

## INTERFACE SPECIFICATION

### winIDEA automation server reference

## Introduction

This document document in detail the winIDEA automation server interface.

# Table of contents

|   |    |
|---|----|
| <a href="#">Introduction</a>                | 1  |
| <a href="#">Table of contents</a>           | 2  |
| <a href="#">Group: Connect</a>              | 11 |
| <a href="#">HRESULT Connect()</a>           | 12 |
| <a href="#">HRESULT ScriptConnect()</a>     | 12 |
| <a href="#">HRESULT ConnectEx(</a>          |    |
| <a href="#">[in] BSTR strAddress</a>        |    |
| <a href="#">[in] short iPort</a>            |    |
| <a href="#">[in] BSTR strLibPath</a>        |    |
| <a href="#">[in] long iLaunchFlags)</a>     | 13 |
| <a href="#">HRESULT ScriptConnectEx(</a>    |    |
| <a href="#">[in] BSTR strAddress</a>        |    |
| <a href="#">[in] short iPort</a>            |    |
| <a href="#">[in] BSTR strLibPath</a>        |    |
| <a href="#">[in] long iLaunchFlags)</a>     | 13 |
| <a href="#">strAddress</a>                  | 13 |
| <a href="#">iPort</a>                       | 13 |
| <a href="#">strLibPath</a>                  | 13 |
| <a href="#">iLaunchFlags</a>                | 13 |
| <a href="#">HRESULT ConnectEx2(</a>         |    |
| <a href="#">[in] BSTR strAddress</a>        |    |
| <a href="#">[in] long iLaunchFlags</a>      |    |
| <a href="#">[in] BSTR strProjectPath</a>    |    |
| <a href="#">[in] BSTR strCmdLine)</a>       | 15 |
| <a href="#">HRESULT ScriptConnectEx2(</a>   |    |
| <a href="#">[in] BSTR strAddress</a>        |    |
| <a href="#">[in] long iLaunchFlags</a>      |    |
| <a href="#">[in] BSTR strProjectPath</a>    |    |
| <a href="#">[in] BSTR strCmdLine)</a>       | 15 |
| <a href="#">strAddress</a>                  | 15 |
| <a href="#">iLaunchFlags</a>                | 15 |
| <a href="#">strProjectPath</a>              | 15 |
| <a href="#">strCmdLine</a>                  | 16 |
| <a href="#">HRESULT ConnectEx3(</a>         |    |
| <a href="#">[in] BSTR strAddress</a>        |    |
| <a href="#">[in] short iPort)</a>           | 17 |
| <a href="#">HRESULT ScriptConnectEx3(</a>   |    |
| <a href="#">[in] BSTR strAddress</a>        |    |
| <a href="#">[in] short iPort)</a>           | 17 |
| <a href="#">strAddress</a>                  | 17 |
| <a href="#">iPort</a>                       | 17 |
| <a href="#">HRESULT Disconnect()</a>        | 18 |
| <a href="#">HRESULT ScriptDisconnect()</a>  | 18 |
| <a href="#">HRESULT Disconnect2(</a>        |    |
| <a href="#">[in] DetachFlags iFlags)</a>    | 19 |
| <a href="#">HRESULT ScriptDisconnect2(</a>  |    |
| <a href="#">[in] DetachFlags iFlags)</a>    | 19 |
| <a href="#">Group: IDE</a>                  | 20 |
| <a href="#">HRESULT IdeDocument (</a>       |    |
| <a href="#">[in] FileFlags iFileFlags</a>   |    |
| <a href="#">[in] BSTR strFileName</a>       |    |
| <a href="#">[in] BSTR strParameter</a>      |    |
| <a href="#">[in] long iParameter)</a>       | 21 |
| <a href="#">HRESULT ScriptIdeDocument (</a> |    |
| <a href="#">[in] FileFlags iFileFlags</a>   |    |
| <a href="#">[in] BSTR strFileName</a>       |    |

