305. ULONGLONG ullVal;
306. [case(VT\_INT)]
307. INT intVal;
308. [case(VT\_UINT)]
309. UINT uintVal;
310. [case(VT\_DECIMAL)]
311. DECIMAL decVal;
312. [case(VT\_I1|VT\_BYREF)]
313. CHAR\* pcVal;
314. [case(VT\_UI2|VT\_BYREF)]
315. USHORT\* puiVal;
316. [case(VT\_UI4|VT\_BYREF)]
317. ULONG\* pulVal;
318. [case(VT\_UI8|VT\_BYREF)]
319. ULONGLONG\* pullVal;
320. [case(VT\_INT|VT\_BYREF)]
321. INT\* pintVal;
322. [case(VT\_UINT|VT\_BYREF)]
323. UINT\* puintVal;
324. [case(VT\_DECIMAL|VT\_BYREF)]
325. DECIMAL\* pdecVal;
326. [case(VT\_EMPTY)]
327. ; /\*nothing\*/
328. [case(VT\_NULL)]
329. ; /\*nothing\*/
330. } \_varUnion;
331. } wireVARIANTStr;
332. typedef struct tagSAFEARRAYBOUND {
333. ULONG cElements;
334. LONG lLbound;
335. } SAFEARRAYBOUND,
336. \*LPSAFEARRAYBOUND;
337. typedef struct \_wireSAFEARR\_BSTR {
338. ULONG Size;
339. [size\_is(Size), ref] BSTR\* aBstr;
340. } SAFEARR\_BSTR;
341. typedef struct \_wireSAFEARR\_UNKNOWN {
342. ULONG Size;
343. [size\_is(Size), ref] IUnknown\*\* apUnknown;
344. } SAFEARR\_UNKNOWN;
345. typedef struct \_wireSAFEARR\_DISPATCH {
346. ULONG Size;
347. [size\_is(Size), ref] IDispatch\*\* apDispatch;
348. } SAFEARR\_DISPATCH;
349. typedef struct \_wireSAFEARR\_VARIANT {
350. ULONG Size;
351. [size\_is(Size), ref] VARIANT\* aVariant;
352. } SAFEARR\_VARIANT;
353. typedef struct \_wireSAFEARR\_BRECORD {
354. ULONG Size;
355. [size\_is(Size), ref] BRECORD\* aRecord;
356. } SAFEARR\_BRECORD;
357. typedef struct \_wireSAFEARR\_HAVEIID {
358. ULONG Size;
359. [size\_is(Size), ref] IUnknown\*\* apUnknown;
360. IID iid;
361. } SAFEARR\_HAVEIID;
362. typedef struct \_BYTE\_SIZEDARR {
363. unsigned long clSize;
364. [size\_is(clSize)] byte\* pData;
365. } BYTE\_SIZEDARR;
366. typedef struct \_SHORT\_SIZEDARR {
367. unsigned long clSize;
368. [size\_is(clSize)] unsigned short\* pData;
369. } WORD\_SIZEDARR;
370. typedef struct \_LONG\_SIZEDARR {
371. unsigned long clSize;
372. [size\_is(clSize)] unsigned long\* pData;
373. } DWORD\_SIZEDARR;
374. typedef struct \_HYPER\_SIZEDARR {
375. unsigned long clSize;
376. [size\_is(clSize)] hyper\* pData;
377. } HYPER\_SIZEDARR;
378. typedef union \_wireSAFEARRAY\_UNION
379. switch(unsigned long sfType) u {
380. case SF\_BSTR: SAFEARR\_BSTR BstrStr;
381. case SF\_UNKNOWN: SAFEARR\_UNKNOWN UnknownStr;
382. case SF\_DISPATCH: SAFEARR\_DISPATCH DispatchStr;
383. case SF\_VARIANT: SAFEARR\_VARIANT VariantStr;
384. case SF\_RECORD: SAFEARR\_BRECORD RecordStr;
385. case SF\_HAVEIID: SAFEARR\_HAVEIID HaveIidStr;
386. case SF\_I1: BYTE\_SIZEDARR ByteStr;
387. case SF\_I2: WORD\_SIZEDARR WordStr;
388. case SF\_I4: DWORD\_SIZEDARR LongStr;
389. case SF\_I8: HYPER\_SIZEDARR HyperStr;
390. } SAFEARRAYUNION;
391. typedef
392. [unique]
393. struct \_wireSAFEARRAY {
394. USHORT cDims;
395. USHORT fFeatures;
396. ULONG cbElements;
397. ULONG cLocks;
398. SAFEARRAYUNION uArrayStructs;
399. [size\_is(cDims)] SAFEARRAYBOUND rgsabound[];
400. } \*SAFEARRAY;
401. typedef struct tagRecordInfo {
402. GUID libraryGuid;
403. DWORD verMajor;
404. GUID recGuid;
405. DWORD verMinor;
406. DWORD Lcid;
407. } RecordInfo;
408. typedef LONG DISPID;
409. typedef struct tagDISPPARAMS {
410. [size\_is(cArgs)] VARIANT\* rgvarg;
411. [size\_is(cNamedArgs)] DISPID\* rgdispidNamedArgs;
412. UINT cArgs;
413. UINT cNamedArgs;
414. } DISPPARAMS;
415. typedef struct tagEXCEPINFO {
416. WORD wCode;
417. WORD wReserved;
418. BSTR bstrSource;
419. BSTR bstrDescription;
420. BSTR bstrHelpFile;
421. DWORD dwHelpContext;
422. ULONG\_PTR pvReserved;
423. ULONG\_PTR pfnDeferredFillIn;
424. HRESULT scode;
425. } EXCEPINFO;
426. typedef DISPID MEMBERID;
427. typedef DWORD HREFTYPE;
428. typedef struct tagTYPEDESC {
429. [switch\_type(USHORT), switch\_is(vt)] union {
430. [case(VT\_PTR, VT\_SAFEARRAY)] struct tagTYPEDESC \* lptdesc;
431. [case(VT\_CARRAY)] struct tagARRAYDESC \* lpadesc;
432. [case(VT\_USERDEFINED)] HREFTYPE hreftype;
433. [default] ;
434. } \_tdUnion;
435. USHORT vt;
436. } TYPEDESC;
437. typedef struct tagARRAYDESC {
438. TYPEDESC tdescElem;
439. USHORT cDims;
440. [size\_is(cDims)] SAFEARRAYBOUND rgbounds[];
441. } ARRAYDESC;
442. typedef struct tagPARAMDESCEX {
443. ULONG cBytes;
444. VARIANT varDefaultValue;
445. } PARAMDESCEX;
446. typedef struct tagPARAMDESC {
447. PARAMDESCEX \*pparamdescex;
448. USHORT wParamFlags;
449. } PARAMDESC;
450. typedef struct tagELEMDESC {
451. TYPEDESC tdesc;
452. PARAMDESC paramdesc;
453. } ELEMDESC;
454. typedef struct tagFUNCDESC {
455. MEMBERID memid;
456. [size\_is(cReserved2)] SCODE \* lReserved1;
457. [size\_is(cParams)] ELEMDESC \* lprgelemdescParam;
458. FUNCKIND funckind;
459. INVOKEKIND invkind;
460. CALLCONV callconv;
461. SHORT cParams;
462. SHORT cParamsOpt;
463. SHORT oVft;
464. SHORT cReserved2;
465. ELEMDESC elemdescFunc;
466. WORD wFuncFlags;
467. } FUNCDESC, \*LPFUNCDESC;
468. typedef struct tagVARDESC {
469. MEMBERID memid;
470. LPOLESTR lpstrReserved;
471. [switch\_type(VARKIND), switch\_is(varkind)] union {
472. [case(VAR\_PERINSTANCE, VAR\_DISPATCH, VAR\_STATIC)] ULONG oInst;
473. [case(VAR\_CONST)] VARIANT \* lpvarValue;
474. } \_vdUnion;
475. ELEMDESC elemdescVar;
476. WORD wVarFlags;
477. VARKIND varkind;
478. } VARDESC, \*LPVARDESC;
479. typedef struct tagTYPEATTR {
480. GUID guid;
481. LCID lcid;
482. DWORD dwReserved1;
483. DWORD dwReserved2;
484. DWORD dwReserved3;
485. LPOLESTR lpstrReserved4;
486. ULONG cbSizeInstance;
487. TYPEKIND typekind;
488. WORD cFuncs;
489. WORD cVars;
490. WORD cImplTypes;
491. WORD cbSizeVft;
492. WORD cbAlignment;
493. WORD wTypeFlags;
494. WORD wMajorVerNum;
495. WORD wMinorVerNum;
496. TYPEDESC tdescAlias;
497. DWORD dwReserved5;
498. WORD wReserved6;
499. } TYPEATTR, \*LPTYPEATTR;
500. typedef struct tagTLIBATTR {
501. GUID guid;
502. LCID lcid;
503. SYSKIND syskind;
504. unsigned short wMajorVerNum;
505. unsigned short wMinorVerNum;
506. unsigned short wLibFlags;
507. } TLIBATTR, \*LPTLIBATTR;
508. typedef struct tagCUSTDATAITEM {
509. GUID guid;
510. VARIANT varValue;
511. } CUSTDATAITEM;
512. typedef struct tagCUSTDATA {
513. DWORD cCustData;
514. [size\_is(cCustData)] CUSTDATAITEM \* prgCustData;
515. } CUSTDATA;
516. [
517. object,
518. uuid(00020400-0000-0000-C000-000000000046),
519. pointer\_default(unique)
520. ]
521. interface IDispatch : IUnknown
522. {
523. typedef [unique] IDispatch \* LPDISPATCH;
524. HRESULT GetTypeInfoCount(
525. [out] UINT \* pctinfo
526. );
527. HRESULT GetTypeInfo(
528. [in] UINT iTInfo,
529. [in] LCID lcid,
530. [out] ITypeInfo \*\* ppTInfo
531. );
532. HRESULT GetIDsOfNames(
533. [in] REFIID riid,
534. [in, size\_is(cNames)] LPOLESTR \* rgszNames,
535. [in, range(0, 16384)] UINT cNames,
536. [in] LCID lcid,
537. [out, size\_is(cNames)] DISPID \* rgDispId
538. );
539. HRESULT Invoke(
540. [in] DISPID dispIdMember,
541. [in] REFIID riid,
542. [in] LCID lcid,
543. [in] DWORD dwFlags,
544. [in] DISPPARAMS \* pDispParams,
545. [out] VARIANT \* pVarResult,
546. [out] EXCEPINFO \* pExcepInfo,
547. [out] UINT \* pArgErr,
548. [in] UINT cVarRef,
549. [in, size\_is(cVarRef)] UINT \* rgVarRefIdx,
550. [in, out, size\_is(cVarRef)] VARIANT \* rgVarRef
551. );
552. const DWORD DISPATCH\_METHOD = 0x00000001;
553. const DWORD DISPATCH\_PROPERTYGET = 0x00000002;
554. const DWORD DISPATCH\_PROPERTYPUT = 0x00000004;
555. const DWORD DISPATCH\_PROPERTYPUTREF = 0x00000008;
556. const DWORD DISPATCH\_zeroVarResult = 0x00020000;
557. const DWORD DISPATCH\_zeroExcepInfo = 0x00040000;
558. const DWORD DISPATCH\_zeroArgErr = 0x00080000;
559. const DISPID DISPID\_VALUE = 0;
560. const DISPID DISPID\_UNKNOWN = -1;
561. const DISPID DISPID\_PROPERTYPUT = -3;
562. const DISPID DISPID\_NEWENUM = -4;
563. }
564. [
565. object,
566. uuid(00020404-0000-0000-C000-000000000046),
567. pointer\_default(unique)
568. ]
569. interface IEnumVARIANT : IUnknown
570. {
571. HRESULT Next(
572. [in] ULONG celt,
573. [out, size\_is(celt), length\_is(\*pCeltFetched)]
574. VARIANT \* rgVar,
575. [out] ULONG \* pCeltFetched
576. );
577. HRESULT Skip(
578. [in] ULONG celt
579. );
580. HRESULT Reset();
581. HRESULT Clone(
582. [out] IEnumVARIANT \*\* ppEnum
583. );
584. }
585. [
586. object,
587. uuid(00020403-0000-0000-C000-000000000046),
588. pointer\_default(unique)
589. ]
590. interface ITypeComp : IUnknown
591. {
592. HRESULT Bind(
593. [in] LPOLESTR szName,
594. [in] ULONG lHashVal,
595. [in] WORD wFlags,
596. [out] ITypeInfo \*\* ppTInfo,
597. [out] DESCKIND \* pDescKind,
598. [out] LPFUNCDESC \* ppFuncDesc,
599. [out] LPVARDESC \* ppVarDesc,
600. [out] ITypeComp \*\* ppTypeComp,
601. [out] DWORD \* pReserved
602. );
603. HRESULT BindType(
604. [in] LPOLESTR szName,
605. [in] ULONG lHashVal,
606. [out] ITypeInfo \*\* ppTInfo
607. );
608. }
609. [
610. object,
611. uuid(00020401-0000-0000-C000-000000000046),
612. pointer\_default(unique)
613. ]
614. interface ITypeInfo : IUnknown
615. {
616. HRESULT GetTypeAttr(
617. [out] LPTYPEATTR \* ppTypeAttr,
618. [out] DWORD \* pReserved
619. );
620. HRESULT GetTypeComp(
621. [out] ITypeComp \*\* ppTComp
622. );
623. HRESULT GetFuncDesc(
624. [in] UINT index,
625. [out] LPFUNCDESC \* ppFuncDesc,
626. [out] DWORD \* pReserved
627. );
628. HRESULT GetVarDesc(
629. [in] UINT index,
630. [out] LPVARDESC \* ppVarDesc,
631. [out] DWORD \* pReserved
632. );
633. HRESULT GetNames(
634. [in] MEMBERID memid,
635. [out,size\_is(cMaxNames),length\_is(\*pcNames)]
636. BSTR \* rgBstrNames,
637. [in] UINT cMaxNames,
638. [out] UINT \* pcNames
639. );
640. HRESULT GetRefTypeOfImplType(
641. [in] UINT index,
642. [out] HREFTYPE \* pRefType
643. );
644. HRESULT GetImplTypeFlags(
645. [in] UINT index,
646. [out] INT \* pImplTypeFlags
647. );
648. HRESULT Opnum10NotUsedOnWire(
649. void
650. );
651. HRESULT Opnum11NotUsedOnWire(
652. void
653. );
654. HRESULT GetDocumentation(
655. [in] MEMBERID memid,
656. [in] DWORD refPtrFlags,
657. [out] BSTR \* pBstrName,
658. [out] BSTR \* pBstrDocString,
659. [out] DWORD \* pdwHelpContext,
660. [out] BSTR \* pBstrHelpFile
661. );
662. HRESULT GetDllEntry(
663. [in] MEMBERID memid,
664. [in] INVOKEKIND invKind,
665. [in] DWORD refPtrFlags,
666. [out] BSTR \* pBstrDllName,
667. [out] BSTR \* pBstrName,
668. [out] WORD \* pwOrdinal
669. );
670. HRESULT GetRefTypeInfo(
671. [in] HREFTYPE hRefType,
672. [out] ITypeInfo \*\* ppTInfo
673. );
674. HRESULT Opnum15NotUsedOnWire(
675. void
676. );
677. HRESULT CreateInstance(
678. [in] REFIID riid,
679. [out, iid\_is(riid)] IUnknown \*\* ppvObj
680. );
681. HRESULT GetMops(
682. [in] MEMBERID memid,
683. [out] BSTR \* pBstrMops
684. );
685. HRESULT GetContainingTypeLib(
686. [out] ITypeLib \*\* ppTLib,
687. [out] UINT \* pIndex
688. );
689. HRESULT Opnum19NotUsedOnWire(
690. void
691. );
692. HRESULT Opnum20NotUsedOnWire(
693. void
694. );
695. HRESULT Opnum21NotUsedOnWire(
696. void
697. );
698. }
699. [
700. object,
701. uuid(00020412-0000-0000-C000-000000000046),
702. pointer\_default(unique)
703. ]
704. interface ITypeInfo2 : ITypeInfo
705. {
706. HRESULT GetTypeKind(
707. [out] TYPEKIND \* pTypeKind
708. );
709. HRESULT GetTypeFlags(
710. [out] ULONG \* pTypeFlags
711. );
712. HRESULT GetFuncIndexOfMemId(
713. [in] MEMBERID memid,
714. [in] INVOKEKIND invKind,
715. [out] UINT \* pFuncIndex
716. );
717. HRESULT GetVarIndexOfMemId(
718. [in] MEMBERID memid,
719. [out] UINT \* pVarIndex
720. );
721. HRESULT GetCustData(
722. [in] REFGUID guid,
723. [out] VARIANT \* pVarVal
724. );
725. HRESULT GetFuncCustData(
726. [in] UINT index,
727. [in] REFGUID guid,
728. [out] VARIANT \* pVarVal
729. );
730. HRESULT GetParamCustData(
731. [in] UINT indexFunc,
732. [in] UINT indexParam,
733. [in] REFGUID guid,
734. [out] VARIANT \* pVarVal
735. );
736. HRESULT GetVarCustData(
737. [in] UINT index,
738. [in] REFGUID guid,
739. [out] VARIANT \* pVarVal
740. );
741. HRESULT GetImplTypeCustData(
742. [in] UINT index,
743. [in] REFGUID guid,
744. [out] VARIANT \* pVarVal
745. );
746. HRESULT GetDocumentation2(
747. [in] MEMBERID memid,
748. [in] LCID lcid,
749. [in] DWORD refPtrFlags,
750. [out] BSTR \*pbstrHelpString,
751. [out] DWORD \*pdwHelpStringContext,
752. [out] BSTR \*pbstrHelpStringDll
753. );
754. HRESULT GetAllCustData(
755. [out] CUSTDATA \* pCustData
756. );
757. HRESULT GetAllFuncCustData(
758. [in] UINT index,
759. [out] CUSTDATA \* pCustData
760. );
761. HRESULT GetAllParamCustData(
762. [in] UINT indexFunc,
763. [in] UINT indexParam,
764. [out] CUSTDATA \* pCustData
765. );
766. HRESULT GetAllVarCustData(
767. [in] UINT index,
768. [out] CUSTDATA \* pCustData
769. );
770. HRESULT GetAllImplTypeCustData(
771. [in] UINT index,
772. [out] CUSTDATA \* pCustData
773. );
774. }
775. [
776. object,
777. uuid(00020402-0000-0000-C000-000000000046),
778. pointer\_default(unique)
779. ]
780. interface ITypeLib : IUnknown
781. {
782. HRESULT GetTypeInfoCount(
783. [out] UINT \* pcTInfo
784. );
785. HRESULT GetTypeInfo(
786. [in] UINT index,
787. [out] ITypeInfo \*\* ppTInfo
788. );
789. HRESULT GetTypeInfoType(
790. [in] UINT index,
791. [out] TYPEKIND \* pTKind
792. );
793. HRESULT GetTypeInfoOfGuid(
794. [in] REFGUID guid,
795. [out] ITypeInfo \*\* ppTInfo
796. );
797. HRESULT GetLibAttr(
798. [out] LPTLIBATTR \* ppTLibAttr,
799. [out] DWORD \* pReserved
800. );
801. HRESULT GetTypeComp(
802. [out] ITypeComp \*\* ppTComp
803. );
804. HRESULT GetDocumentation(
805. [in] INT index,
806. [in] DWORD refPtrFlags,
807. [out] BSTR \* pBstrName,
808. [out] BSTR \* pBstrDocString,
809. [out] DWORD \* pdwHelpContext,
810. [out] BSTR \* pBstrHelpFile
811. );
812. HRESULT IsName(
813. [in] LPOLESTR szNameBuf,
814. [in] ULONG lHashVal,
815. [out] BOOL \* pfName,
816. [out] BSTR \* pBstrNameInLibrary
817. );
818. HRESULT FindName(
819. [in] LPOLESTR szNameBuf,
820. [in] ULONG lHashVal,
821. [out,size\_is(\*pcFound),length\_is(\*pcFound)] ITypeInfo \*\*ppTInfo,
822. [out,size\_is(\*pcFound),length\_is(\*pcFound)] MEMBERID \* rgMemId,
823. [in, out] USHORT \* pcFound,
824. [out] BSTR \* pBstrNameInLibrary
825. );
826. HRESULT Opnum12NotUsedOnWire(
827. void
828. );
829. }
830. [
831. object,
832. uuid(00020411-0000-0000-C000-000000000046),
833. pointer\_default(unique)
834. ]
835. interface ITypeLib2 : ITypeLib
836. {
837. HRESULT GetCustData(
838. [in] REFGUID guid,
839. [out] VARIANT \* pVarVal
840. );
841. HRESULT GetLibStatistics(
842. [out] ULONG \* pcUniqueNames,
843. [out] ULONG \* pcchUniqueNames
844. );
845. HRESULT GetDocumentation2(
846. [in] INT index,
847. [in] LCID lcid,
848. [in] DWORD refPtrFlags,
849. [out] BSTR \*pbstrHelpString,
850. [out] DWORD \*pdwHelpStringContext,
851. [out] BSTR \*pbstrHelpStringDll
852. );
853. HRESULT GetAllCustData(
854. [out] CUSTDATA \* pCustData
855. );
856. }