|   |                    |
|---|--------------------|
| <a href="#">[in] BSTR strParameter</a>                  |                    |
| <a href="#">[in] long iParameter)</a>                   | <a href="#">21</a> |
| <a href="#">iFileFlags</a>                              | <a href="#">21</a> |
| <a href="#">strParameter</a>                            | <a href="#">23</a> |
| <a href="#">iParameter</a>                              | <a href="#">23</a> |
| <a href="#">HRESULT IdeWorkspace (</a>                  |                    |
| <a href="#">[in] FileFlags iFileFlags</a>               |                    |
| <a href="#">[in] BSTR strWorkspaceName)</a>             | <a href="#">24</a> |
| <a href="#">HRESULT ScriptIdeWorkspace (</a>            |                    |
| <a href="#">[in] FileFlags iFileFlags</a>               |                    |
| <a href="#">[in] BSTR strWorkspaceName)</a>             | <a href="#">24</a> |
| <a href="#">iFileFlags</a>                              | <a href="#">24</a> |
| <a href="#">strWorkspaceName</a>                        | <a href="#">24</a> |
| <a href="#">HRESULT IdeApplication (</a>                |                    |
| <a href="#">[in] ApplicationFlags iApplicationFlags</a> |                    |
| <a href="#">[out] long * iLeft</a>                      |                    |
| <a href="#">[out] long * iTop</a>                       |                    |
| <a href="#">[out] long * iRight</a>                     |                    |
| <a href="#">[out] long * iBottom</a>                    |                    |
| <a href="#">[out] long * iPID)</a>                      | <a href="#">25</a> |
| <a href="#">HRESULT ScriptIdeApplication(</a>           |                    |
| <a href="#">[in] ApplicationFlags iApplicationFlags</a> |                    |
| <a href="#">[out] VARIANT * vLeft</a>                   |                    |
| <a href="#">[out] VARIANT * vTop</a>                    |                    |
| <a href="#">[out] VARIANT * vRight</a>                  |                    |
| <a href="#">[out] VARIANT * vBottom</a>                 |                    |
| <a href="#">[out] VARIANT * vPID)</a>                   | <a href="#">25</a> |
| <a href="#">iApplicationFlags</a>                       | <a href="#">25</a> |
| <a href="#">iLeft, vLeft</a>                            | <a href="#">25</a> |
| <a href="#">iTop, vTop</a>                              | <a href="#">25</a> |
| <a href="#">iRight, vRight</a>                          | <a href="#">25</a> |
| <a href="#">iBottom, vBottom</a>                        | <a href="#">25</a> |
| <a href="#">Group: Debug</a>                            | <a href="#">26</a> |
| <a href="#">HRESULT GetInfo (</a>                       |                    |
| <a href="#">[out] short * iCpu</a>                      |                    |
| <a href="#">[out] short * iCpuVariant)</a>              | <a href="#">27</a> |
| <a href="#">HRESULT ScriptDebugGetInfo(</a>             |                    |
| <a href="#">[out] VARIANT * vCpu</a>                    |                    |
| <a href="#">[out] VARIANT * vCpuVariant)</a>            | <a href="#">27</a> |
| <a href="#">iCpu, vCpu</a>                              | <a href="#">27</a> |
| <a href="#">iCpuVariant, vCpuVariant</a>                | <a href="#">27</a> |
| <a href="#">HRESULT GetAddress (</a>                    |                    |
| <a href="#">[in] AccessFlags iAccessFlags</a>           |                    |
| <a href="#">[in] BSTR strExpression</a>                 |                    |
| <a href="#">[out] short * iMemArea</a>                  |                    |
| <a href="#">[out] long * iAddress</a>                   |                    |
| <a href="#">[out] long * iSizeMAU</a>                   |                    |
| <a href="#">[out] short * iType)</a>                    | <a href="#">28</a> |
| <a href="#">HRESULT ScriptDebugGetAddress(</a>          |                    |
| <a href="#">[in] AccessFlags iAccessFlags</a>           |                    |
| <a href="#">[in] BSTR strExpression</a>                 |                    |
| <a href="#">[out] VARIANT * vMemArea</a>                |                    |
| <a href="#">[out] VARIANT * vAddress</a>                |                    |
| <a href="#">[out] VARIANT * vSizeMAU</a>                |                    |
| <a href="#">[out] VARIANT * vType)</a>                  | <a href="#">28</a> |
| <a href="#">iAccessFlags</a>                            | <a href="#">28</a> |
| <a href="#">strExpression</a>                           | <a href="#">28</a> |
| <a href="#">iMemArea, vMemArea</a>                      | <a href="#">28</a> |
| <a href="#">iAddress, vAddress</a>                      | <a href="#">28</a> |
| <a href="#">iSizeMAU, vSizeMAU</a>                      | <a href="#">28</a> |