# Appendix B: Product Behavior

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

* Windows NT operating system
* Windows 2000 operating system
* Windows XP operating system
* Windows Server 2003 operating system
* Windows Vista operating system
* Windows Server 2008 operating system
* Windows 7 operating system
* Windows Server 2008 R2 operating system
* Windows 8 operating system
* Windows Server 2012 operating system
* Windows 8.1 operating system
* Windows Server 2012 R2 operating system
* Windows 10 operating system
* Windows Server 2016 operating system

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

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

[<1> Section 1.8](#Appendix_A_Target_1): Windows uses only Windows Errors Codes, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

[<2> Section 2.2.15](#Appendix_A_Target_2): For type libraries that are generated by means of the [**Microsoft Interface Definition Language (MIDL)**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c), a parameter that has the [custom] attribute does not specify PARAMFLAG\_FHASCUSTDATA. For type libraries that are generated by means of *MkTypLib*, a parameter that has the [custom] attribute always specifies PARAMFLAG\_FHASCUSTDATA.

[<3> Section 2.2.16](#Appendix_A_Target_3): The TYPEFLAG\_FDISPATCHABLE flag value is computed based on the presence of IDispatch. It is never set directly.

[<4> Section 2.2.20](#Appendix_A_Target_4): Type libraries generated by means of Microsoft Interface Definition Language (MIDL) always specify LIBFLAG\_FHASDISKIMAGE.

[<5> Section 2.2.28.2.1](#Appendix_A_Target_5): Windows uses IID\_IRecordInfo as the [**IID**](#gt_76ad3105-3f05-479d-a40c-c9c8fa2ebd83) of a local-only interface.

[<6> Section 2.2.29.1](#Appendix_A_Target_6): wReserved1 is not set to 0 by Windows automation clients.

[<7> Section 2.2.29.1](#Appendix_A_Target_7): wReserved2 is not set to 0 by Windows automation clients.

[<8> Section 2.2.29.1](#Appendix_A_Target_8): wReserved3 is not set to 0 by Windows automation clients.

[<9> Section 2.2.29.2](#Appendix_A_Target_9): Windows uses these data type names when defining the local Windows **VARIANT** data types, and another set of data types whose names are prefixed by "\_wire", such as **\_wireVARIANT**, to define the wire formats for these data types. Because the local Windows data types are not used on the network, the protocol specification uses the original data type names such as "VARIANT" when specifying wire format data type definitions for VARIANT data types".

[<10> Section 2.2.30.10](#Appendix_A_Target_10): Windows uses these data type names when defining the local Windows **SAFEARRAY** data types, and another set of data types whose names are prefixed by "\_wire", such as **\_wireSAFEARRAY**, to define the wire formats for these data types. Because the local Windows data types are not used on the network, the protocol specification uses the original data type names such as "SAFEARRAY" when specifying wire format data type definitions for SAFEARRAY data types.

[<11> Section 2.2.30.10](#Appendix_A_Target_11): The low word of **cLocks** represents the number of times the [SAFEARRAY](#Section_2e87a537930541c6a88bb79809b3703a) was "locked" using the SafeArrayAccessData API. For more information, see [[MSDN-SafeArrayAccessData]](https://go.microsoft.com/fwlink/?LinkId=90119).

[<12> Section 2.2.30.10](#Appendix_A_Target_12): The consistency checks are not enforced in Windows NT, Windows 2000 and Windows XP without SP 2. If any of the consistency checks fails, the protocol implementation raises an RPC\_X\_BAD\_STUB\_DATA exception.

[<13> Section 2.2.31](#Appendix_A_Target_13): On Windows platforms, the type library that defines the UDT is registered on both the client and the server.

[<14> Section 2.2.34](#Appendix_A_Target_14): The **wCode** field is always set to 0.

[<15> Section 2.2.34](#Appendix_A_Target_15): The **bstrSource** field is set to a textual, human-readable name of the source of the exception, typically the application name of the server.

[<16> Section 2.2.34](#Appendix_A_Target_16): The Windows implementation of the protocol uses any value passed to it by higher-layer software.

[<17> Section 2.2.34](#Appendix_A_Target_17): bstrHelpFile can be set to the fully qualified path name of a Help file with more information about the error.

[<18> Section 2.2.34](#Appendix_A_Target_18): dwHelpContext can be set to a help context ID. For more information, see [[MSDN-WinHelp]](https://go.microsoft.com/fwlink/?LinkId=90163).

[<19> Section 2.2.34](#Appendix_A_Target_19): pfnDeferredFillIn can be non-NULL when the [**automation server**](#gt_5dcdba04-9cfd-40b3-b0e1-0b8e4374aeaa) implementing IDispatch sets it to a non-NULL value. This function is meant to defer the need to fill in the rest of the structure until the client actually requests it. This value is bound to the server process address space. When the client and the server are not hosted in the same process, this value is ignored.

[<20> Section 2.2.39](#Appendix_A_Target_20): The value, in bytes, of **cBytes** is the in-memory size of the [PARAMDESCEX](#Section_683c767d2e8e4d2f8804afeb3a73969a) structure.

[<21> Section 2.2.42](#Appendix_A_Target_21): If a MIDL-generated type library has an [lcid] parameter following the [optional] parameters, cParamsOpt is set to 0. To count the optional parameters specified by the method, iterate through the members of the lprgelemdescParam array and evaluate the paramdesc.wParamFlags bit flags of each element. Each optional parameter must have the PARAMFLAG\_FOPT bit flag set.

[<22> Section 2.2.43](#Appendix_A_Target_22): For a per-instance field, **\_vdUnion** specifies the offset of the field in memory relative to the starting address of the structure, or 0 if the [VARDESC](#Section_ae7791d243994dffb7c6b0d4f3dce982) describes a member of a union.

[<23> Section 2.2.44](#Appendix_A_Target_23): The sizes of data-only types in Windows are specified in [[MSDN]](https://go.microsoft.com/fwlink/?LinkId=124362). The size of a structure is specified in [MSDN].

[<24> Section 2.2.44](#Appendix_A_Target_24): The sizes of data-only types in Windows are specified in [MSDN]. The size of a structure is specified in [MSDN].

[<25> Section 2.2.44](#Appendix_A_Target_25): The sizes of data-only types in Windows are specified in [MSDN]. The size of a structure is specified in [MSDN].

[<26> Section 2.2.44](#Appendix_A_Target_26): The sizes of data-only types in Windows are specified in [MSDN]. The size of a structure is specified in [MSDN].

[<27> Section 2.2.44](#Appendix_A_Target_27): Windows does not use the value of the cbAlignment field. Windows sets this value to the required byte alignment for an instance of the type, as in the following table.

| Value | Meaning |
| --- | --- |
| 0 | Specifies alignment with a 64-KB boundary. |
| 1 | Specifies byte alignment. |
| 2 | Specifies word alignment. |
| 4 | Specifies dword alignment. |

[<28> Section 2.2.49](#Appendix_A_Target_28): There are two Windows compilers that process [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) specifications that contain automation definitions: mktyplib.exe and midl.exe. Mktyplib.exe has been deprecated, so do not use it. Mktyplib accepts only a subset of the following specified syntax and keywords, while midl.exe accepts all of them. The OLE Automation Protocol supports the entire range.

[<29> Section 2.2.49.1.2](#Appendix_A_Target_29): Connectable servers implement the following interfaces: IConnectionPointContainer, IConnectionPoint, IEnumConnectionPoints, and IEnumConnections as described in [[MSDN-COM]](https://go.microsoft.com/fwlink/?LinkId=89977).

[<30> Section 2.2.49.1.3](#Appendix_A_Target_30): Clients implement IPropertyNotifySink::OnChanged to handle calls from [**bindable server**](#gt_7d037fdd-af75-47fe-a235-d1bb937aa424) properties that are compiled with the [bindable] attribute and IPropertyNotifySink::OnRequestEdit to handle calls from properties that are declared with the [requestedit] attribute. Both methods identify each property by its [**DISPID**](#gt_3792c5cc-783c-4b2a-a71e-c1ec3f432474). The proposed replacement value is not available to IPropertyNotifySink::OnRequestEdit; so its use is limited to determining whether the existing value can be changed. The value cannot be used for data validation.

[<31> Section 2.2.49.2](#Appendix_A_Target_31): The value of the **[helpcontext]** attribute specifies a 32-bit context identifier that is used to associate the library, type, or type member with a topic in the Help file.

[<32> Section 2.2.49.2](#Appendix_A_Target_32): The value of the **[helpfile]** attribute specifies the fully qualified name of the Help file that is used by all types in the type library.

[<33> Section 2.2.49.2](#Appendix_A_Target_33): The value of the **[helpstring]** attribute provides a description of the element to which it is applied.

[<34> Section 2.2.49.2](#Appendix_A_Target_34): The value of the **[helpstringcontext]** attribute specifies a 32-bit identifier that is used to associate the library, type, or type member with a string resource in the DLL specified by the [helpstringdll] attribute.

[<35> Section 2.2.49.2](#Appendix_A_Target_35): The value of the **[helpstringdll]** attribute specifies the fully qualified name of a dynamic link library that contains localized Help resources.

[<36> Section 2.2.49.2](#Appendix_A_Target_36): Windows uses the **[restricted]** attribute to indicate that an [**interface**](#gt_95913fbd-3262-47ae-b5eb-18e6806824b9) or [**dispinterface**](#gt_35b38e6d-42ee-4b7b-8ca1-ab93308458d5) cannot be available to macro languages. For libraries and modules, it is a visibility attribute with the same meaning as the **[hidden]** attribute: do not display to users.

[<37> Section 2.2.49.3](#Appendix_A_Target_37): There are two Windows compilers that process IDL specifications that contain automation definitions: mktyplib.exe and midl.exe. Mktyplib.exe has been deprecated, so do not use it anymore. Mktyplib accepts only a subset of the types specified earlier in this section, while midl.exe accepts all of them. The OLE Automation Protocol supports the entire range.

[<38> Section 2.2.49.3](#Appendix_A_Target_38): Windows uses the **[restricted]** attribute to indicate that an interface or dispinterface cannot be available to macro languages. For libraries and modules, it is a visibility attribute with the same meaning as the **[hidden]** attribute: do not display to users.

[<39> Section 2.2.49.4](#Appendix_A_Target_39): Windows uses the [proxy] attribute to specify that the code for marshaling the interface data is external to the type library.

[<40> Section 2.2.49.5.1](#Appendix_A_Target_40): Windows uses the **[defaultcollelem]** attribute to enable Visual Basic–specific optimizations, some of which treat the property as the default collection of the [**coclass**](#gt_670b0ee2-d101-41b0-ac77-6ac7dbeee7dc) that contains it. In cases where the application of the attribute is inconsistent (such as coclasses with multiple **[defaultcollelem]** assignments or an assignment to a property that returns objects that are not enumerable), some or all of the optimizations are not performed, and the attribute is ignored.

[<41> Section 2.2.49.5.1](#Appendix_A_Target_41): MIDL does not enforce a restriction on the number of properties with the **[defaultcollelem]** attribute, but some Visual Basic–specific optimizations are not applied if a type has more than one property.

[<42> Section 2.2.49.5.1](#Appendix_A_Target_42): Windows type libraries do not use the **replaceable** attribute.

[<43> Section 2.2.49.5.2](#Appendix_A_Target_43): Windows uses the [immediatebind] attribute to distinguish between controls such as check boxes (in which the bound data source is updated every time the control state changes), and list boxes (in which the bound data source is updated only when the control is saved or loses focus).

[<44> Section 2.2.49.8](#Appendix_A_Target_44): Windows type browsers distinguish between COM components that explicitly support a windowed user interface and components that do not. Non-visual type browsers do not display components with the **control** attribute to users.

[<45> Section 2.2.49.8](#Appendix_A_Target_45): By default, Windows type browsers do not display elements with the **hidden** attribute to users.

[<46> Section 2.2.49.8](#Appendix_A_Target_46): Coclasses defined with the [licensed] attribute can be instantiated using only the IClassFactory2 interface.

[<47> Section 2.2.49.8](#Appendix_A_Target_47): Coclasses defined with the [noncreatable] attribute cannot be instantiated using IClassFactory::CreateInstance, CoCreateInstance, or OleCreate.

[<48> Section 2.2.49.9](#Appendix_A_Target_48): The FUNCFLAG\_FUSESGETLASTERROR bit flag indicates that the method was declared with the **[usesgetlasterror]** attribute and supports the **GetLastError** method (see [[MSDN-ErrorHandling]](https://go.microsoft.com/fwlink/?LinkId=94931)). The **GetLastError** method is local-only and this flag has no effect on the wire.

[<49> Section 2.2.49.9](#Appendix_A_Target_49): The cdecl, stdcall, and pascal calling conventions are specified in [[MSDN-CALLCONV]](https://go.microsoft.com/fwlink/?LinkId=94981).

[<50> Section 2.2.49.10](#Appendix_A_Target_50): The string specified in an importlib statement specifies the fully qualified name of a compiled type library (\*.tlb) file.

[<51> Section 2.2.51](#Appendix_A_Target_51): Windows uses the hash value to quickly reject names that do not correspond to any entities defined in the [**automation type library**](#gt_4057fc86-007f-496f-a966-01c762088a8e).

[<52> Section 2.2.51](#Appendix_A_Target_52): If the hash value is zero, Windows computes a new hash value before evaluating the name.

[<53> Section 3.1.1](#Appendix_A_Target_53): Windows automation servers can generate the mappings on the fly according to the requirements of the application.

[<54> Section 3.1.4.3](#Appendix_A_Target_54): The range restriction is not present for Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<55> Section 3.1.4.3](#Appendix_A_Target_55): The default implementation of Automation performs this mapping; however, any automation server can override this behavior by providing its own implementation for [IDispatch::GetIDsOfNames](#Section_7166d6ffb8514216bfaa34128274a242).

[<56> Section 3.1.4.4.3](#Appendix_A_Target_56): Windows [**automation clients**](#gt_1beeac58-f059-4b12-ad3d-79384cb65c28) use the value specified in the *defaultvalue* parameter.

[<57> Section 3.1.4.4.4](#Appendix_A_Target_57): The default Automation implementation does attempt to convert the actual arguments to the formal parameters' type, as declared in the IDL of the method or property. If no such conversion exists, the default Automation implementation returns DISP\_E\_TYPEMISMATCH. However, any automation server can choose to implement [IDispatch::Invoke](#Section_5c2a199760d7496d8d9aed940bbb82eb) and exhibit different behavior

[<58> Section 3.7.1.1](#Appendix_A_Target_58): The implementation-specific documentation values correspond to the values declared with the [helpstring], [helpcontext], and [helpfile] attributes. If the server also implements [ITypeInfo2](#Section_2d6024dad2294d78bbb0b9d5bf6459b7), the documentation values include the values declared with the [helpstringcontext] and [helpstringdll] attributes.

[<59> Section 3.7.4.8](#Appendix_A_Target_59): If the library, type, or type member was declared without the [helpstring] attribute and the Type information server implements ITypeInfo2, the [GetDocumentation](#Section_2ea2f705bc334cecbbc7613d6ae0f0c6) method attempts to return the localized value specified by the *pBstrHelpString* parameter of [ITypeInfo2::GetDocumentation2](#Section_541262a3d8704c8ebe311eb6ab1d9259), using an LCID of 0.