|   |    |
|---|----|
| iType, vType.....   | 29 |
| HRESULT DebugGetSymbol(<br>[in] SymbolFlags iSymbolFlags,<br>[in] short iMemArea,<br>[in] long iAddress,<br>[out] BSTR * strName).....                        | 30 |
| HRESULT ScriptDebugGetSymbol(<br>[in] SymbolFlags iSymbolFlags,<br>[in] short iMemArea,<br>[in] long iAddress,<br>[out] VARIANT * vName).....                 | 30 |
| iSymbolFlags.....   | 30 |
| iMemArea.....   | 30 |
| iAddress.....   | 30 |
| strName, vName.....   | 30 |
| HRESULT DebugGetSourceAddress(<br>[in] SourceFlags iSourceFlags,<br>[in] BSTR strFileName,<br>[in] long iLine,<br>[in] [out] VARIANT * vAddresses).....       | 31 |
| HRESULT ScriptDebugGetSourceAddress(<br>[in] SourceFlags iSourceFlags,<br>[in] BSTR strFileName,<br>[in] long iLine,<br>[in] [out] VARIANT * vAddresses)..... | 31 |
| iSourceFlags.....   | 31 |
| strFileName.....  | 31 |
| iLine.....  | 31 |
| vAddress.....   | 31 |
| HRESULT DebugGetAddressSource(<br>[in] SourceFlags iSourceFlags,<br>[in] long iAddress,<br>[out] BSTR * strFileName,<br>[out] long * iLine).....              | 32 |
| HRESULT ScriptDebugGetAddressSource(<br>[in] SourceFlags iSourceFlags,<br>[in] long iAddress,<br>[out] VARIANT * vFileName,<br>[out] VARIANT * vLine).....    | 32 |
| iSourceFlags.....   | 32 |
| iAddress.....   | 32 |
| strFileName, vFileName.....   | 32 |
| iLine, vLine.....   | 32 |
| HRESULT DebugEvaluateString(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strExpression,<br>[out] BSTR * strResult).....                                    | 33 |
| HRESULT ScriptDebugEvaluateString(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strExpression,<br>[out] VARIANT * vResult).....                             | 33 |
| HRESULT DebugEvaluateVariant(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strExpression,<br>[out] VARIANT * vResult).....                                  | 33 |
| HRESULT ScriptDebugEvaluateVariant(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strExpression,<br>[out] VARIANT * vResult).....                            | 33 |

|   |    |
|---|----|
| <a href="#"><u>HRESULT DebugEvaluateAddress(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[out] short * iMemArea,<br/>[out] long * iAddress)</u></a> .....   | 33 |
| <a href="#"><u>HRESULT ScriptDebugEvaluateAddress(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[out] VARIANT * vMemArea,<br/>[out] VARIANT * vAddress)</u></a> .....  | 33 |
| <a href="#"><u>iAccessFlags</u></a> .....   | 33 |
| <a href="#"><u>strExpression</u></a> .....  | 34 |
| <a href="#"><u>strResult, vResult</u></a> .....   | 34 |
| <a href="#"><u>iMemArea, vMemArea, iAddress, vAddress</u></a> .....   | 34 |
| <a href="#"><u>HRESULT DebugModifyString(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[in] BSTR strValue)</u></a> .....   | 35 |
| <a href="#"><u>HRESULT ScriptDebugModifyString(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[in] BSTR strValue)</u></a> .....   | 35 |
| <a href="#"><u>HRESULT DebugModifyVariant(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[in] VARIANT vValue)</u></a> .....   | 35 |
| <a href="#"><u>HRESULT ScriptDebugModifyVariant(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[in] VARIANT vValue)</u></a> .....   | 35 |
| <a href="#"><u>HRESULT DebugModifyAddress(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[in] short iMemArea,<br/>[in] long iAddress)</u></a> .....   | 35 |
| <a href="#"><u>HRESULT ScriptDebugModifyAddress(<br/>[in] AccessFlags iAccessFlags,<br/>[in] BSTR strExpression,<br/>[in] short iMemArea,<br/>[in] long iAddress)</u></a> .....   | 35 |
| <a href="#"><u>iAccessFlags</u></a> .....   | 35 |
| <a href="#"><u>strExpression</u></a> .....  | 36 |
| <a href="#"><u>strValue</u></a> .....   | 36 |
| <a href="#"><u>iMemArea, vMemArea, iAddress, vAddress</u></a> .....   | 36 |
| <a href="#"><u>HRESULT DebugReadMemory(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] long iNumMau,<br/>[in] short iBytesPerMau,<br/>[in] [out] VARIANT * vDataBuffer,<br/>[in] [out] VARIANT * vAccessBuffer)</u></a> .....       | 37 |
| <a href="#"><u>HRESULT ScriptDebugReadMemory(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] long iNumMau,<br/>[in] short iBytesPerMau,<br/>[in] [out] VARIANT * vDataBuffer,<br/>[in] [out] VARIANT * vAccessBuffer)</u></a> ..... | 37 |