[<60> Section 3.9.4.10](#Appendix_A_Target_60): *pbstrHelpString* is set to the value of the string resource that is contained in the DLL specified by *pBstrHelpStringDll* and that is associated with the resource handle specified by *pdwHelpStringContext* and LocaleID specified by *lcid*, or is set to NULL if no such resource exists.

[<61> Section 3.11.4.7](#Appendix_A_Target_61): If the library or type was declared without the [helpstring] attribute and the Type library server implements ITypeLib2, the [GetDocumentation](#Section_ceb2d9eb975a47019a793bb9e6ad419b) method attempts to return the localized value specified by the *pBstrHelpString* parameter of [ITypeLib2::GetDocumentation2](#Section_137253c0736e4616833c527ef2dc1618), using an LCID of 0.

[<62> Section 3.11.4.9](#Appendix_A_Target_62): Matching members of the binding member table of a [**reference dispinterface**](#gt_dc320a3c-71b6-4055-bfd1-d9fa6f3f770f) that are defined outside the [**automation scope**](#gt_07ad91dc-c12a-44f7-9c64-7e496933183a) are included in the *ppTInfo* and *rgMemId* arrays if the automation scope includes at least two named non-parameter elements whose names match *szNameBuf*.

[<63> Section 3.13.4.3](#Appendix_A_Target_63): *pBstrHelpString* is set to the value of the string resource contained in the DLL specified by *pBstrHelpStringDll* and associated with the resource handle specified by *pdwHelpStringContext* and LocaleID specified by *lcid*, or NULL if no such resource exists.

# Appendix C: Full ABNF

1. start-rule =
2. \*( interface / import / export / oa-scope / oa-dispinterface )
3. ; Automation scope
4. oa-scope = oa-library-header LWSP "{" oa-library-body "}" LWSP [";"]
5. oa-library-header =
6. "[" LWSP library-attributes LWSP "]" LWSP kw-library LWSP
7. Identifier
8. oa-library-body = \*oa-library-declarator
9. library-attributes =
10. library-attribute \*( "," LWSP library-attribute LWSP )
11. library-attribute = uuid-attr /
12. version-attr /
13. lcid-attr /
14. help-attr /
15. custom-attr /
16. kw-control /
17. kw-hidden /
18. kw-restricted
19. uuid-attr = kw-uuid LWSP "(" LWSP uuid-rep LWSP ")"
20. version-attr =
21. kw-version LWSP "(" LWSP 1\*DIGIT \*( "." 1\*DIGIT ) LWSP ")"
22. lcid-attr = kw-lcid LWSP "(" LWSP integer-const-exp LWSP ")"
23. help-attr = helpcontext-attr /
24. helpfile-attr /
25. helpstring-attr /
26. helpstringcontext-attr /
27. helpstringdll-attr
28. helpcontext-attr =
29. kw-helpcontext LWSP "(" LWSP integer-const-exp LWSP ")"
30. helpfile-attr = kw-helpfile LWSP "(" LWSP string LWSP ")"
31. helpstring-attr = kw-helpstring LWSP "(" LWSP string LWSP ")"
32. helpstringcontext-attr =
33. kw-helpstringcontext LWSP "(" LWSP integer-const-exp LWSP ")"
34. helpstringdll-attr = kw-helpstringdll LWSP "(" LWSP string LWSP ")"
35. custom-attr =
36. kw-custom LWSP "(" uuid-rep LWSP "," LWSP const-exp LWSP ")"
37. oa-library-declarator = interface /
38. import /
39. export /
40. oa-importlib /
41. oa-module /
42. oa-dispinterface /
43. oa-coclass
44. type-attribute = rpcidl-defined /
45. uuid-attr /
46. help-attr /
47. custom-attr /
48. kw-public /
49. kw-restricted
50. ; Automation Compatible Types
51. oa-type-spec = oa-base-type-spec /
52. oa-safearray-type-spec /
53. oa-ptr-type-spec /
54. Identifier
55. oa-base-type-spec = oa-base-nondecimal-type-spec / kw-Decimal
56. oa-base-nondecimal-type-spec = kw-boolean /
57. [kw-unsigned] LWSP kw-char /
58. [kw-unsigned] LWSP kw-short /
59. [kw-unsigned] LWSP kw-int /
60. [kw-unsigned] LWSP kw-long /
61. kw-double /
62. kw-float /
63. kw-BSTR /
64. kw-CURRENCY /
65. kw-DATE /
66. kw-SCODE
68. oa-safearray-type-spec =
69. kw-SAFEARRAY "(" oa-base-nondecimal-type-spec ")" /
70. kw-SAFEARRAY "(" oa-ptr-type-spec ")" /
71. kw-SAFEARRAY "(" Identifier ")"
72. oa-ptr-type-spec = oa-base-nondecimal-type-spec "\*" /
73. oa-safearray-type-spec "\*" /
74. Identifier "\*"
75. ; Automation Interfaces
76. interface-attribute = rpcidl-defined /
77. attr-oleautomation /
78. attr-dual /
79. kw-nonextensible /
80. kw-proxy /
81. custom-attr
82. attr-oleautomation = kw-oleautomation
83. attr-dual = kw-dual
84. oa-dispinterface =
85. oa-dispitf-header LWSP "{" LWSP oa-dispitf-body LWSP "}"
86. oa-dispitf-header = "[" interface-attributes "]"
87. LWSP kw-dispinterface LWSP Identifier
88. oa-dispitf-body = oa-itf-ref / oa-odl-body
89. oa-itf-ref = kw-interface LWSP Identifier LWSP ";"
90. ; Automation Members
91. operation-attribute = rpcidl-defined /
92. kw-id LWSP "(" LWSP integer-const-exp LWSP ")" /
93. kw-propget /
94. kw-propput /
95. kw-propputref /
96. kw-vararg /
97. kw-defaultcollelem /
98. kw-nonbrowsable /
99. kw-replaceable /
100. kw-restricted /
101. kw-uidefault /
102. kw-hidden /
103. oa-bindable-attr /
104. readonly-attr /
105. help-attr /
106. custom-attr
107. oa-bindable-attr = kw-bindable /
108. kw-immediatebind /
109. kw-defaultbind /
110. kw-displaybind /
111. kw-requestedit
112. oa-odl-body =
113. kw-properties LWSP ":" LWSP \*oa-odl-prop LWSP
114. kw-methods LWSP ":" LWSP \*oa-odl-method
115. oa-odl-prop =
116. \*( operation-attributes ) LWSP oa-type-spec LWSP
117. Identifier LWSP ";"
118. oa-odl-method = op-declarator
119. readonly-attr = kw-readonly
121. ; Automation Parameters
122. param-attribute = rpcidl-defined /
123. kw-defaultvalue LWSP "(" LWSP const-exp LWSP ")" /
124. kw-optional /
125. kw-lcid /
126. kw-retval /
127. custom-attr
128. oa-coclass = "[" LWSP oa-coclass-attrs LWSP "]" LWSP
129. kw-coclass LWSP Identifier
130. LWSP "{" LWSP oa-coclass-body LWSP "}"
131. oa-coclass-attrs = oa-coclass-attr \*( LWSP "," LWSP
132. oa-coclass-attr)
133. oa-coclass-attr = uuid-attr /
134. help-attr /
135. version-attr /
136. custom-attr /
137. kw-aggregatable /
138. kw-appobject /
139. kw-control /
140. kw-hidden /
141. kw-licensed /
142. kw-noncreatable /
143. kw-predeclid
144. oa-coclass-body = \*( oa-coclass-itf-decl )
145. oa-coclass-itf-decl = [ "[" LWSP oa-cid-attrs LWSP "]" LWSP ]
146. ( kw-interface / kw-dispinterface ) LWSP Identifier LWSP ";"
147. oa-cid-attrs = oa-cid-attr \*( LWSP "," LWSP oa-cid-attr )
148. oa-cid-attr = kw-source /
149. kw-default /
150. kw-defaultvtable /
151. kw-restricted
152. oa-module = [ "[" LWSP oa-module-attrs LWSP "]" LWSP ]
153. kw-module LWSP Identifier LWSP
154. "{" oa-module-body "}"
155. oa-module-attrs = oa-module-attr \*( LWSP "," LWSP oa-module-attr )
156. oa-module-attr = uuid-attr /
157. version-attr /
158. help-attr /
159. kw-dllname LWSP "(" LWSP string LWSP ")" /
160. kw-hidden
161. oa-module-body = \*( oa-const-stmt / oa-mmethod-stmt )
162. oa-const-stmt = [ "[" LWSP \*( help-attr ) LWSP "]" LWSP ]
163. ( kw-const / kw-static ) LWSP
164. oa-base-type-spec LWSP Identifier LWSP
165. "=" LWSP const-exp LWSP ";"
166. oa-mmethod-stmt = [ "[" LWSP oa-mmethod-attrs LWSP "]" LWSP ]
167. oa-type-spec LWSP [ oa-mmethod-cc LWSP ]
168. Identifier LWSP param-declarators LWSP ";"
169. oa-mmethod-attrs = oa-mmethod-attr \*( LWSP "," LWSP oa-mmethod-attr )
170. oa-mmethod-attr =
171. kw-entry LWSP "(" LWSP oa-entry-id LWSP ")" /
172. kw-propget /
173. kw-propput /
174. kw-propputref /
175. kw-usesgetlasterror /
176. kw-vararg /
177. help-attr
178. oa-entry-id = string / integer-const-exp
179. oa-mmethod-cc = kw-cdecl /
180. kw-stdcall /
181. kw-pascal
182. oa-importlib = kw-importlib LWSP "(" LWSP string LWSP ")" LWSP ";"
183. ; Automation keywords: case sensitive
184. kw-aggregatable = %d97.103.103.114.101.103.97.116.97.98.108.101
185. kw-appobject = %d97.112.112.111.98.106.101.99.116
186. kw-bindable = %d98.105.110.100.97.98.108.101
187. kw-boolean = %d98.111.111.108.101.97.110
188. kw-BSTR = %d66.83.84.82
189. kw-cdecl = %d99.100.101.99.108
190. kw-char = %d99.104.97.114
191. kw-coclass = %d99.111.99.108.97.115.115
192. kw-const = %d99.111.110.115.116
193. kw-control = %d99.111.110.116.114.111.108
194. kw-CURRENCY = %d67.85.82.82.69.78.67.89
195. kw-custom = %d99.117.115.116.111.109
196. kw-DATE = %d68.65.84.69
197. kw-Decimal = %d68.101.99.105.109.97.108
198. kw-default = %d100.101.102.97.117.108.116
199. kw-defaultbind = %d100.101.102.97.117.108.116.98.105.110.100
200. kw-defaultcollelem =
201. %d100.101.102.97.117.108.116.99.111.108.108.101.108.101.109
202. kw-defaultvalue =
203. %d100.101.102.97.117.108.116.118.97.108.117.101
204. kw-defaultvtable =
205. %d100.101.102.97.117.108.116.118.116.97.98.108.101
206. kw-dispinterface =
207. %d100.105.115.112.105.110.116.101.114.102.97.99.101
208. kw-displaybind =
209. %d100.105.115.112.108.97.121.98.105.110.100
210. kw-dllname = %d100.108.108.110.97.109.101
211. kw-double = %d100.111.117.98.108.101
212. kw-dual = %d100.117.97.108
213. kw-entry = %d101.110.116.114.121
214. kw-float = %d102.108.111.97.116
215. kw-helpcontext = %d104.101.108.112.99.111.110.116.101.120.116
216. kw-helpfile = %d104.101.108.112.102.105.108.101
217. kw-helpstring = %d104.101.108.112.115.116.114.105.110.103
218. kw-helpstringcontext =
219. %d104.101.108.112.115.116.114.105.110.103.99.111.110.116.101.120.116
220. kw-helpstringdll =
221. %d104.101.108.112.115.116.114.105.110.103.100.108.108
222. kw-hidden = %d104.105.100.100.101.110
223. kw-id = %d105.100
224. kw-immediatebind =
225. %d105.109.109.101.100.105.97.116.101.98.105.110.100
226. kw-importlib = %d105.109.112.111.114.116.108.105.98
227. kw-int = %d105.110.116
228. kw-interface = %d105.110.116.101.114.102.97.99.101
229. kw-lcid = %d108.99.105.100
230. kw-library = %d108.105.98.114.97.114.121
231. kw-licensed = %d108.105.99.101.110.115.101.100
232. kw-long = %d108.111.110.103
233. kw-methods = %d109.101.116.104.111.100.115
234. kw-module = %d109.111.100.117.108.101
235. kw-nonbrowsable = %d110.111.110.98.114.111.119.115.97.98.108.101
236. kw-noncreatable = %d110.111.110.99.114.101.97.116.97.98.108.101
237. kw-nonextensible =
238. %d110.111.110.101.120.116.101.110.115.105.98.108.101
239. kw-oleautomation =
240. %d111.108.101.97.117.116.111.109.97.116.105.111.110
241. kw-optional = %d111.112.116.105.111.110.97.108
242. kw-pascal = %d112.97.115.99.97.108
243. kw-predeclid = %d112.114.101.100.101.99.108.105.100
244. kw-properties = %d112.114.111.112.101.114.116.105.101.115
245. kw-propget = %d112.114.111.112.103.101.116
246. kw-propput = %d112.114.111.112.112.117.116
247. kw-propputref = %d112.114.111.112.112.117.116.114.101.102
248. kw-proxy = %d112.114.111.120.121
249. kw-public = %d112.117.98.108.105.99
250. kw-readonly = %d114.101.97.100.111.110.108.121
251. kw-replaceable = %d114.101.112.108.97.99.101.97.98.108.101
252. kw-requestedit = %d114.101.113.117.101.115.116.101.100.105.116
253. kw-restricted = %d114.101.115.116.114.105.99.116.101.100
254. kw-retval = %d114.101.116.118.97.108
255. kw-SAFEARRAY = %d83.65.70.69.65.82.82.65.89
256. kw-SCODE = %d83.67.79.68.69
257. kw-short = %d115.104.111.114.116
258. kw-source = %d115.111.117.114.99.101
259. kw-static = %d115.116.97.116.105.99
260. kw-stdcall = %d115.116.100.99.97.108.108
261. kw-uidefault = %d117.105.100.101.102.97.117.108.116
262. kw-unsigned = %d117.110.115.105.103.110.101.100
263. kw-uuid = %d117.117.105.100
264. kw-usesgetlasterror =
265. %d117.115.101.115.103.101.116.108.97.115.116.101.114.114.111.114
266. kw-vararg = %d118.97.114.97.114.103
267. kw-version = %d118.101.114.115.105.111.110
268. ; Rules defined in the [C706] IDL specification
269. uuid-rep = rpcidl-defined
270. string = rpcidl-defined
271. const-exp = rpcidl-defined
272. integer-const-exp = rpcidl-defined
273. operation-attributes = rpcidl-defined
274. op-declarator = rpcidl-defined
275. interface-attributes = rpcidl-defined
276. interface = rpcidl-defined
277. import = rpcidl-defined
278. export = rpcidl-defined
279. param-declarators = rpcidl-defined
280. Identifier = rpcidl-defined
281. rpcidl-defined = "already defined"
282. ;Tokens
283. ; The ABNF Core rules
284. ALPHA = %x41-5A / %x61-7A ; A-Z / a-z
285. BIT = "0" / "1"
286. CHAR = %x01-7F
287. ; any 7-bit US-ASCII character,
288. ; excluding NUL
289. CR = %x0D
290. ; carriage return
291. CRLF = CR LF
292. ; Internet standard newline
293. CTL = %x00-1F / %x7F
294. ; controls
295. DIGIT = %x30-39
296. ; 0-9
297. DQUOTE = %x22
298. ; " (Double Quote)
299. HEXDIG = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
300. HTAB = %x09
301. ; horizontal tab
302. LF = %x0A
303. ; linefeed
304. LWSP = \*(WSP / CRLF WSP)
305. ; linear white space (past newline)
306. OCTET = %x00-FF
307. ; 8 bits of data
308. SP = %x20
309. VCHAR = %x21-7E
310. ; visible (printing) characters
311. WSP = SP / HTAB
312. ; white space

# Change Tracking

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

# Index

A

[ABNF](#section_2b2b5513b36a41b7a114e955f80332b3) 181

Abstract data model

[automation client](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95

[automation server](#section_74f390deb6fc42548c322a8730ca38b4) 88

client ([section 3.2.1](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95, [section 3.4.1](#section_253a87eb0f7c405d9524caf1a5cdf357) 99, [section 3.6.1](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105, [section 3.8.1](#section_5f1a181dce344b6d9e5d8602574c8102) 119, [section 3.10.1](#section_8a2130ee60354c6f9e5334559a2c879b) 131, [section 3.12.1](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140, [section 3.14.1](#section_6f07e35d587b432c86776a8047037163) 144)

[IEnumVARIANT client](#section_253a87eb0f7c405d9524caf1a5cdf357) 99

[IEnumVARIANT server](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96

[ITypeComp client](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105

[ITypeComp server](#section_d1644be480344e5f89e87d55c8625b14) 100

[ITypeInfo client](#section_5f1a181dce344b6d9e5d8602574c8102) 119

ITypeInfo server

[common Automation type description elements](#section_59e465b3b60a41948231fdb99b513633) 106

[overview](#section_1e542e10fe4d475f96156b6d956b7073) 105

[TYPEKIND-dependent Automation type description elements](#section_7b1b8bd1a0674edb9d726aa500d035a3) 107

[ITypeInfo2 client](#section_8a2130ee60354c6f9e5334559a2c879b) 131

[ITypeInfo2 server](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120

[ITypeLib client](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140

[ITypeLib server](#section_87fd9a39606742a7b8e613637df3bd0d) 132

[ITypeLib2 client](#section_6f07e35d587b432c86776a8047037163) 144

[ITypeLib2 server](#section_d1f11539021541839b25e1588b061531) 141

server ([section 3.1.1](#section_74f390deb6fc42548c322a8730ca38b4) 88, [section 3.3.1](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96, [section 3.5.1](#section_d1644be480344e5f89e87d55c8625b14) 100, [section 3.7.1](#section_1e542e10fe4d475f96156b6d956b7073) 105, [section 3.9.1](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120, [section 3.11.1](#section_87fd9a39606742a7b8e613637df3bd0d) 132, [section 3.13.1](#section_d1f11539021541839b25e1588b061531) 141)

[ADVFEATUREFLAGS enumeration](#section_f06ee3d2a61f4e0ba9299369d334ea33) 28

[Aggregatable servers](#section_491b9a5d4b3548a5b7ca5d6a8f3456aa) 63

[AIDL interfaces](#section_18d74e75f9a744079fe83406679f7dd8) 72

AIDL-ODL equivalence examples

[method](#section_7df4503ea14b44ddb81908a381609b9d) 146

[property](#section_0fd123156d004d8883a40093d78156a4) 146

[Aidl-odl method equivalence example](#section_7df4503ea14b44ddb81908a381609b9d) 146

[Aidl-odl property equivalence example](#section_0fd123156d004d8883a40093d78156a4) 146

[Applicability](#section_9b4886bf2e3a4957a9136ee4aaf91d85) 18

[Argument coercion - IDispatch::Invoke](#section_5c01ab3cf71944ccbb45d36850cf4c5b) 94

[Argument-parameter mapping - IDispatch::Invoke](#section_9cf379f7fb3141fe9f9cc9a0136616e0) 94

[ARRAYDESC structure](#section_2e06e2b6054e48b1b867ad1e87a7ebe2) 53

Automation

[hash values](#section_7a8ed4c314a4433cbaa5b6ec88135352) 77

[interfaces](#section_3b4b512c8c9445a1810dba4ff0152698) 68

[members](#section_da55c4194395453582c4bac998dae862) 69

[parameters](#section_ff2bd74bcb4d48b3ab896bb32cda3833) 72

[scope](#section_ae4d27fafaaa4d5fb0f915bfbaaae9b4) 64

Automation client

[abstract data model](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95

[initialization](#section_407ee4e6921d43c3a952632f26a4a9cf) 95

[local events](#section_c646b09f59bb4192a970f520157f3fdb) 96

[message processing](#section_c4ebe7bc21754fd0af542653820f435c) 95

[sequencing rules](#section_c4ebe7bc21754fd0af542653820f435c) 95

[timer events](#section_c02df6e407cc4cc8811f900791eee84e) 95

[timers](#section_006bdf07d2de4ebf8bbdad3dfc3b2fcd) 95

[automation interface](#section_c2c7dbe2bafa49da93a77b75499ef90a) 88

Automation server

[abstract data model](#section_74f390deb6fc42548c322a8730ca38b4) 88

[initialization](#section_bc5d2d2181854ff58fa55d2f7de45df9) 88

[local events](#section_36c98baac2cf40d1934b32d85d9bb4b1) 94

[message processing](#section_ac9c502bac1c42028ad4048ac98afcc9) 89

[overview](#section_c2c7dbe2bafa49da93a77b75499ef90a) 88

[sequencing rules](#section_ac9c502bac1c42028ad4048ac98afcc9) 89

[timer events](#section_e9d847bc05194b7da39ba97216389f9a) 94

[timers](#section_39dcc7d407784979909de7434a3455c9) 88

[Automation type description binding context](#section_0f9202a0f3264406899ce57aeefeb52d) 103

Automation type description elements

[common](#section_59e465b3b60a41948231fdb99b513633) 106

[TYPEKIND-dependent](#section_7b1b8bd1a0674edb9d726aa500d035a3) 107

[Automation type library binding context](#section_693e9d4fc27448868e4f07899ba023c7) 102

Automation-compatible

[interfaces](#section_222fe935db004920a7d9583a6b26c45b) 68

[types](#section_7b5fa59bd8f64a479695630d3c10363e) 66

B

[Bind method](#section_476f00da080640d9bbf36059154abbb7) 101

[Bindable properties](#section_ac18004e7af74feb8ae6d42e7a14267c) 70

[Bindable servers](#section_2a89365308e7415a922628f8767c16ba) 63

Binding context - ITypeComp::Bind

[Automation type description](#section_0f9202a0f3264406899ce57aeefeb52d) 103

[Automation type library](#section_693e9d4fc27448868e4f07899ba023c7) 102

[overview](#section_cc88e6254b3a4cf9896626bdc913f62d) 102

[BindType method](#section_cf61a786b1814267bb6a0987eeb17b38) 104

[BNF](#section_2b2b5513b36a41b7a114e955f80332b3) 181

Bound elements - types returned with - ITypeComp::Bind

[ITypeInfo members](#section_ceb4299738644689a04e5e77adf7b2b2) 104

[ITypeLib members](#section_b31964b8e3134bc6ad82fe112f394dd0) 103

[overview](#section_3ca3ffe31f9547159b54574a300ffd88) 103

[BRECORD](#section_ea064b3d9fb3448699924fe2463e83e8) 40

[BSTR](#section_9c5a5ce4ff5b45ceb915ada381b34ac1) 37

[Byref](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[BYTE\_SIZEDARR structure](#section_e27714f46cfb42c3b4df313c2b77a142) 46

C

[CALLCONV enumeration](#section_2e3cc99094f241e5ae6fcdd00414b776) 29

[Calling a method with byref and optional arguments example](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[Calling method example](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[Capability negotiation](#section_aa944ebae686410f9b2c128ec9fd5fc1) 18

[Change tracking](#section_d1f84f6f256b4836aaee029e939b4ebf) 187

Client

abstract data model ([section 3.2.1](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95, [section 3.4.1](#section_253a87eb0f7c405d9524caf1a5cdf357) 99, [section 3.6.1](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105, [section 3.8.1](#section_5f1a181dce344b6d9e5d8602574c8102) 119, [section 3.10.1](#section_8a2130ee60354c6f9e5334559a2c879b) 131, [section 3.12.1](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140, [section 3.14.1](#section_6f07e35d587b432c86776a8047037163) 144)

[ienumvariant interface](#section_079100caf62f47028861a1f64e66e9e2) 99

initialization ([section 3.2.3](#section_407ee4e6921d43c3a952632f26a4a9cf) 95, [section 3.4.3](#section_8b92ea3e75824ba6b12fe44b5e739de5) 99, [section 3.6.3](#section_29b2d79f778c480980b5b60ea1bb94f1) 105, [section 3.8.3](#section_3aab2fdab8704f3bbf0bba9eae9ee746) 119, [section 3.10.3](#section_d137d5baaa4549828dfa5c2c75010937) 131, [section 3.12.3](#section_5a327992cce046539187215ed564fe9d) 140, [section 3.14.3](#section_7ecf29c130a54e8d88d10c36b637e6d9) 144)

local events ([section 3.2.6](#section_c646b09f59bb4192a970f520157f3fdb) 96, [section 3.4.6](#section_85855ba838aa4387bb2b890f07eb2bcf) 100, [section 3.6.6](#section_0716f9922d224145b911bdc2ec94ab32) 105, [section 3.8.6](#section_de0a7e91e237478fb3789e8b4b05d9f5) 120, [section 3.10.6](#section_a18c66ef338c4aed83e8773b83fa410f) 132, [section 3.12.6](#section_27433f25698e4a5c883d3660690827c3) 141, [section 3.14.6](#section_74ca3f6c283f439994eb4611bca35be5) 145)

message processing ([section 3.6.4](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105, [section 3.8.4](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119, [section 3.10.4](#section_cdfd465a484a4649bc80d2420cd13e8b) 131, [section 3.12.4](#section_42f7fe088f65412989924c67d4ecf825) 140, [section 3.14.4](#section_e89644ede3704dd28c6a15c78dd1736a) 144)

[overview](#section_079100caf62f47028861a1f64e66e9e2) 99

sequencing rules ([section 3.6.4](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105, [section 3.8.4](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119, [section 3.10.4](#section_cdfd465a484a4649bc80d2420cd13e8b) 131, [section 3.12.4](#section_42f7fe088f65412989924c67d4ecf825) 140, [section 3.14.4](#section_e89644ede3704dd28c6a15c78dd1736a) 144)

timer events ([section 3.2.5](#section_c02df6e407cc4cc8811f900791eee84e) 95, [section 3.4.5](#section_9d2903537b4b4b1b93a97c2803b02251) 100, [section 3.6.5](#section_1f7d6a20f5b24132952ffb20bf2830d2) 105, [section 3.8.5](#section_83c146e4762c41f39767c4a6e30b6e3f) 119, [section 3.10.5](#section_b739d1cceda74019b1e60f70811e2aaf) 131, [section 3.12.5](#section_a9a2efbe122e43a387cdb3e4a689ed70) 141, [section 3.14.5](#section_b5d711a4c24f4312a1b7375fd570e57d) 145)

timers ([section 3.2.2](#section_006bdf07d2de4ebf8bbdad3dfc3b2fcd) 95, [section 3.4.2](#section_2ac5018fb5cb4083bbd023c633f64954) 99, [section 3.6.2](#section_fc22f1ada5bd452f80289396b26fb480) 105, [section 3.8.2](#section_2379d54297234b55a39099f90c24ab26) 119, [section 3.10.2](#section_c31155253c364dc581ece7affc0c7bbf) 131, [section 3.12.2](#section_0c1e84e201c549b082f87ecc19206416) 140, [section 3.14.2](#section_ca3e8f7089264fc18e5111b5b59b0f10) 144)

Client - automation

[abstract data model](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95

[initialization](#section_407ee4e6921d43c3a952632f26a4a9cf) 95

[local events](#section_c646b09f59bb4192a970f520157f3fdb) 96

[message processing](#section_c4ebe7bc21754fd0af542653820f435c) 95

[sequencing rules](#section_c4ebe7bc21754fd0af542653820f435c) 95

[timer events](#section_c02df6e407cc4cc8811f900791eee84e) 95

[timers](#section_006bdf07d2de4ebf8bbdad3dfc3b2fcd) 95

Client - IEnumVARIANT

[abstract data model](#section_253a87eb0f7c405d9524caf1a5cdf357) 99

[initialization](#section_8b92ea3e75824ba6b12fe44b5e739de5) 99

[local events](#section_85855ba838aa4387bb2b890f07eb2bcf) 100

[message processing](#section_a0d58a4ee3b944c28a69d376deb30259) 99

[overview](#section_079100caf62f47028861a1f64e66e9e2) 99

[sequencing rules](#section_a0d58a4ee3b944c28a69d376deb30259) 99

[timer events](#section_9d2903537b4b4b1b93a97c2803b02251) 100

[timers](#section_2ac5018fb5cb4083bbd023c633f64954) 99

Client - ITypeComp

[abstract data model](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105

[initialization](#section_29b2d79f778c480980b5b60ea1bb94f1) 105

[local events](#section_0716f9922d224145b911bdc2ec94ab32) 105

[message processing](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105

[sequencing rules](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105

[timer events](#section_1f7d6a20f5b24132952ffb20bf2830d2) 105

[timers](#section_fc22f1ada5bd452f80289396b26fb480) 105

Client - ITypeInfo

[abstract data model](#section_5f1a181dce344b6d9e5d8602574c8102) 119

[initialization](#section_3aab2fdab8704f3bbf0bba9eae9ee746) 119

[local events](#section_de0a7e91e237478fb3789e8b4b05d9f5) 120

[message processing](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119

[sequencing rules](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119

[timer events](#section_83c146e4762c41f39767c4a6e30b6e3f) 119

[timers](#section_2379d54297234b55a39099f90c24ab26) 119

Client - ITypeInfo2

[abstract data model](#section_8a2130ee60354c6f9e5334559a2c879b) 131

[initialization](#section_d137d5baaa4549828dfa5c2c75010937) 131

[local events](#section_a18c66ef338c4aed83e8773b83fa410f) 132

[message processing](#section_cdfd465a484a4649bc80d2420cd13e8b) 131

[sequencing rules](#section_cdfd465a484a4649bc80d2420cd13e8b) 131

[timer events](#section_b739d1cceda74019b1e60f70811e2aaf) 131

[timers](#section_c31155253c364dc581ece7affc0c7bbf) 131

Client - ITypeLib

[abstract data model](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140

[initialization](#section_5a327992cce046539187215ed564fe9d) 140

[local events](#section_27433f25698e4a5c883d3660690827c3) 141

[message processing](#section_42f7fe088f65412989924c67d4ecf825) 140

[sequencing rules](#section_42f7fe088f65412989924c67d4ecf825) 140

[timer events](#section_a9a2efbe122e43a387cdb3e4a689ed70) 141

[timers](#section_0c1e84e201c549b082f87ecc19206416) 140

Client - ITypeLib2

[abstract data model](#section_6f07e35d587b432c86776a8047037163) 144

[initialization](#section_7ecf29c130a54e8d88d10c36b637e6d9) 144

[local events](#section_74ca3f6c283f439994eb4611bca35be5) 145

[message processing](#section_e89644ede3704dd28c6a15c78dd1736a) 144

[sequencing rules](#section_e89644ede3704dd28c6a15c78dd1736a) 144

[timer events](#section_b5d711a4c24f4312a1b7375fd570e57d) 145

[timers](#section_ca3e8f7089264fc18e5111b5b59b0f10) 144

[Clone method](#section_94aaabbd5e8941d08acf1dd200b39288) 98

[CLSID\_RecordInfo](#section_58504586e4af44a3be04f1dc281b7429) 18

[Coclass specifications](#section_e86fe771d83647b6b846846de41d592c) 73

[COM server categories](#section_b64411f397674fa3a62faa35d2209cc1) 63

[Common data types](#section_0ccdeadd9fa84c5593fa055d17906ae0) 20

[ComputeHash method](#section_3606b119b1df40a6b93d0d2c288b765b) 77

[ComputeHashDBCS method](#section_c838b52d0e1d44e79ec9cc0954a0bcec) 78

[Connectable servers](#section_4b96268c74984be6b20ed9220eb0057e) 63

[Consistency checks - IDispatch::Invoke](#section_31e8ad0e1d5f4b6094030082eb1a5080) 93

[CreateInstance method](#section_b50cde468de74809ac5e884e3500b93c) 117

[CURRENCY structure](#section_5a2b34c4d109438e9ec884816d8de40d) 38

[CUSTDATA structure](#section_b74500e231534cc6bebf9e11320f7bed) 59

[CUSTDATAITEM structure](#section_02ca19b927cb48efb2ca7f105ba8f475) 59

D

Data model - abstract

[automation client](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95

[automation server](#section_74f390deb6fc42548c322a8730ca38b4) 88

client ([section 3.2.1](#section_d48a8ab1a96b42a1bd835ebbf54288db) 95, [section 3.4.1](#section_253a87eb0f7c405d9524caf1a5cdf357) 99, [section 3.6.1](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105, [section 3.8.1](#section_5f1a181dce344b6d9e5d8602574c8102) 119, [section 3.10.1](#section_8a2130ee60354c6f9e5334559a2c879b) 131, [section 3.12.1](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140, [section 3.14.1](#section_6f07e35d587b432c86776a8047037163) 144)

[IEnumVARIANT client](#section_253a87eb0f7c405d9524caf1a5cdf357) 99

[IEnumVARIANT server](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96

[ITypeComp client](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105

[ITypeComp server](#section_d1644be480344e5f89e87d55c8625b14) 100

[ITypeInfo client](#section_5f1a181dce344b6d9e5d8602574c8102) 119

ITypeInfo server

[common Automation type description elements](#section_59e465b3b60a41948231fdb99b513633) 106

[overview](#section_1e542e10fe4d475f96156b6d956b7073) 105

[TYPEKIND-dependent Automation type description elements](#section_7b1b8bd1a0674edb9d726aa500d035a3) 107

[ITypeInfo2 client](#section_8a2130ee60354c6f9e5334559a2c879b) 131

[ITypeInfo2 server](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120

[ITypeLib client](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140

[ITypeLib server](#section_87fd9a39606742a7b8e613637df3bd0d) 132

[ITypeLib2 client](#section_6f07e35d587b432c86776a8047037163) 144

[ITypeLib2 server](#section_d1f11539021541839b25e1588b061531) 141

server ([section 3.1.1](#section_74f390deb6fc42548c322a8730ca38b4) 88, [section 3.3.1](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96, [section 3.5.1](#section_d1644be480344e5f89e87d55c8625b14) 100, [section 3.7.1](#section_1e542e10fe4d475f96156b6d956b7073) 105, [section 3.9.1](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120, [section 3.11.1](#section_87fd9a39606742a7b8e613637df3bd0d) 132, [section 3.13.1](#section_d1f11539021541839b25e1588b061531) 141)

Data types

[common - overview](#section_0ccdeadd9fa84c5593fa055d17906ae0) 20

[overview](#section_0ccdeadd9fa84c5593fa055d17906ae0) 20

[user-defined](#section_7b86dfb8ca9b437bad8abd9f0aadc266) 40

[DBCS substitution tables](#section_273681debad04d3d9f441da41d0e4f70) 85

[DECIMAL structure](#section_b5493025e447410993a8ac29c48d018d) 39

[Default value - IDispatch::Invoke](#section_be6e35f6932741649bdeffcd0fa0e07d) 94

[DESCKIND enumeration](#section_a7d2404b99344fcca69bd396fb51141a) 36

[DISPID](#section_b0b43e39b0804edda26d7134075c75cd) 50

[DISPID\_NEWENUM](#section_cb9d0131c6bd463d9c407264856a10c5) 50

[DISPID\_PROPERTYPUT](#section_cb9d0131c6bd463d9c407264856a10c5) 50

[DISPID\_UNKNOWN](#section_cb9d0131c6bd463d9c407264856a10c5) 50

[DISPID\_VALUE](#section_cb9d0131c6bd463d9c407264856a10c5) 50

[Dispinterface interfaces](#section_07829751cb564eec88ef476f8a09dd43) 69

[Dispinterfaces automation members](#section_e23aaa6d3ad44886b6520203a1a50c58) 71