|   |                    |
|---|--------------------|
| <a href="#">iAccessFlags.....</a>   | <a href="#">37</a> |
| <a href="#">iMemArea.....</a>   | <a href="#">37</a> |
| <a href="#">iAddress.....</a>   | <a href="#">37</a> |
| <a href="#">iNumMau.....</a>  | <a href="#">37</a> |
| <a href="#">iBytesPerMau.....</a>   | <a href="#">37</a> |
| <a href="#">vDataBuffer.....</a>  | <a href="#">38</a> |
| <a href="#">vAccessBuffer.....</a>  | <a href="#">38</a> |
| <a href="#">HRESULT DebugWriteMemory(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] long iNumMau,<br/>[in] short iBytesPerMau,<br/>[in] VARIANT * vDataBuffer,<br/>[in] [out] VARIANT * vAccessBuffer)</a>       | <a href="#">39</a> |
| <a href="#">HRESULT ScriptDebugWriteMemory(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] long iNumMau,<br/>[in] short iBytesPerMau,<br/>[in] VARIANT * vDataBuffer,<br/>[in] [out] VARIANT * vAccessBuffer)</a> | <a href="#">39</a> |
| <a href="#">iAccessFlags.....</a>   | <a href="#">39</a> |
| <a href="#">iMemArea.....</a>   | <a href="#">39</a> |
| <a href="#">iAddress.....</a>   | <a href="#">39</a> |
| <a href="#">iNumMau.....</a>  | <a href="#">39</a> |
| <a href="#">iBytesPerMau.....</a>   | <a href="#">39</a> |
| <a href="#">vDataBuffer.....</a>  | <a href="#">40</a> |
| <a href="#">vAccessBuffer.....</a>  | <a href="#">40</a> |
| <a href="#">HRESULT DebugReadValue(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] [out] VARIANT * vValue)</a>  | <a href="#">41</a> |
| <a href="#">HRESULT ScriptDebugReadValue(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] [out] VARIANT * vValue)</a>  | <a href="#">41</a> |
| <a href="#">iAccessFlags.....</a>   | <a href="#">41</a> |
| <a href="#">iMemArea.....</a>   | <a href="#">41</a> |
| <a href="#">iAddress.....</a>   | <a href="#">41</a> |
| <a href="#">HRESULT DebugWriteValue(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] VARIANT vValue)</a>   | <a href="#">42</a> |
| <a href="#">HRESULT ScriptDebugWriteValue(<br/>[in] AccessFlags iAccessFlags,<br/>[in] short iMemArea,<br/>[in] long iAddress,<br/>[in] VARIANT vValue)</a>   | <a href="#">42</a> |
| <a href="#">iAccessFlags.....</a>   | <a href="#">42</a> |
| <a href="#">iMemArea.....</a>   | <a href="#">42</a> |
| <a href="#">iAddress.....</a>   | <a href="#">42</a> |
| <a href="#">vValue.....</a>   | <a href="#">42</a> |
| <a href="#">HRESULT DebugReadRegister(<br/>[in] AccessFlags iAccessFlags,</a>   |                    |