[DISPPARAMS structure](#section_144b00dd4c2f4b35a28fc17f591b990c) 51

[Dual interfaces](#section_bd30db0d6c384d549c4467c0e9d25551) 69

[DWORD\_SIZEDARR structure](#section_3d713e4b5e3645469101ca8c1d29bee2) 46

E

[ELEMDESC structure](#section_e14ff3cf034a4884a498fc7586f7160c) 54

Equivalence

[method](#section_a803f0884ef54c2bbaa8351074e9f007) 73

[property](#section_72f7c168506d4d939438c541764bc5ce) 73

Events

local - client ([section 3.2.6](#section_c646b09f59bb4192a970f520157f3fdb) 96, [section 3.4.6](#section_85855ba838aa4387bb2b890f07eb2bcf) 100, [section 3.6.6](#section_0716f9922d224145b911bdc2ec94ab32) 105, [section 3.8.6](#section_de0a7e91e237478fb3789e8b4b05d9f5) 120, [section 3.10.6](#section_a18c66ef338c4aed83e8773b83fa410f) 132, [section 3.12.6](#section_27433f25698e4a5c883d3660690827c3) 141, [section 3.14.6](#section_74ca3f6c283f439994eb4611bca35be5) 145)

local - server ([section 3.1.6](#section_36c98baac2cf40d1934b32d85d9bb4b1) 94, [section 3.3.6](#section_6fb559d8baaf4f91b20deb9ff6163154) 99, [section 3.5.6](#section_22060b45ee544f628ad3e3f9ebabd6d8) 105, [section 3.7.6](#section_c19d274c6db84e1aa673da6c11bdf442) 119, [section 3.9.6](#section_95ee9a50b9bf48ae98075e1fb1accb58) 131, [section 3.11.6](#section_3015d841dfad497a81f5824b0ea21145) 140, [section 3.13.6](#section_ac4278fa645d422aa93c2a62641e8c7a) 144)

timer - client ([section 3.2.5](#section_c02df6e407cc4cc8811f900791eee84e) 95, [section 3.4.5](#section_9d2903537b4b4b1b93a97c2803b02251) 100, [section 3.6.5](#section_1f7d6a20f5b24132952ffb20bf2830d2) 105, [section 3.8.5](#section_83c146e4762c41f39767c4a6e30b6e3f) 119, [section 3.10.5](#section_b739d1cceda74019b1e60f70811e2aaf) 131, [section 3.12.5](#section_a9a2efbe122e43a387cdb3e4a689ed70) 141, [section 3.14.5](#section_b5d711a4c24f4312a1b7375fd570e57d) 145)

timer - server ([section 3.1.5](#section_e9d847bc05194b7da39ba97216389f9a) 94, [section 3.3.5](#section_26fcd84f240140de9c9655d3da1ec8c3) 99, [section 3.5.5](#section_80dff6ef8d374a5e9f83a7aefb5a7a7d) 105, [section 3.7.5](#section_21d1f0d7623e4258a3d055548d2b5f8a) 119, [section 3.9.5](#section_ef4dfe3276ea460c896907a2f0f3a7a1) 131, [section 3.11.5](#section_8f6f08df91ff46258d0dcbc0dc9f7235) 140, [section 3.13.5](#section_17aceae4c14c477aa0147db7d05924d0) 144)

Examples

[aidl-odl method equivalence](#section_7df4503ea14b44ddb81908a381609b9d) 146

[aidl-odl property equivalence](#section_0fd123156d004d8883a40093d78156a4) 146

[calling a method with byref and optional arguments](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[calling method - Byref](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[calling method - optional arguments](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[getting property value](#section_779f34785ffb41239f0d21f6d5a32962) 147

[getting the value of a property](#section_779f34785ffb41239f0d21f6d5a32962) 147

[IEnumVARIANT Clone()](#section_d91fa0c7e55d412ab47875ee41677349) 153

[ienumvariant example](#section_25f0f3a2bbab41ce9a8c2c44408c3223) 150

[IEnumVARIANT Next()](#section_5f3ff188f8d44235808eaf5d86f2a5cb) 151

[IEnumVARIANT overview](#section_25f0f3a2bbab41ce9a8c2c44408c3223) 150

[IEnumVARIANT Reset()](#section_95c696c3e9064e6f8d04832b174ea1e2) 153

[IEnumVARIANT Skip()](#section_2d73fe18f4e343acadba8858e2efc5e8) 152

[invoke argument parameter mapping](#section_db3ea4651cf04399a08564a8c105ca9b) 146

[overview](#section_b5d2939ec2714a58bf8443e528d8f6c6) 146

[reading type information](#section_b35d1c41147b43ada1fbd34a5273019b) 154

[setting property value](#section_3fa975d930e040429a3d2f615e3ca77d) 148

[setting the value of a property](#section_3fa975d930e040429a3d2f615e3ca77d) 148

[EXCEPINFO structure](#section_a7bb989f5c5549c798e124ab2593a9fa) 51

F

[Fields - vendor-extensible](#section_30bded73e36d4491995597ffb541a508) 18

[FindName method](#section_8d41f5777cba48f294a4141372f59a0e) 139

[FLAGGED\_WORD\_BLOB structure](#section_f547135ad76a42c3916f30b4e46d79bc) 37

[Full ABNF](#section_2b2b5513b36a41b7a114e955f80332b3) 181

[Full BNF](#section_2b2b5513b36a41b7a114e955f80332b3) 181

[Full IDL](#section_b37f57b1ac274bdf9cf05b8ecccb65af) 161

[FUNCDESC structure](#section_d3349d25e11d4095ba86de3fda178c4e) 54

[FUNCFLAGS enumeration](#section_be8732b4f8d94e6da946311958d8173f) 29

[FUNCKIND enumeration](#section_a33ebe360f9d4230bcbc466136f45e58) 30

G

GetAllCustData method ([section 3.9.4.11](#section_e255f542ccb94eaea2b72cfdab0636eb) 128, [section 3.13.4.4](#section_16112a9f4f20452b9c018b27752c6e72) 144)

[GetAllFuncCustData method](#section_d11e36add3c14c5f9c55c542cce41c2c) 128

[GetAllImplTypeCustData method](#section_1a93cba3831444389025fc2a61fc23e7) 130

[GetAllParamCustData method](#section_9390aaaa3e254f1480b6ac4cf95ba9a8) 129

[GetAllVarCustData method](#section_2b38e4d7189b4f1fa2d2dd95d561bfe6) 130

[GetContainingTypeLib method](#section_1a843f04e55a47dd85026ba3cd94b161) 118

GetCustData method ([section 3.9.4.5](#section_cd7b30efb5b04d7e976774af8854798d) 124, [section 3.13.4.1](#section_7428511033e740c7840a1caab8381609) 142)

[GetDllEntry method](#section_d82eb39db2184484a1587b582ab65e5c) 115

GetDocumentation method ([section 3.7.4.8](#section_2ea2f705bc334cecbbc7613d6ae0f0c6) 114, [section 3.11.4.7](#section_ceb2d9eb975a47019a793bb9e6ad419b) 137)

GetDocumentation2 method ([section 3.9.4.10](#section_541262a3d8704c8ebe311eb6ab1d9259) 127, [section 3.13.4.3](#section_137253c0736e4616833c527ef2dc1618) 143)

[GetFuncCustData method](#section_0e83075806f54526bd6f75dd4dd3bc7a) 124

[GetFuncDesc method](#section_d54aca0905654fa8b5e4cf87723a89ed) 110

[GetFuncIndexOfMemId method](#section_9b417eae849f460889f060e3ea04d8a6) 122

[GetIDsOfNames method](#section_7166d6ffb8514216bfaa34128274a242) 90

[GetImplTypeCustData method](#section_beed4f3625084de2a16f1a7d01652e23) 126

[GetImplTypeFlags method](#section_4232aaa957a844bfb0d0a02dbdb3e9bc) 114

[GetLibAttr method](#section_d2941c4ae6684722a50ff22610c809d4) 136

[GetLibStatistics method](#section_5ebc64439c0f43e0a5ca54387a97c743) 142

[GetMops method](#section_2dbee2076c704c7bb460c0488a13256d) 118

[GetNames method](#section_78533af1f18640f78b3dc65ba9c6ee3c) 112

[GetParamCustData method](#section_ed33bcd0542243f59b657723a6e10280) 125

[GetRefTypeInfo method](#section_8428b8f844574fab99fe38a8c6ab217b) 117

[GetRefTypeOfImplType method](#section_7225fbad7ad0458bb1492e854364fbfd) 113

[Getting the value of a property example](#section_779f34785ffb41239f0d21f6d5a32962) 147

[GetTypeAttr method](#section_b96292a8c06e4b9c905e129b95697ee4) 110

GetTypeComp method ([section 3.7.4.2](#section_2d345f2a4eb3452bb1f6fca33cfe16e6) 110, [section 3.11.4.6](#section_636b2c755fd643b2bfca5894da8623eb) 137)

[GetTypeFlags method](#section_5e6482f38bb044038dd3fce5c1071c30) 122

GetTypeInfo method ([section 3.1.4.2](#section_d1791851649142898c5725967ef7b9ed) 89, [section 3.11.4.2](#section_ee27cb4791624e5caa667557421a88de) 135)

GetTypeInfoCount method ([section 3.1.4.1](#section_d3233e5b657f4c988a6156449c96fe16) 89, [section 3.11.4.1](#section_430a2456664943df966d9d18a8bc2efa) 134)

[GetTypeInfoOfGuid method](#section_01dd3fef481b4957b540baa469cbc3a7) 136

[GetTypeInfoType method](#section_1d132d304f9a4d51bdbcf1e92920e23d) 135

[GetTypeKind method](#section_6a03300ebd2d45e4b15e2a4c121554e5) 122

[GetVarCustData method](#section_f80e601f5a46432fa8cb8d682b6f0162) 126

[GetVarDesc method](#section_a6b5857a38e446cda2e4bfbc7e21c787) 111

[GetVarIndexOfMemId method](#section_34c839261130446ebe590eba8c32aae1) 123

[Globalization](#section_8ba9784a29b8471aa669e493e5f28326) 76

[Glossary](#section_5583e1b8454c41479f56f72416a15bee) 11

H

[HYPER\_SIZEDARR structure](#section_da9b278c717f4b15835811feb023e17d) 47

I

[IDispatch::GetIDsOfNames (Opnum 5) method](#section_7166d6ffb8514216bfaa34128274a242) 90

[IDispatch::GetTypeInfo (Opnum 4) method](#section_d1791851649142898c5725967ef7b9ed) 89

[IDispatch::GetTypeInfoCount (Opnum 3) method](#section_d3233e5b657f4c988a6156449c96fe16) 89

[IDispatch::Invoke (Opnum 6) method](#section_5c2a199760d7496d8d9aed940bbb82eb) 91

[IDL](#section_b37f57b1ac274bdf9cf05b8ecccb65af) 161

[IDL Automation scope](#section_ae4d27fafaaa4d5fb0f915bfbaaae9b4) 64

[IDL syntax extensions](#section_0c5a99a3dc4b4f7ea109c695b6702b85) 60

IEnumVARIANT client

[abstract data model](#section_253a87eb0f7c405d9524caf1a5cdf357) 99

[initialization](#section_8b92ea3e75824ba6b12fe44b5e739de5) 99

[local events](#section_85855ba838aa4387bb2b890f07eb2bcf) 100

[message processing](#section_a0d58a4ee3b944c28a69d376deb30259) 99

[overview](#section_079100caf62f47028861a1f64e66e9e2) 99

[sequencing rules](#section_a0d58a4ee3b944c28a69d376deb30259) 99

[timer events](#section_9d2903537b4b4b1b93a97c2803b02251) 100

[timers](#section_2ac5018fb5cb4083bbd023c633f64954) 99

[IEnumVARIANT example - overview](#section_25f0f3a2bbab41ce9a8c2c44408c3223) 150

[Ienumvariant example example](#section_25f0f3a2bbab41ce9a8c2c44408c3223) 150

ienumvariant interface ([section 3.3](#section_716d04d1cd1640659b191b8808b3df31) 96, [section 3.4](#section_079100caf62f47028861a1f64e66e9e2) 99)

IEnumVARIANT server

[abstract data model](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96

[initialization](#section_342eaff486b241a297cf9c50d720aed7) 96

[local events](#section_6fb559d8baaf4f91b20deb9ff6163154) 99

[message processing](#section_770adf7fd877444da7aa24b7046ebc83) 96

[overview](#section_716d04d1cd1640659b191b8808b3df31) 96

[sequencing rules](#section_770adf7fd877444da7aa24b7046ebc83) 96

[timer events](#section_26fcd84f240140de9c9655d3da1ec8c3) 99

[timers](#section_733f1d26fb654055a6f2c6de72512f65) 96

[IID\_IDispatch](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_IEnumVARIANT](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_IRecordInfo](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_ITypeComp](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_ITypeInfo](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_ITypeInfo2](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_ITypeLib](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_ITypeLib2](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_IUnknown](#section_58504586e4af44a3be04f1dc281b7429) 18

[IID\_NULL](#section_58504586e4af44a3be04f1dc281b7429) 18

[Implementer - security considerations](#section_fa31cc5ab1f6432b930a055a11b25260) 160

[IMPLTYPEFLAGS enumeration](#section_2743c240436242bf9482a910ac1857f7) 30

[Index of security parameters](#section_1fa7c69b2af74830a0bd79660d81f5e6) 160

[Informative references](#section_af1aa68694bb49b6a8019764009bfd8a) 14

Initialization

[automation client](#section_407ee4e6921d43c3a952632f26a4a9cf) 95

[automation server](#section_bc5d2d2181854ff58fa55d2f7de45df9) 88

client ([section 3.2.3](#section_407ee4e6921d43c3a952632f26a4a9cf) 95, [section 3.4.3](#section_8b92ea3e75824ba6b12fe44b5e739de5) 99, [section 3.6.3](#section_29b2d79f778c480980b5b60ea1bb94f1) 105, [section 3.8.3](#section_3aab2fdab8704f3bbf0bba9eae9ee746) 119, [section 3.10.3](#section_d137d5baaa4549828dfa5c2c75010937) 131, [section 3.12.3](#section_5a327992cce046539187215ed564fe9d) 140, [section 3.14.3](#section_7ecf29c130a54e8d88d10c36b637e6d9) 144)

[IEnumVARIANT client](#section_8b92ea3e75824ba6b12fe44b5e739de5) 99

[IEnumVARIANT server](#section_342eaff486b241a297cf9c50d720aed7) 96

[ITypeComp client](#section_29b2d79f778c480980b5b60ea1bb94f1) 105

[ITypeComp server](#section_d5936f5e9c0d4ab1967ea7b172b824ed) 100

[ITypeInfo client](#section_3aab2fdab8704f3bbf0bba9eae9ee746) 119

[ITypeInfo server](#section_752034952f384211835a7e76c48d14b8) 108

[ITypeInfo2 client](#section_d137d5baaa4549828dfa5c2c75010937) 131

[ITypeInfo2 server](#section_1ed931b7ce4c491eaa1bac5e99c8360b) 120

[ITypeLib client](#section_5a327992cce046539187215ed564fe9d) 140

[ITypeLib server](#section_34d15e8d68494e7d818c2478ff25da06) 133

[ITypeLib2 client](#section_7ecf29c130a54e8d88d10c36b637e6d9) 144

[ITypeLib2 server](#section_ddd3d57998e944e88d10f3706db89934) 141

server ([section 3.1.3](#section_bc5d2d2181854ff58fa55d2f7de45df9) 88, [section 3.3.3](#section_342eaff486b241a297cf9c50d720aed7) 96, [section 3.5.3](#section_d5936f5e9c0d4ab1967ea7b172b824ed) 100, [section 3.7.3](#section_752034952f384211835a7e76c48d14b8) 108, [section 3.9.3](#section_1ed931b7ce4c491eaa1bac5e99c8360b) 120, [section 3.11.3](#section_34d15e8d68494e7d818c2478ff25da06) 133, [section 3.13.3](#section_ddd3d57998e944e88d10f3706db89934) 141)

Interfaces - client

[ienumvariant](#section_079100caf62f47028861a1f64e66e9e2) 99

Interfaces - server

[automation](#section_c2c7dbe2bafa49da93a77b75499ef90a) 88

[ienumvariant](#section_716d04d1cd1640659b191b8808b3df31) 96

[itypecomp](#section_7894019fde1e455eb2aa3b899c2e50f6) 100

[itypeinfo](#section_99504cf916d8401ea87383b85d1ee4aa) 105

[itypeinfo2](#section_2d6024dad2294d78bbb0b9d5bf6459b7) 120

[itypelib](#section_5daecf67bc6e4e17bcf8797bdba1748b) 132

[itypelib2](#section_4bb9bc733cf540a185c7aafaff4874cc) 141

[interfaces automation members](#section_232d5f124b8843e3a63360fc157b1a5f) 69

[Introduction](#section_ed4fb08ccbd24c31b7a8833c010a9d7f) 11

[Invoke argument parameter mapping example](#section_db3ea4651cf04399a08564a8c105ca9b) 146

[Invoke method](#section_5c2a199760d7496d8d9aed940bbb82eb) 91

[INVOKEKIND enumeration](#section_a0d3598da3ee440187fd17a7031b0b9a) 31

[IsName method](#section_70ea09581a204d04b3d3ab4d12446c08) 138

ITypeComp client

[abstract data model](#section_cd46e245f0854fcf9ffd8c6b0f74102d) 105

[initialization](#section_29b2d79f778c480980b5b60ea1bb94f1) 105

[local events](#section_0716f9922d224145b911bdc2ec94ab32) 105

[message processing](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105

[sequencing rules](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105

[timer events](#section_1f7d6a20f5b24132952ffb20bf2830d2) 105

[timers](#section_fc22f1ada5bd452f80289396b26fb480) 105

[itypecomp interface](#section_7894019fde1e455eb2aa3b899c2e50f6) 100

ITypeComp server

[abstract data model](#section_d1644be480344e5f89e87d55c8625b14) 100

[initialization](#section_d5936f5e9c0d4ab1967ea7b172b824ed) 100

[local events](#section_22060b45ee544f628ad3e3f9ebabd6d8) 105

[message processing](#section_1c48719721de4126b487268529217367) 100

[overview](#section_7894019fde1e455eb2aa3b899c2e50f6) 100

[sequencing rules](#section_1c48719721de4126b487268529217367) 100

[timer events](#section_80dff6ef8d374a5e9f83a7aefb5a7a7d) 105

[timers](#section_28c586ba93c14d85b352af58c6ddca0f) 100

[ITypeComp::Bind (Opnum 3) method](#section_476f00da080640d9bbf36059154abbb7) 101

[ITypeComp::BindType (Opnum 4) method](#section_cf61a786b1814267bb6a0987eeb17b38) 104

ITypeInfo client

[abstract data model](#section_5f1a181dce344b6d9e5d8602574c8102) 119

[initialization](#section_3aab2fdab8704f3bbf0bba9eae9ee746) 119

[local events](#section_de0a7e91e237478fb3789e8b4b05d9f5) 120

[message processing](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119

[sequencing rules](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119

[timer events](#section_83c146e4762c41f39767c4a6e30b6e3f) 119

[timers](#section_2379d54297234b55a39099f90c24ab26) 119

[itypeinfo interface](#section_99504cf916d8401ea87383b85d1ee4aa) 105

[ITypeInfo members - types returned with](#section_ceb4299738644689a04e5e77adf7b2b2) 104

ITypeInfo server

abstract data model

[common Automation type description elements](#section_59e465b3b60a41948231fdb99b513633) 106

[overview](#section_1e542e10fe4d475f96156b6d956b7073) 105

[TYPEKIND-dependent Automation type description elements](#section_7b1b8bd1a0674edb9d726aa500d035a3) 107

[initialization](#section_752034952f384211835a7e76c48d14b8) 108

[local events](#section_c19d274c6db84e1aa673da6c11bdf442) 119

[message processing](#section_6ca989bf8b69467d96be9634a30155cb) 108

[overview](#section_99504cf916d8401ea87383b85d1ee4aa) 105

[sequencing rules](#section_6ca989bf8b69467d96be9634a30155cb) 108

[timer events](#section_21d1f0d7623e4258a3d055548d2b5f8a) 119

[timers](#section_415cab158f214f6aa1b761966545fbd2) 108

[ITypeInfo::CreateInstance (Opnum 16) method](#section_b50cde468de74809ac5e884e3500b93c) 117

[ITypeInfo::GetContainingTypeLib (Opnum 18) method](#section_1a843f04e55a47dd85026ba3cd94b161) 118

[ITypeInfo::GetDllEntry (Opnum 13) method](#section_d82eb39db2184484a1587b582ab65e5c) 115

[ITypeInfo::GetDocumentation (Opnum 12) method](#section_2ea2f705bc334cecbbc7613d6ae0f0c6) 114

[ITypeInfo::GetFuncDesc (Opnum 5) method](#section_d54aca0905654fa8b5e4cf87723a89ed) 110

[ITypeInfo::GetImplTypeFlags (Opnum 9) method](#section_4232aaa957a844bfb0d0a02dbdb3e9bc) 114

[ITypeInfo::GetMops (Opnum 17) method](#section_2dbee2076c704c7bb460c0488a13256d) 118

[ITypeInfo::GetNames (Opnum 7) method](#section_78533af1f18640f78b3dc65ba9c6ee3c) 112

[ITypeInfo::GetRefTypeInfo (Opnum 14) method](#section_8428b8f844574fab99fe38a8c6ab217b) 117

[ITypeInfo::GetRefTypeOfImplType (Opnum 8) method](#section_7225fbad7ad0458bb1492e854364fbfd) 113

[ITypeInfo::GetTypeAttr (Opnum 3) method](#section_b96292a8c06e4b9c905e129b95697ee4) 110

[ITypeInfo::GetTypeComp (Opnum 4) method](#section_2d345f2a4eb3452bb1f6fca33cfe16e6) 110

[ITypeInfo::GetVarDesc (Opnum 6) method](#section_a6b5857a38e446cda2e4bfbc7e21c787) 111

ITypeInfo2 client

[abstract data model](#section_8a2130ee60354c6f9e5334559a2c879b) 131

[initialization](#section_d137d5baaa4549828dfa5c2c75010937) 131

[local events](#section_a18c66ef338c4aed83e8773b83fa410f) 132

[message processing](#section_cdfd465a484a4649bc80d2420cd13e8b) 131

[sequencing rules](#section_cdfd465a484a4649bc80d2420cd13e8b) 131

[timer events](#section_b739d1cceda74019b1e60f70811e2aaf) 131

[timers](#section_c31155253c364dc581ece7affc0c7bbf) 131

[itypeinfo2 interface](#section_2d6024dad2294d78bbb0b9d5bf6459b7) 120

ITypeInfo2 server

[abstract data model](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120

[initialization](#section_1ed931b7ce4c491eaa1bac5e99c8360b) 120

[local events](#section_95ee9a50b9bf48ae98075e1fb1accb58) 131

[message processing](#section_dc0a9d195bc34fedb56aba2424379d33) 120

[overview](#section_2d6024dad2294d78bbb0b9d5bf6459b7) 120

[sequencing rules](#section_dc0a9d195bc34fedb56aba2424379d33) 120

[timer events](#section_ef4dfe3276ea460c896907a2f0f3a7a1) 131

[timers](#section_da6ced9ca9a64da3b460421ccbfca511) 120

[ITypeInfo2::GetAllCustData (Opnum 32) method](#section_e255f542ccb94eaea2b72cfdab0636eb) 128

[ITypeInfo2::GetAllFuncCustData (Opnum 33) method](#section_d11e36add3c14c5f9c55c542cce41c2c) 128

[ITypeInfo2::GetAllImplTypeCustData (Opnum 36) method](#section_1a93cba3831444389025fc2a61fc23e7) 130

[ITypeInfo2::GetAllParamCustData (Opnum 34) method](#section_9390aaaa3e254f1480b6ac4cf95ba9a8) 129

[ITypeInfo2::GetAllVarCustData (Opnum 35) method](#section_2b38e4d7189b4f1fa2d2dd95d561bfe6) 130

[ITypeInfo2::GetCustData (Opnum 26) method](#section_cd7b30efb5b04d7e976774af8854798d) 124

[ITypeInfo2::GetDocumentation2 (Opnum 31) method](#section_541262a3d8704c8ebe311eb6ab1d9259) 127

[ITypeInfo2::GetFuncCustData (Opnum 27) method](#section_0e83075806f54526bd6f75dd4dd3bc7a) 124

[ITypeInfo2::GetFuncIndexOfMemId (Opnum 24) method](#section_9b417eae849f460889f060e3ea04d8a6) 122

[ITypeInfo2::GetImplTypeCustData (Opnum 30) method](#section_beed4f3625084de2a16f1a7d01652e23) 126

[ITypeInfo2::GetParamCustData (Opnum 28) method](#section_ed33bcd0542243f59b657723a6e10280) 125

[ITypeInfo2::GetTypeFlags (Opnum 23) method](#section_5e6482f38bb044038dd3fce5c1071c30) 122

[ITypeInfo2::GetTypeKind (Opnum 22) method](#section_6a03300ebd2d45e4b15e2a4c121554e5) 122

[ITypeInfo2::GetVarCustData (Opnum 29) method](#section_f80e601f5a46432fa8cb8d682b6f0162) 126

[ITypeInfo2::GetVarIndexOfMemId (Opnum 25) method](#section_34c839261130446ebe590eba8c32aae1) 123

ITypeLib client

[abstract data model](#section_3a6fc94b8d114844b785c8f5d0f9ad77) 140

[initialization](#section_5a327992cce046539187215ed564fe9d) 140

[local events](#section_27433f25698e4a5c883d3660690827c3) 141

[message processing](#section_42f7fe088f65412989924c67d4ecf825) 140

[sequencing rules](#section_42f7fe088f65412989924c67d4ecf825) 140

[timer events](#section_a9a2efbe122e43a387cdb3e4a689ed70) 141

[timers](#section_0c1e84e201c549b082f87ecc19206416) 140

[itypelib interface](#section_5daecf67bc6e4e17bcf8797bdba1748b) 132

[ITypeLib members - types returned with](#section_b31964b8e3134bc6ad82fe112f394dd0) 103

ITypeLib server

[abstract data model](#section_87fd9a39606742a7b8e613637df3bd0d) 132

[initialization](#section_34d15e8d68494e7d818c2478ff25da06) 133

[local events](#section_3015d841dfad497a81f5824b0ea21145) 140

[message processing](#section_a1436b20e676495ab4f39e9251a40e7b) 133

[overview](#section_5daecf67bc6e4e17bcf8797bdba1748b) 132

[sequencing rules](#section_a1436b20e676495ab4f39e9251a40e7b) 133

[timer events](#section_8f6f08df91ff46258d0dcbc0dc9f7235) 140

[timers](#section_d7d411ad46584b748b47767320641711) 133

[ITypeLib::FindName (Opnum 11) method](#section_8d41f5777cba48f294a4141372f59a0e) 139

[ITypeLib::GetDocumentation (Opnum 9) method](#section_ceb2d9eb975a47019a793bb9e6ad419b) 137

[ITypeLib::GetLibAttr (Opnum 7) method](#section_d2941c4ae6684722a50ff22610c809d4) 136

[ITypeLib::GetTypeComp (Opnum 8) method](#section_636b2c755fd643b2bfca5894da8623eb) 137

[ITypeLib::GetTypeInfo (Opnum 4) method](#section_ee27cb4791624e5caa667557421a88de) 135

[ITypeLib::GetTypeInfoCount (Opnum 3) method](#section_430a2456664943df966d9d18a8bc2efa) 134

[ITypeLib::GetTypeInfoOfGuid (Opnum 6) method](#section_01dd3fef481b4957b540baa469cbc3a7) 136

[ITypeLib::GetTypeInfoType (Opnum 5) method](#section_1d132d304f9a4d51bdbcf1e92920e23d) 135

[ITypeLib::IsName (Opnum 10) method](#section_70ea09581a204d04b3d3ab4d12446c08) 138

ITypeLib2 client

[abstract data model](#section_6f07e35d587b432c86776a8047037163) 144

[initialization](#section_7ecf29c130a54e8d88d10c36b637e6d9) 144

[local events](#section_74ca3f6c283f439994eb4611bca35be5) 145

[message processing](#section_e89644ede3704dd28c6a15c78dd1736a) 144

[sequencing rules](#section_e89644ede3704dd28c6a15c78dd1736a) 144

[timer events](#section_b5d711a4c24f4312a1b7375fd570e57d) 145

[timers](#section_ca3e8f7089264fc18e5111b5b59b0f10) 144

[itypelib2 interface](#section_4bb9bc733cf540a185c7aafaff4874cc) 141

ITypeLib2 server

[abstract data model](#section_d1f11539021541839b25e1588b061531) 141

[initialization](#section_ddd3d57998e944e88d10f3706db89934) 141

[local events](#section_ac4278fa645d422aa93c2a62641e8c7a) 144

[message processing](#section_64099fb5ded14584a82725af7a5f2b80) 141

[sequencing rules](#section_64099fb5ded14584a82725af7a5f2b80) 141

[timer events](#section_17aceae4c14c477aa0147db7d05924d0) 144

[timers](#section_c529cc6cdf3f419299b486012344e6c8) 141

[ITypeLib2::GetAllCustData (Opnum 16) method](#section_16112a9f4f20452b9c018b27752c6e72) 144

[ITypeLib2::GetCustData (Opnum 13) method](#section_7428511033e740c7840a1caab8381609) 142

[ITypeLib2::GetDocumentation2 (Opnum 15) method](#section_137253c0736e4616833c527ef2dc1618) 143

[ITypeLib2::GetLibStatistics (Opnum 14) method](#section_5ebc64439c0f43e0a5ca54387a97c743) 142

L

[LIBFLAGS enumeration](#section_08ca1c9a5ac54630aeafb09d495640b1) 36

Local events

[automation client](#section_c646b09f59bb4192a970f520157f3fdb) 96

[automation server](#section_36c98baac2cf40d1934b32d85d9bb4b1) 94

client ([section 3.2.6](#section_c646b09f59bb4192a970f520157f3fdb) 96, [section 3.4.6](#section_85855ba838aa4387bb2b890f07eb2bcf) 100, [section 3.6.6](#section_0716f9922d224145b911bdc2ec94ab32) 105, [section 3.8.6](#section_de0a7e91e237478fb3789e8b4b05d9f5) 120, [section 3.10.6](#section_a18c66ef338c4aed83e8773b83fa410f) 132, [section 3.12.6](#section_27433f25698e4a5c883d3660690827c3) 141, [section 3.14.6](#section_74ca3f6c283f439994eb4611bca35be5) 145)

[IEnumVARIANT client](#section_85855ba838aa4387bb2b890f07eb2bcf) 100

[IEnumVARIANT server](#section_6fb559d8baaf4f91b20deb9ff6163154) 99

[ITypeComp client](#section_0716f9922d224145b911bdc2ec94ab32) 105

[ITypeComp server](#section_22060b45ee544f628ad3e3f9ebabd6d8) 105

[ITypeInfo client](#section_de0a7e91e237478fb3789e8b4b05d9f5) 120

[ITypeInfo server](#section_c19d274c6db84e1aa673da6c11bdf442) 119

[ITypeInfo2 client](#section_a18c66ef338c4aed83e8773b83fa410f) 132

[ITypeInfo2 server](#section_95ee9a50b9bf48ae98075e1fb1accb58) 131

[ITypeLib client](#section_27433f25698e4a5c883d3660690827c3) 141

[ITypeLib server](#section_3015d841dfad497a81f5824b0ea21145) 140

[ITypeLib2 client](#section_74ca3f6c283f439994eb4611bca35be5) 145

[ITypeLib2 server](#section_ac4278fa645d422aa93c2a62641e8c7a) 144

server ([section 3.1.6](#section_36c98baac2cf40d1934b32d85d9bb4b1) 94, [section 3.3.6](#section_6fb559d8baaf4f91b20deb9ff6163154) 99, [section 3.5.6](#section_22060b45ee544f628ad3e3f9ebabd6d8) 105, [section 3.7.6](#section_c19d274c6db84e1aa673da6c11bdf442) 119, [section 3.9.6](#section_95ee9a50b9bf48ae98075e1fb1accb58) 131, [section 3.11.6](#section_3015d841dfad497a81f5824b0ea21145) 140, [section 3.13.6](#section_ac4278fa645d422aa93c2a62641e8c7a) 144)

[Locale names](#section_382297c6fcb14711924979d31e161545) 81

[LPFUNCDESC](#section_d3349d25e11d4095ba86de3fda178c4e) 54

[LPSAFEARRAYBOUND](#section_1941311d9b7d4a5bb2462d2eaad00f8c) 43

[LPTLIBATTR](#section_b568f4be95e5431bbb3b08dc56e9b224) 58

[LPTYPEATTR](#section_0ca10d0861d2405991097bbaf545715e) 56

[LPVARDESC](#section_ae7791d243994dffb7c6b0d4f3dce982) 55

M

[MapDBChar method](#section_a527d694bd9e4d9daffce1b172612d2d) 81

[MEMBERID\_DEFAULTINST](#section_5fbb485125f645ef9f83e9dd633e1e00) 52

[MEMBERID\_NIL](#section_5fbb485125f645ef9f83e9dd633e1e00) 52

Message processing

[automation client](#section_c4ebe7bc21754fd0af542653820f435c) 95

[automation server](#section_ac9c502bac1c42028ad4048ac98afcc9) 89

client ([section 3.6.4](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105, [section 3.8.4](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119, [section 3.10.4](#section_cdfd465a484a4649bc80d2420cd13e8b) 131, [section 3.12.4](#section_42f7fe088f65412989924c67d4ecf825) 140, [section 3.14.4](#section_e89644ede3704dd28c6a15c78dd1736a) 144)

[IEnumVARIANT client](#section_a0d58a4ee3b944c28a69d376deb30259) 99

[IEnumVARIANT server](#section_770adf7fd877444da7aa24b7046ebc83) 96

[ITypeComp client](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105

[ITypeComp server](#section_1c48719721de4126b487268529217367) 100

[ITypeInfo client](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119

[ITypeInfo server](#section_6ca989bf8b69467d96be9634a30155cb) 108

[ITypeInfo2 client](#section_cdfd465a484a4649bc80d2420cd13e8b) 131

[ITypeInfo2 server](#section_dc0a9d195bc34fedb56aba2424379d33) 120

[ITypeLib client](#section_42f7fe088f65412989924c67d4ecf825) 140

[ITypeLib server](#section_a1436b20e676495ab4f39e9251a40e7b) 133

[ITypeLib2 client](#section_e89644ede3704dd28c6a15c78dd1736a) 144

[ITypeLib2 server](#section_64099fb5ded14584a82725af7a5f2b80) 141

server ([section 3.1.4](#section_ac9c502bac1c42028ad4048ac98afcc9) 89, [section 3.5.4](#section_1c48719721de4126b487268529217367) 100, [section 3.7.4](#section_6ca989bf8b69467d96be9634a30155cb) 108, [section 3.9.4](#section_dc0a9d195bc34fedb56aba2424379d33) 120, [section 3.11.4](#section_a1436b20e676495ab4f39e9251a40e7b) 133, [section 3.13.4](#section_64099fb5ded14584a82725af7a5f2b80) 141)

Messages

[common data types](#section_0ccdeadd9fa84c5593fa055d17906ae0) 20

data types

[common](#section_0ccdeadd9fa84c5593fa055d17906ae0) 20

[user-defined](#section_7b86dfb8ca9b437bad8abd9f0aadc266) 40

[overview](#section_ff283cc08a9542eea42f9934411932cc) 20

[transport](#section_890e4e5a3cbf44e18d1ec6221c6235af) 20

[Method calling example](#section_e93f90dc8afb434fafa994f8564fe19b) 149

[Method equivalence](#section_a803f0884ef54c2bbaa8351074e9f007) 73

Methods

[IDispatch::GetIDsOfNames (Opnum 5)](#section_7166d6ffb8514216bfaa34128274a242) 90

[IDispatch::GetTypeInfo (Opnum 4)](#section_d1791851649142898c5725967ef7b9ed) 89

[IDispatch::GetTypeInfoCount (Opnum 3)](#section_d3233e5b657f4c988a6156449c96fe16) 89

[IDispatch::Invoke (Opnum 6)](#section_5c2a199760d7496d8d9aed940bbb82eb) 91

[ITypeComp::Bind (Opnum 3)](#section_476f00da080640d9bbf36059154abbb7) 101

[ITypeComp::BindType (Opnum 4)](#section_cf61a786b1814267bb6a0987eeb17b38) 104

[ITypeInfo::CreateInstance (Opnum 16)](#section_b50cde468de74809ac5e884e3500b93c) 117

[ITypeInfo::GetContainingTypeLib (Opnum 18)](#section_1a843f04e55a47dd85026ba3cd94b161) 118

[ITypeInfo::GetDllEntry (Opnum 13)](#section_d82eb39db2184484a1587b582ab65e5c) 115