|  |    |
|--|----|
| [in] BSTR strRegisterName,<br>[out] VARIANT * vValue).....   | 43 |
| HRESULT ScriptDebugReadRegister(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strRegisterName,<br>[out] VARIANT * vValue).....   | 43 |
| iAccessFlags.....  | 43 |
| strRegisterName.....   | 43 |
| vValue.....  | 43 |
| HRESULT DebugWriteRegister(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strRegisterName,<br>[in] VARIANT vValue).....   | 44 |
| HRESULT ScriptDebugWriteRegister(<br>[in] AccessFlags iAccessFlags,<br>[in] BSTR strRegisterName,<br>[in] VARIANT vValue).....   | 44 |
| iAccessFlags.....  | 44 |
| strRegisterName.....   | 44 |
| vValue.....  | 44 |
| HRESULT DebugGetStatus(<br>[in] StatusFlags iStatusFlags,<br>[out] short * iStatus,<br>[out] short * iExecutionArea,<br>[out] long * iAddress).....                          | 45 |
| HRESULT ScriptDebugGetStatus(<br>[in] StatusFlags iStatusFlags,<br>[out] VARIANT * vStatus,<br>[out] VARIANT * vExecutionArea,<br>[out] VARIANT * vAddress).....             | 45 |
| dwStatusFlags.....   | 45 |
| iStatus, vStatus.....  | 45 |
| iExecutionArea, vExecutionArea.....  | 45 |
| iAddress, vAddress.....  | 45 |
| HRESULT DebugSetBreakpoint(<br>[in] BreakpointFlags iBreakpointFlags,<br>[in] short iMemArea,<br>[in] long iAddress,<br>[in] BSTR strAddress,<br>[in] long iLine).....       | 46 |
| HRESULT ScriptDebugSetBreakpoint(<br>[in] BreakpointFlags iBreakpointFlags,<br>[in] short iMemArea,<br>[in] long iAddress,<br>[in] BSTR strAddress,<br>[in] long iLine)..... | 46 |
| iBreakpointFlags.....  | 46 |
| iMemArea.....  | 46 |
| iAddress.....  | 46 |
| strAddress.....  | 46 |
| iLine.....   | 46 |
| HRESULT DebugRunControl(<br>[in] RunControlFlags iRunControlFlags,<br>[in] short iMemArea,<br>[in] long iAddress).....   | 47 |
| HRESULT ScriptDebugRunControl(<br>[in] RunControlFlags iRunControlFlags,<br>[in] short iMemArea,<br>[in] long iAddress).....   | 47 |
| iRunControlFlags.....  | 47 |

|   |    |
|---|----|
| iMemArea.....                                       | 47 |
| iAddress.....                                       | 47 |
| Remarks.....  | 47 |
| HRESULT DebugDownload(                              |    |
| [in] DownloadFlags iDownloadFlags..                 |    |
| [in] [out] DownloadFileType * iDownloadFileType..   |    |
| [in] [out] DownloadFileFlags * iDownloadFileFlags.. |    |
| [in] [out] short iListIndex..                       |    |
| [in] [out] BSTR * strFileName..                     |    |
| [in] [out] BSTR * strOptions..                      |    |
| [in] [out] short * iMemArea..                       |    |
| [in] [out] long * iCodeOffset..                     |    |
| [in] [out] long * iSymbolsOffset):.....             | 48 |
| HRESULT ScriptDebugDownload(                        |    |
| [in] DownloadFlags iDownloadFlags..                 |    |
| [in] [out] VARIANT * vDownloadFileType..            |    |
| [in] [out] VARIANT * vDownloadFileFlags..           |    |
| [in] [out] VARIANT * vListIndex..                   |    |
| [in] [out] VARIANT * vFileName..                    |    |
| [in] [out] VARIANT * vOptions..                     |    |
| [in] [out] short * iMemArea..                       |    |
| [in] [out] long * iCodeOffset..                     |    |
| [in] [out] long * iSymbolsOffset):.....             | 48 |
| iDownloadFlags.....                                 | 48 |
| iDownloadFileType,vDownloadFileType.....            | 48 |
| iDownloadFileFlags,vDownloadFileFlags.....          | 49 |
| strFileName,vFileName.....                          | 49 |
| strOptions,vOptions.....                            | 50 |
| iMemArea,vMemArea.....                              | 50 |
| iCodeOffset,vCodeOffset.....                        | 50 |
| iSymbolsOffset,vSymbolsOffset.....                  | 50 |
| Group: Profiler.....                                | 51 |
| HRESULT ProfilerGetInfo(                            |    |
| [in] [out] ProfilerInfoCapabilitiesFlags * iFlags.. |    |
| [out] long * iNumExecAreas..                        |    |
| [out] long * iNumDataAreas..                        |    |
| [out] short * iSingleDataWidth):.....               | 52 |
| HRESULT ScriptProfilerGetInfo(                      |    |
| [in] [out] VARIANT * vFlags..                       |    |
| [out] VARIANT * vNumExecAreas..                     |    |
| [out] VARIANT * vNumDataAreas..                     |    |
| [out] VARIANT * vSingleDataWidth):.....             | 52 |
| iFlags.....   | 52 |
| iNumExecAreas, vNumExecAreas.....                   | 52 |
| iNumDataAreas, vNumDataAreas.....                   | 52 |
| iSingleDataWidth, vSingleDataWidth.....             | 52 |
| HRESULT ProfilerStartConfig(                        |    |
| [in] StartConfigFlags iFlags..                      |    |
| [in] ProfilerStartingPointEvent iEvent..            |    |
| [in] short iMemArea..                               |    |
| [in] short iDataSize..                              |    |
| [in] long iAddress..                                |    |
| [in] long iAddressRM..                              |    |
| [in] long iDataAux..                                |    |
| [in] long iDataAuxRm):.....                         | 53 |
| HRESULT ScriptProfilerStartConfig(                  |    |
| [in] StartConfigFlags iFlags..                      |    |
| [in] ProfilerStartingPointEvent iEvent..            |    |
| [in] short iMemArea..                               |    |
| [in] short iDataSize..                              |    |

|  |                    |
|--|--------------------|
| <a href="#">[in] long iAddress.</a>                          |                    |
| <a href="#">[in] long iAddressRM.</a>                        |                    |
| <a href="#">[in] long iDataAux.</a>                          |                    |
| <a href="#">[in] long iDataAuxRm).....</a>                   | <a href="#">53</a> |
| <a href="#">  dwStartConfigFlags.....</a>                    | <a href="#">53</a> |
| <a href="#">  iEvent.....</a>                                | <a href="#">53</a> |
| <a href="#">  iMemArea.....</a>                              | <a href="#">54</a> |
| <a href="#">  iDataSize.....</a>                             | <a href="#">54</a> |
| <a href="#">  iAddress and iAddressRM.....</a>               | <a href="#">54</a> |
| <a href="#">  iDataAux and iDataAuxRm.....</a>               | <a href="#">54</a> |
| <a href="#">HRESULT ProfilerAddArea(</a>                     |                    |
| <a href="#">  [in] ProfilerAreaFlags iFlags.</a>             |                    |
| <a href="#">  [out] long * iHandle.</a>                      |                    |
| <a href="#">  [in] BSTR strName.</a>                         |                    |
| <a href="#">  [in] short iMemArea.</a>                       |                    |
| <a href="#">  [in] long iAddress.</a>                        |                    |
| <a href="#">  [in] long iSize.</a>                           |                    |
| <a href="#">  [in] [out] VARIANT * vExits):.....</a>         | <a href="#">55</a> |
| <a href="#">HRESULT ScriptProfilerAddArea(</a>               |                    |
| <a href="#">  [in] ProfilerAreaFlags iFlags.</a>             |                    |
| <a href="#">  [out] VARIANT * vHandle.</a>                   |                    |
| <a href="#">  [in] BSTR strName.</a>                         |                    |
| <a href="#">  [in] short iMemArea.</a>                       |                    |
| <a href="#">  [in] long iAddress.</a>                        |                    |
| <a href="#">  [in] long iSize.</a>                           |                    |
| <a href="#">  [in] [out] VARIANT * vExits):.....</a>         | <a href="#">55</a> |
| <a href="#">  iFlags.....</a>                                | <a href="#">55</a> |
| <a href="#">  iHandle.....</a>                               | <a href="#">55</a> |
| <a href="#">  strName.....</a>                               | <a href="#">55</a> |
| <a href="#">  iMemArea, iAddress.....</a>                    | <a href="#">55</a> |
| <a href="#">  iSize.....</a>                                 | <a href="#">55</a> |
| <a href="#">  vExits.....</a>                                | <a href="#">56</a> |
| <a href="#">HRESULT ProfilerActivate(</a>                    |                    |
| <a href="#">  [in] ProfilerActivateFlags iFlags):.....</a>   | <a href="#">57</a> |
| <a href="#">HRESULT ScriptProfilerActivate(</a>              |                    |
| <a href="#">  [in] ProfilerActivateFlags iFlags):.....</a>   | <a href="#">57</a> |
| <a href="#">  iFlags.....</a>                                | <a href="#">57</a> |
| <a href="#">HRESULT ProfilerGetStatus(</a>                   |                    |
| <a href="#">  [in] ProfilerGetStatusFlags iFlags.</a>        |                    |
| <a href="#">  [out] ProfilerStatus * iStatus.</a>            |                    |
| <a href="#">  [out] long * iNumAvailable):.....</a>          | <a href="#">58</a> |
| <a href="#">HRESULT ScriptProfilerGetStatus(</a>             |                    |
| <a href="#">  [in] ProfilerGetStatusFlags iFlags.</a>        |                    |
| <a href="#">  [out] VARIANT * vStatus.</a>                   |                    |
| <a href="#">  [out] VARIANT * vNumAvailable):.....</a>       | <a href="#">58</a> |
| <a href="#">  iFlags.....</a>                                | <a href="#">58</a> |
| <a href="#">  iStatus, vStatus.....</a>                      | <a href="#">58</a> |
| <a href="#">  iNumAvailable, vNumAvailable.....</a>          | <a href="#">58</a> |
| <a href="#">HRESULT ProfilerGetStatistics(</a>               |                    |
| <a href="#">  [in] ProfilerStatisticsResultFlags iFlags.</a> |                    |
| <a href="#">  [in] long iHandle.</a>                         |                    |
| <a href="#">  [in] long iTASK.</a>                           |                    |
| <a href="#">  [in] long iValue.</a>                          |                    |
| <a href="#">  [in] [out] long * iNumStatistics.</a>          |                    |
| <a href="#">  [in] [out] VARIANT * vHandle.</a>              |                    |
| <a href="#">  [in] [out] VARIANT * vValue.</a>               |                    |
| <a href="#">  [in] [out] VARIANT * vTotalTime.</a>           |                    |
| <a href="#">  [in] [out] VARIANT * vMaxTime.</a>             |                    |
| <a href="#">  [in] [out] VARIANT * vMinTime.</a>             |                    |
| <a href="#">  [in] [out] VARIANT * vNumHits):.....</a>       | <a href="#">59</a> |