[ITypeInfo::GetDocumentation (Opnum 12)](#section_2ea2f705bc334cecbbc7613d6ae0f0c6) 114

[ITypeInfo::GetFuncDesc (Opnum 5)](#section_d54aca0905654fa8b5e4cf87723a89ed) 110

[ITypeInfo::GetImplTypeFlags (Opnum 9)](#section_4232aaa957a844bfb0d0a02dbdb3e9bc) 114

[ITypeInfo::GetMops (Opnum 17)](#section_2dbee2076c704c7bb460c0488a13256d) 118

[ITypeInfo::GetNames (Opnum 7)](#section_78533af1f18640f78b3dc65ba9c6ee3c) 112

[ITypeInfo::GetRefTypeInfo (Opnum 14)](#section_8428b8f844574fab99fe38a8c6ab217b) 117

[ITypeInfo::GetRefTypeOfImplType (Opnum 8)](#section_7225fbad7ad0458bb1492e854364fbfd) 113

[ITypeInfo::GetTypeAttr (Opnum 3)](#section_b96292a8c06e4b9c905e129b95697ee4) 110

[ITypeInfo::GetTypeComp (Opnum 4)](#section_2d345f2a4eb3452bb1f6fca33cfe16e6) 110

[ITypeInfo::GetVarDesc (Opnum 6)](#section_a6b5857a38e446cda2e4bfbc7e21c787) 111

[ITypeInfo2::GetAllCustData (Opnum 32)](#section_e255f542ccb94eaea2b72cfdab0636eb) 128

[ITypeInfo2::GetAllFuncCustData (Opnum 33)](#section_d11e36add3c14c5f9c55c542cce41c2c) 128

[ITypeInfo2::GetAllImplTypeCustData (Opnum 36)](#section_1a93cba3831444389025fc2a61fc23e7) 130

[ITypeInfo2::GetAllParamCustData (Opnum 34)](#section_9390aaaa3e254f1480b6ac4cf95ba9a8) 129

[ITypeInfo2::GetAllVarCustData (Opnum 35)](#section_2b38e4d7189b4f1fa2d2dd95d561bfe6) 130

[ITypeInfo2::GetCustData (Opnum 26)](#section_cd7b30efb5b04d7e976774af8854798d) 124

[ITypeInfo2::GetDocumentation2 (Opnum 31)](#section_541262a3d8704c8ebe311eb6ab1d9259) 127

[ITypeInfo2::GetFuncCustData (Opnum 27)](#section_0e83075806f54526bd6f75dd4dd3bc7a) 124

[ITypeInfo2::GetFuncIndexOfMemId (Opnum 24)](#section_9b417eae849f460889f060e3ea04d8a6) 122

[ITypeInfo2::GetImplTypeCustData (Opnum 30)](#section_beed4f3625084de2a16f1a7d01652e23) 126

[ITypeInfo2::GetParamCustData (Opnum 28)](#section_ed33bcd0542243f59b657723a6e10280) 125

[ITypeInfo2::GetTypeFlags (Opnum 23)](#section_5e6482f38bb044038dd3fce5c1071c30) 122

[ITypeInfo2::GetTypeKind (Opnum 22)](#section_6a03300ebd2d45e4b15e2a4c121554e5) 122

[ITypeInfo2::GetVarCustData (Opnum 29)](#section_f80e601f5a46432fa8cb8d682b6f0162) 126

[ITypeInfo2::GetVarIndexOfMemId (Opnum 25)](#section_34c839261130446ebe590eba8c32aae1) 123

[ITypeLib::FindName (Opnum 11)](#section_8d41f5777cba48f294a4141372f59a0e) 139

[ITypeLib::GetDocumentation (Opnum 9)](#section_ceb2d9eb975a47019a793bb9e6ad419b) 137

[ITypeLib::GetLibAttr (Opnum 7)](#section_d2941c4ae6684722a50ff22610c809d4) 136

[ITypeLib::GetTypeComp (Opnum 8)](#section_636b2c755fd643b2bfca5894da8623eb) 137

[ITypeLib::GetTypeInfo (Opnum 4)](#section_ee27cb4791624e5caa667557421a88de) 135

[ITypeLib::GetTypeInfoCount (Opnum 3)](#section_430a2456664943df966d9d18a8bc2efa) 134

[ITypeLib::GetTypeInfoOfGuid (Opnum 6)](#section_01dd3fef481b4957b540baa469cbc3a7) 136

[ITypeLib::GetTypeInfoType (Opnum 5)](#section_1d132d304f9a4d51bdbcf1e92920e23d) 135

[ITypeLib::IsName (Opnum 10)](#section_70ea09581a204d04b3d3ab4d12446c08) 138

[ITypeLib2::GetAllCustData (Opnum 16)](#section_16112a9f4f20452b9c018b27752c6e72) 144

[ITypeLib2::GetCustData (Opnum 13)](#section_7428511033e740c7840a1caab8381609) 142

[ITypeLib2::GetDocumentation2 (Opnum 15)](#section_137253c0736e4616833c527ef2dc1618) 143

[ITypeLib2::GetLibStatistics (Opnum 14)](#section_5ebc64439c0f43e0a5ca54387a97c743) 142

[Module specifications](#section_82f9465bae46474e87ffe65e9751affb) 75

N

[Next method](#section_ba5dded9503a4c29884138174de2295d) 97

[Normative references](#section_70bbb865e30c4ab6808bcdedc647935a) 14

O

[ODL dispinterfaces](#section_18d74e75f9a744079fe83406679f7dd8) 72

[Optional arguments - IDispatch::Invoke](#section_be6e35f6932741649bdeffcd0fa0e07d) 94

[Overview](#section_2e27b5335de446e99b3ca6b5bcb161de) 15

[Overview (synopsis)](#section_2e27b5335de446e99b3ca6b5bcb161de) 15

P

[PARAMDESC structure](#section_a965ce8e6c064d7cab302f14b1d8488a) 54

[PARAMDESCEX structure](#section_683c767d2e8e4d2f8804afeb3a73969a) 54

[Parameters - security index](#section_1fa7c69b2af74830a0bd79660d81f5e6) 160

[PARAMFLAGS enumeration](#section_4ca6f07bf89f469bba9e81fda041c8ac) 32

[Preconditions](#section_908b03b1068046029bbfa31ea4f046fd) 18

[Prerequisites](#section_908b03b1068046029bbfa31ea4f046fd) 18

[Primary lookup tables](#section_0dd943a0c4c04140abb9e6e6e45f307b) 81

[Product behavior](#section_ae19ad637433456888e9f70e5593547d) 176

[Property equivalence](#section_72f7c168506d4d939438c541764bc5ce) 73

Property value example

[getting](#section_779f34785ffb41239f0d21f6d5a32962) 147

[setting](#section_3fa975d930e040429a3d2f615e3ca77d) 148

[Protocol Details](#section_3bca16bbefa6434f8f094bcc9487fd11) 88

[overview](#section_3bca16bbefa6434f8f094bcc9487fd11) 88

R

[Reading type information example](#section_b35d1c41147b43ada1fbd34a5273019b) 154

Reading type information examples

[binding to a member of a default nonsource interface of an appobject coclass](#section_eb7ee58a5c514d828f23159b6782847c) 158

[binding to a member of a partner interface](#section_f9f604d0e04942799c4ce2538d64197b) 158

[enumerating all methods in an interface](#section_681c87bfcffc4b188dc9c23b31877880) 156

[enumerating all nonsource interfaces in a coclass](#section_6cce2d07a33642d7a750e01c62881755) 155

[enumerating on all enumerations in a typelibrary](#section_3572898a3de749a9a2b4890c4a4001ed) 155

[enumerating on all types in a typelibrary](#section_8f2deb66b3d54ed6971f5b99323f256f) 155

[getting ITypeLib implementations from automation server](#section_fea1146ad7d24c1aa1cd2de196fcbc9a) 154

[overview](#section_b35d1c41147b43ada1fbd34a5273019b) 154

[retrieving type information](#section_2bf2080b5f264074a3d3ff2927b85fa0) 157

[RecordInfo structure](#section_deb939dfef4d49c384677265669e89ed) 49

[References](#section_8c3bf1771fd74c42a1a91c1dd35bff80) 14

[informative](#section_af1aa68694bb49b6a8019764009bfd8a) 14

[normative](#section_70bbb865e30c4ab6808bcdedc647935a) 14

[Referencing external types](#section_ca898dc2450e4e2e983e1eb1f0a68a45) 76

[Relationship to other protocols](#section_29c26bdcafd24330903d167d67897e75) 18

[Reset method](#section_fc80414cad314b0ebac8c874ccc11725) 98

S

[SAFEARR\_BRECORD structure](#section_ed0bd7042797407a84b5902a2a28c315) 45

[SAFEARR\_BSTR structure](#section_ceb8cfd549f14b33974dae8f3e767150) 44

[SAFEARR\_DISPATCH structure](#section_d3ed62ad68fb4783987821d290bf5a45) 44

[SAFEARR\_HAVEIID structure](#section_78bd655922e24d0da4873fb521118e0b) 45

[SAFEARR\_UNKNOWN structure](#section_32cf502df3d54fe68894c8c12228931c) 44

[SAFEARR\_VARIANT structure](#section_247d19b18ced4f82a394a74e00a41e4d) 45

[SAFEARRAY](#section_04e72b3f573145089bb4de29fbd0f781) 43

[SAFEARRAY structure](#section_2e87a537930541c6a88bb79809b3703a) 47

[SAFEARRAYBOUND structure](#section_1941311d9b7d4a5bb2462d2eaad00f8c) 43

[Scalar-sized arrays](#section_2a7674d26c7746fea6a4ca614c5133df) 46

Security

[implementer considerations](#section_fa31cc5ab1f6432b930a055a11b25260) 160

[overview](#section_8065800baf92469183c72a33eb3b08ee) 160

[parameter index](#section_1fa7c69b2af74830a0bd79660d81f5e6) 160

Sequencing rules

[automation client](#section_c4ebe7bc21754fd0af542653820f435c) 95

[automation server](#section_ac9c502bac1c42028ad4048ac98afcc9) 89

client ([section 3.6.4](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105, [section 3.8.4](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119, [section 3.10.4](#section_cdfd465a484a4649bc80d2420cd13e8b) 131, [section 3.12.4](#section_42f7fe088f65412989924c67d4ecf825) 140, [section 3.14.4](#section_e89644ede3704dd28c6a15c78dd1736a) 144)

[IEnumVARIANT client](#section_a0d58a4ee3b944c28a69d376deb30259) 99

[IEnumVARIANT server](#section_770adf7fd877444da7aa24b7046ebc83) 96

[ITypeComp client](#section_e0b414221b0a42f5b17ca8ffd18433a4) 105

[ITypeComp server](#section_1c48719721de4126b487268529217367) 100

[ITypeInfo client](#section_c7ce3c67c5604ceea55d3e9687f5f44e) 119

[ITypeInfo server](#section_6ca989bf8b69467d96be9634a30155cb) 108

[ITypeInfo2 client](#section_cdfd465a484a4649bc80d2420cd13e8b) 131

[ITypeInfo2 server](#section_dc0a9d195bc34fedb56aba2424379d33) 120

[ITypeLib client](#section_42f7fe088f65412989924c67d4ecf825) 140

[ITypeLib server](#section_a1436b20e676495ab4f39e9251a40e7b) 133

[ITypeLib2 client](#section_e89644ede3704dd28c6a15c78dd1736a) 144

[ITypeLib2 server](#section_64099fb5ded14584a82725af7a5f2b80) 141

server ([section 3.1.4](#section_ac9c502bac1c42028ad4048ac98afcc9) 89, [section 3.5.4](#section_1c48719721de4126b487268529217367) 100, [section 3.7.4](#section_6ca989bf8b69467d96be9634a30155cb) 108, [section 3.9.4](#section_dc0a9d195bc34fedb56aba2424379d33) 120, [section 3.11.4](#section_a1436b20e676495ab4f39e9251a40e7b) 133, [section 3.13.4](#section_64099fb5ded14584a82725af7a5f2b80) 141)

Server

abstract data model ([section 3.1.1](#section_74f390deb6fc42548c322a8730ca38b4) 88, [section 3.3.1](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96, [section 3.5.1](#section_d1644be480344e5f89e87d55c8625b14) 100, [section 3.7.1](#section_1e542e10fe4d475f96156b6d956b7073) 105, [section 3.9.1](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120, [section 3.11.1](#section_87fd9a39606742a7b8e613637df3bd0d) 132, [section 3.13.1](#section_d1f11539021541839b25e1588b061531) 141)

[automation interface](#section_c2c7dbe2bafa49da93a77b75499ef90a) 88

[IDispatch::GetIDsOfNames (Opnum 5) method](#section_7166d6ffb8514216bfaa34128274a242) 90

[IDispatch::GetTypeInfo (Opnum 4) method](#section_d1791851649142898c5725967ef7b9ed) 89

[IDispatch::GetTypeInfoCount (Opnum 3) method](#section_d3233e5b657f4c988a6156449c96fe16) 89

[IDispatch::Invoke (Opnum 6) method](#section_5c2a199760d7496d8d9aed940bbb82eb) 91

[ienumvariant interface](#section_716d04d1cd1640659b191b8808b3df31) 96

initialization ([section 3.1.3](#section_bc5d2d2181854ff58fa55d2f7de45df9) 88, [section 3.3.3](#section_342eaff486b241a297cf9c50d720aed7) 96, [section 3.5.3](#section_d5936f5e9c0d4ab1967ea7b172b824ed) 100, [section 3.7.3](#section_752034952f384211835a7e76c48d14b8) 108, [section 3.9.3](#section_1ed931b7ce4c491eaa1bac5e99c8360b) 120, [section 3.11.3](#section_34d15e8d68494e7d818c2478ff25da06) 133, [section 3.13.3](#section_ddd3d57998e944e88d10f3706db89934) 141)

[itypecomp interface](#section_7894019fde1e455eb2aa3b899c2e50f6) 100

[ITypeComp::Bind (Opnum 3) method](#section_476f00da080640d9bbf36059154abbb7) 101

[ITypeComp::BindType (Opnum 4) method](#section_cf61a786b1814267bb6a0987eeb17b38) 104

[itypeinfo interface](#section_99504cf916d8401ea87383b85d1ee4aa) 105

[ITypeInfo::CreateInstance (Opnum 16) method](#section_b50cde468de74809ac5e884e3500b93c) 117

[ITypeInfo::GetContainingTypeLib (Opnum 18) method](#section_1a843f04e55a47dd85026ba3cd94b161) 118

[ITypeInfo::GetDllEntry (Opnum 13) method](#section_d82eb39db2184484a1587b582ab65e5c) 115

[ITypeInfo::GetDocumentation (Opnum 12) method](#section_2ea2f705bc334cecbbc7613d6ae0f0c6) 114

[ITypeInfo::GetFuncDesc (Opnum 5) method](#section_d54aca0905654fa8b5e4cf87723a89ed) 110

[ITypeInfo::GetImplTypeFlags (Opnum 9) method](#section_4232aaa957a844bfb0d0a02dbdb3e9bc) 114

[ITypeInfo::GetMops (Opnum 17) method](#section_2dbee2076c704c7bb460c0488a13256d) 118

[ITypeInfo::GetNames (Opnum 7) method](#section_78533af1f18640f78b3dc65ba9c6ee3c) 112

[ITypeInfo::GetRefTypeInfo (Opnum 14) method](#section_8428b8f844574fab99fe38a8c6ab217b) 117

[ITypeInfo::GetRefTypeOfImplType (Opnum 8) method](#section_7225fbad7ad0458bb1492e854364fbfd) 113

[ITypeInfo::GetTypeAttr (Opnum 3) method](#section_b96292a8c06e4b9c905e129b95697ee4) 110

[ITypeInfo::GetTypeComp (Opnum 4) method](#section_2d345f2a4eb3452bb1f6fca33cfe16e6) 110

[ITypeInfo::GetVarDesc (Opnum 6) method](#section_a6b5857a38e446cda2e4bfbc7e21c787) 111

[itypeinfo2 interface](#section_2d6024dad2294d78bbb0b9d5bf6459b7) 120

[ITypeInfo2::GetAllCustData (Opnum 32) method](#section_e255f542ccb94eaea2b72cfdab0636eb) 128

[ITypeInfo2::GetAllFuncCustData (Opnum 33) method](#section_d11e36add3c14c5f9c55c542cce41c2c) 128

[ITypeInfo2::GetAllImplTypeCustData (Opnum 36) method](#section_1a93cba3831444389025fc2a61fc23e7) 130

[ITypeInfo2::GetAllParamCustData (Opnum 34) method](#section_9390aaaa3e254f1480b6ac4cf95ba9a8) 129

[ITypeInfo2::GetAllVarCustData (Opnum 35) method](#section_2b38e4d7189b4f1fa2d2dd95d561bfe6) 130

[ITypeInfo2::GetCustData (Opnum 26) method](#section_cd7b30efb5b04d7e976774af8854798d) 124

[ITypeInfo2::GetDocumentation2 (Opnum 31) method](#section_541262a3d8704c8ebe311eb6ab1d9259) 127

[ITypeInfo2::GetFuncCustData (Opnum 27) method](#section_0e83075806f54526bd6f75dd4dd3bc7a) 124

[ITypeInfo2::GetFuncIndexOfMemId (Opnum 24) method](#section_9b417eae849f460889f060e3ea04d8a6) 122

[ITypeInfo2::GetImplTypeCustData (Opnum 30) method](#section_beed4f3625084de2a16f1a7d01652e23) 126

[ITypeInfo2::GetParamCustData (Opnum 28) method](#section_ed33bcd0542243f59b657723a6e10280) 125

[ITypeInfo2::GetTypeFlags (Opnum 23) method](#section_5e6482f38bb044038dd3fce5c1071c30) 122

[ITypeInfo2::GetTypeKind (Opnum 22) method](#section_6a03300ebd2d45e4b15e2a4c121554e5) 122

[ITypeInfo2::GetVarCustData (Opnum 29) method](#section_f80e601f5a46432fa8cb8d682b6f0162) 126

[ITypeInfo2::GetVarIndexOfMemId (Opnum 25) method](#section_34c839261130446ebe590eba8c32aae1) 123

[itypelib interface](#section_5daecf67bc6e4e17bcf8797bdba1748b) 132

[ITypeLib::FindName (Opnum 11) method](#section_8d41f5777cba48f294a4141372f59a0e) 139

[ITypeLib::GetDocumentation (Opnum 9) method](#section_ceb2d9eb975a47019a793bb9e6ad419b) 137

[ITypeLib::GetLibAttr (Opnum 7) method](#section_d2941c4ae6684722a50ff22610c809d4) 136

[ITypeLib::GetTypeComp (Opnum 8) method](#section_636b2c755fd643b2bfca5894da8623eb) 137

[ITypeLib::GetTypeInfo (Opnum 4) method](#section_ee27cb4791624e5caa667557421a88de) 135

[ITypeLib::GetTypeInfoCount (Opnum 3) method](#section_430a2456664943df966d9d18a8bc2efa) 134

[ITypeLib::GetTypeInfoOfGuid (Opnum 6) method](#section_01dd3fef481b4957b540baa469cbc3a7) 136

[ITypeLib::GetTypeInfoType (Opnum 5) method](#section_1d132d304f9a4d51bdbcf1e92920e23d) 135

[ITypeLib::IsName (Opnum 10) method](#section_70ea09581a204d04b3d3ab4d12446c08) 138

[itypelib2 interface](#section_4bb9bc733cf540a185c7aafaff4874cc) 141

[ITypeLib2::GetAllCustData (Opnum 16) method](#section_16112a9f4f20452b9c018b27752c6e72) 144

[ITypeLib2::GetCustData (Opnum 13) method](#section_7428511033e740c7840a1caab8381609) 142

[ITypeLib2::GetDocumentation2 (Opnum 15) method](#section_137253c0736e4616833c527ef2dc1618) 143

[ITypeLib2::GetLibStatistics (Opnum 14) method](#section_5ebc64439c0f43e0a5ca54387a97c743) 142

local events ([section 3.1.6](#section_36c98baac2cf40d1934b32d85d9bb4b1) 94, [section 3.3.6](#section_6fb559d8baaf4f91b20deb9ff6163154) 99, [section 3.5.6](#section_22060b45ee544f628ad3e3f9ebabd6d8) 105, [section 3.7.6](#section_c19d274c6db84e1aa673da6c11bdf442) 119, [section 3.9.6](#section_95ee9a50b9bf48ae98075e1fb1accb58) 131, [section 3.11.6](#section_3015d841dfad497a81f5824b0ea21145) 140, [section 3.13.6](#section_ac4278fa645d422aa93c2a62641e8c7a) 144)

message processing ([section 3.1.4](#section_ac9c502bac1c42028ad4048ac98afcc9) 89, [section 3.5.4](#section_1c48719721de4126b487268529217367) 100, [section 3.7.4](#section_6ca989bf8b69467d96be9634a30155cb) 108, [section 3.9.4](#section_dc0a9d195bc34fedb56aba2424379d33) 120, [section 3.11.4](#section_a1436b20e676495ab4f39e9251a40e7b) 133, [section 3.13.4](#section_64099fb5ded14584a82725af7a5f2b80) 141)

overview ([section 3.1](#section_c2c7dbe2bafa49da93a77b75499ef90a) 88, [section 3.3](#section_716d04d1cd1640659b191b8808b3df31) 96, [section 3.5](#section_7894019fde1e455eb2aa3b899c2e50f6) 100, [section 3.7](#section_99504cf916d8401ea87383b85d1ee4aa) 105, [section 3.9](#section_2d6024dad2294d78bbb0b9d5bf6459b7) 120, [section 3.11](#section_5daecf67bc6e4e17bcf8797bdba1748b) 132, [section 3.13](#section_4bb9bc733cf540a185c7aafaff4874cc) 141)

sequencing rules ([section 3.1.4](#section_ac9c502bac1c42028ad4048ac98afcc9) 89, [section 3.5.4](#section_1c48719721de4126b487268529217367) 100, [section 3.7.4](#section_6ca989bf8b69467d96be9634a30155cb) 108, [section 3.9.4](#section_dc0a9d195bc34fedb56aba2424379d33) 120, [section 3.11.4](#section_a1436b20e676495ab4f39e9251a40e7b) 133, [section 3.13.4](#section_64099fb5ded14584a82725af7a5f2b80) 141)

timer events ([section 3.1.5](#section_e9d847bc05194b7da39ba97216389f9a) 94, [section 3.3.5](#section_26fcd84f240140de9c9655d3da1ec8c3) 99, [section 3.5.5](#section_80dff6ef8d374a5e9f83a7aefb5a7a7d) 105, [section 3.7.5](#section_21d1f0d7623e4258a3d055548d2b5f8a) 119, [section 3.9.5](#section_ef4dfe3276ea460c896907a2f0f3a7a1) 131, [section 3.11.5](#section_8f6f08df91ff46258d0dcbc0dc9f7235) 140, [section 3.13.5](#section_17aceae4c14c477aa0147db7d05924d0) 144)

timers ([section 3.1.2](#section_39dcc7d407784979909de7434a3455c9) 88, [section 3.3.2](#section_733f1d26fb654055a6f2c6de72512f65) 96, [section 3.5.2](#section_28c586ba93c14d85b352af58c6ddca0f) 100, [section 3.7.2](#section_415cab158f214f6aa1b761966545fbd2) 108, [section 3.9.2](#section_da6ced9ca9a64da3b460421ccbfca511) 120, [section 3.11.2](#section_d7d411ad46584b748b47767320641711) 133, [section 3.13.2](#section_c529cc6cdf3f419299b486012344e6c8) 141)

Server - automation

[abstract data model](#section_74f390deb6fc42548c322a8730ca38b4) 88

[initialization](#section_bc5d2d2181854ff58fa55d2f7de45df9) 88

[local events](#section_36c98baac2cf40d1934b32d85d9bb4b1) 94

[message processing](#section_ac9c502bac1c42028ad4048ac98afcc9) 89

[overview](#section_c2c7dbe2bafa49da93a77b75499ef90a) 88

[sequencing rules](#section_ac9c502bac1c42028ad4048ac98afcc9) 89

[timer events](#section_e9d847bc05194b7da39ba97216389f9a) 94

[timers](#section_39dcc7d407784979909de7434a3455c9) 88

Server - IEnumVARIANT

[abstract data model](#section_4bc7ca2a61c54fea9d157ec8473350a0) 96

[initialization](#section_342eaff486b241a297cf9c50d720aed7) 96

[local events](#section_6fb559d8baaf4f91b20deb9ff6163154) 99

[message processing](#section_770adf7fd877444da7aa24b7046ebc83) 96

[overview](#section_716d04d1cd1640659b191b8808b3df31) 96

[sequencing rules](#section_770adf7fd877444da7aa24b7046ebc83) 96

[timer events](#section_26fcd84f240140de9c9655d3da1ec8c3) 99

[timers](#section_733f1d26fb654055a6f2c6de72512f65) 96

Server - ITypeComp

[abstract data model](#section_d1644be480344e5f89e87d55c8625b14) 100

[initialization](#section_d5936f5e9c0d4ab1967ea7b172b824ed) 100

[local events](#section_22060b45ee544f628ad3e3f9ebabd6d8) 105

[message processing](#section_1c48719721de4126b487268529217367) 100

[overview](#section_7894019fde1e455eb2aa3b899c2e50f6) 100

[sequencing rules](#section_1c48719721de4126b487268529217367) 100

[timer events](#section_80dff6ef8d374a5e9f83a7aefb5a7a7d) 105

[timers](#section_28c586ba93c14d85b352af58c6ddca0f) 100

Server - ITypeInfo

abstract data model

[common Automation type description elements](#section_59e465b3b60a41948231fdb99b513633) 106

[overview](#section_1e542e10fe4d475f96156b6d956b7073) 105

[TYPEKIND-dependent Automation type description elements](#section_7b1b8bd1a0674edb9d726aa500d035a3) 107

[initialization](#section_752034952f384211835a7e76c48d14b8) 108

[local events](#section_c19d274c6db84e1aa673da6c11bdf442) 119

[message processing](#section_6ca989bf8b69467d96be9634a30155cb) 108

[overview](#section_99504cf916d8401ea87383b85d1ee4aa) 105

[sequencing rules](#section_6ca989bf8b69467d96be9634a30155cb) 108

[timer events](#section_21d1f0d7623e4258a3d055548d2b5f8a) 119

[timers](#section_415cab158f214f6aa1b761966545fbd2) 108

Server - ITypeInfo2

[abstract data model](#section_e95ba4d60c504d60ba489d4152f7bfe0) 120

[initialization](#section_1ed931b7ce4c491eaa1bac5e99c8360b) 120

[local events](#section_95ee9a50b9bf48ae98075e1fb1accb58) 131

[message processing](#section_dc0a9d195bc34fedb56aba2424379d33) 120

[overview](#section_2d6024dad2294d78bbb0b9d5bf6459b7) 120

[sequencing rules](#section_dc0a9d195bc34fedb56aba2424379d33) 120

[timer events](#section_ef4dfe3276ea460c896907a2f0f3a7a1) 131

[timers](#section_da6ced9ca9a64da3b460421ccbfca511) 120

Server - ITypeLib

[abstract data model](#section_87fd9a39606742a7b8e613637df3bd0d) 132

[initialization](#section_34d15e8d68494e7d818c2478ff25da06) 133

[local events](#section_3015d841dfad497a81f5824b0ea21145) 140

[message processing](#section_a1436b20e676495ab4f39e9251a40e7b) 133

[overview](#section_5daecf67bc6e4e17bcf8797bdba1748b) 132

[sequencing rules](#section_a1436b20e676495ab4f39e9251a40e7b) 133

[timer events](#section_8f6f08df91ff46258d0dcbc0dc9f7235) 140

[timers](#section_d7d411ad46584b748b47767320641711) 133

Server - ITypeLib2

[abstract data model](#section_d1f11539021541839b25e1588b061531) 141

[initialization](#section_ddd3d57998e944e88d10f3706db89934) 141

[local events](#section_ac4278fa645d422aa93c2a62641e8c7a) 144

[message processing](#section_64099fb5ded14584a82725af7a5f2b80) 141

[sequencing rules](#section_64099fb5ded14584a82725af7a5f2b80) 141

[timer events](#section_17aceae4c14c477aa0147db7d05924d0) 144

[timers](#section_c529cc6cdf3f419299b486012344e6c8) 141

[Setting the value of a property example](#section_3fa975d930e040429a3d2f615e3ca77d) 148

[SF\_TYPE enumeration](#section_8c78fede6f6c4822a5f80fcbbc8d8242) 26

[Skip method](#section_cc090461ef064bef9b4cc51a11b29aea) 98

[Standards assignments](#section_58504586e4af44a3be04f1dc281b7429) 18

[String equivalence](#section_88ad8ff6717b4fcf89c5a9f48c052306) 76

[String handling](#section_ef05bc9b062d467bad880f19e3e545f6) 76

[SYSKIND enumeration](#section_0d81289ef0ef474d8e61dedae9ea5a08) 36

T

Timer events

[automation client](#section_c02df6e407cc4cc8811f900791eee84e) 95

[automation server](#section_e9d847bc05194b7da39ba97216389f9a) 94

client ([section 3.2.5](#section_c02df6e407cc4cc8811f900791eee84e) 95, [section 3.4.5](#section_9d2903537b4b4b1b93a97c2803b02251) 100, [section 3.6.5](#section_1f7d6a20f5b24132952ffb20bf2830d2) 105, [section 3.8.5](#section_83c146e4762c41f39767c4a6e30b6e3f) 119, [section 3.10.5](#section_b739d1cceda74019b1e60f70811e2aaf) 131, [section 3.12.5](#section_a9a2efbe122e43a387cdb3e4a689ed70) 141, [section 3.14.5](#section_b5d711a4c24f4312a1b7375fd570e57d) 145)

[IEnumVARIANT client](#section_9d2903537b4b4b1b93a97c2803b02251) 100

[IEnumVARIANT server](#section_26fcd84f240140de9c9655d3da1ec8c3) 99

[ITypeComp client](#section_1f7d6a20f5b24132952ffb20bf2830d2) 105

[ITypeComp server](#section_80dff6ef8d374a5e9f83a7aefb5a7a7d) 105

[ITypeInfo client](#section_83c146e4762c41f39767c4a6e30b6e3f) 119

[ITypeInfo server](#section_21d1f0d7623e4258a3d055548d2b5f8a) 119

[ITypeInfo2 client](#section_b739d1cceda74019b1e60f70811e2aaf) 131

[ITypeInfo2 server](#section_ef4dfe3276ea460c896907a2f0f3a7a1) 131

[ITypeLib client](#section_a9a2efbe122e43a387cdb3e4a689ed70) 141

[ITypeLib server](#section_8f6f08df91ff46258d0dcbc0dc9f7235) 140

[ITypeLib2 client](#section_b5d711a4c24f4312a1b7375fd570e57d) 145

[ITypeLib2 server](#section_17aceae4c14c477aa0147db7d05924d0) 144

server ([section 3.1.5](#section_e9d847bc05194b7da39ba97216389f9a) 94, [section 3.3.5](#section_26fcd84f240140de9c9655d3da1ec8c3) 99, [section 3.5.5](#section_80dff6ef8d374a5e9f83a7aefb5a7a7d) 105, [section 3.7.5](#section_21d1f0d7623e4258a3d055548d2b5f8a) 119, [section 3.9.5](#section_ef4dfe3276ea460c896907a2f0f3a7a1) 131, [section 3.11.5](#section_8f6f08df91ff46258d0dcbc0dc9f7235) 140, [section 3.13.5](#section_17aceae4c14c477aa0147db7d05924d0) 144)

Timers

[automation client](#section_006bdf07d2de4ebf8bbdad3dfc3b2fcd) 95

[automation server](#section_39dcc7d407784979909de7434a3455c9) 88

client ([section 3.2.2](#section_006bdf07d2de4ebf8bbdad3dfc3b2fcd) 95, [section 3.4.2](#section_2ac5018fb5cb4083bbd023c633f64954) 99, [section 3.6.2](#section_fc22f1ada5bd452f80289396b26fb480) 105, [section 3.8.2](#section_2379d54297234b55a39099f90c24ab26) 119, [section 3.10.2](#section_c31155253c364dc581ece7affc0c7bbf) 131, [section 3.12.2](#section_0c1e84e201c549b082f87ecc19206416) 140, [section 3.14.2](#section_ca3e8f7089264fc18e5111b5b59b0f10) 144)

[IEnumVARIANT client](#section_2ac5018fb5cb4083bbd023c633f64954) 99

[IEnumVARIANT server](#section_733f1d26fb654055a6f2c6de72512f65) 96

[ITypeComp client](#section_fc22f1ada5bd452f80289396b26fb480) 105

[ITypeComp server](#section_28c586ba93c14d85b352af58c6ddca0f) 100

[ITypeInfo client](#section_2379d54297234b55a39099f90c24ab26) 119

[ITypeInfo server](#section_415cab158f214f6aa1b761966545fbd2) 108

[ITypeInfo2 client](#section_c31155253c364dc581ece7affc0c7bbf) 131

[ITypeInfo2 server](#section_da6ced9ca9a64da3b460421ccbfca511) 120

[ITypeLib client](#section_0c1e84e201c549b082f87ecc19206416) 140

[ITypeLib server](#section_d7d411ad46584b748b47767320641711) 133

[ITypeLib2 client](#section_ca3e8f7089264fc18e5111b5b59b0f10) 144

[ITypeLib2 server](#section_c529cc6cdf3f419299b486012344e6c8) 141

server ([section 3.1.2](#section_39dcc7d407784979909de7434a3455c9) 88, [section 3.3.2](#section_733f1d26fb654055a6f2c6de72512f65) 96, [section 3.5.2](#section_28c586ba93c14d85b352af58c6ddca0f) 100, [section 3.7.2](#section_415cab158f214f6aa1b761966545fbd2) 108, [section 3.9.2](#section_da6ced9ca9a64da3b460421ccbfca511) 120, [section 3.11.2](#section_d7d411ad46584b748b47767320641711) 133, [section 3.13.2](#section_c529cc6cdf3f419299b486012344e6c8) 141)

[TLIBATTR structure](#section_b568f4be95e5431bbb3b08dc56e9b224) 58

[Tracking changes](#section_d1f84f6f256b4836aaee029e939b4ebf) 187

[Transport](#section_890e4e5a3cbf44e18d1ec6221c6235af) 20

[Transport - message](#section_890e4e5a3cbf44e18d1ec6221c6235af) 20

[TYPEATTR structure](#section_0ca10d0861d2405991097bbaf545715e) 56

[TYPEDESC structure](#section_95bb92a7f783477facbcc947d754fa8b) 53

[TYPEFLAGS enumeration](#section_155c66e2ffe14f18b849f827ca989aa7) 32

[TYPEKIND enumeration](#section_78ccbd1cd8ff43019afcdf562372fb33) 33

Types returned with bound elements - ITypeComp::Bind

[ITypeInfo members](#section_ceb4299738644689a04e5e77adf7b2b2) 104

[ITypeLib members](#section_b31964b8e3134bc6ad82fe112f394dd0) 103

[overview](#section_3ca3ffe31f9547159b54574a300ffd88) 103

U

[User-defined data types](#section_7b86dfb8ca9b437bad8abd9f0aadc266) 40

V

[VARDESC structure](#section_ae7791d243994dffb7c6b0d4f3dce982) 55

[VARENUM enumeration](#section_3fe7db9f58034dc49d145425d3f5461f) 21

[VARFLAGS enumeration](#section_8ec5cfa4e710446aab896715dece4aec) 34

[VARIANT](#section_b2ee2b50665e43e6a92c8f2a29fd7add) 41

[VARKIND enumeration](#section_a0e9d46351a249cc8935a65c9338d3df) 35

[Vendor-extensible fields](#section_30bded73e36d4491995597ffb541a508) 18

[Versioning](#section_aa944ebae686410f9b2c128ec9fd5fc1) 18

W

[wireBRECORDStr structure](#section_d9237563093e4bc9b8244c306bfc19e3) 40

[wireVARIANTStr structure](#section_4e2e9bff2ac54bab83081806b256833e) 41

[WORD\_SIZEDARR structure](#section_1535410dab5b40e4817753162c92b031) 46