|  |    |
|--|----|
| <a href="#">HRESULT ScriptProfilerGetStatistics(<br/>[in] ProfilerStatisticsResultFlags iFlags,<br/>[in] long iHandle,<br/>[in] long iTASK,<br/>[in] long iValue,<br/>[in] [out] VARIANT vNumStatistics,<br/>[in] [out] VARIANT * vHandle,<br/>[in] [out] VARIANT * vValue,<br/>[in] [out] VARIANT * vTotalTime,<br/>[in] [out] VARIANT * vMaxTime,<br/>[in] [out] VARIANT * vMinTime,<br/>[in] [out] VARIANT * vNumHits):</a> | 59 |
| <a href="#">iFlags</a>   | 59 |
| <a href="#">iHandle</a>  | 59 |
| <a href="#">iTask</a>  | 59 |
| <a href="#">iValue</a>   | 59 |
| <a href="#">iNumStatistics, vNumStatistics</a>   | 59 |
| <a href="#">vHandle, vValue, vTotalTime, vMaxTime, vMinTime, vNumHits</a>  | 60 |
| <a href="#">HRESULT ProfilerGetHistory(<br/>[in] ProfilerResultFlags iFlags,<br/>[in] long iHandle,<br/>[in] long iTASK,<br/>[in] long iValue,<br/>[in] long iHistoryBase,<br/>[in] [out] long * iNumHistories,<br/>[in] [out] VARIANT * vHandle,<br/>[in] [out] VARIANT * vValue,<br/>[in] [out] VARIANT * vTime):</a>  | 61 |
| <a href="#">HRESULT ScriptProfilerGetHistory(<br/>[in] ProfilerResultFlags iFlags,<br/>[in] long iHandle,<br/>[in] long iTASK,<br/>[in] long iValue,<br/>[in] long iHistoryBase,<br/>[in] [out] VARIANT * vNumHistories,<br/>[in] [out] VARIANT * vHandle,<br/>[in] [out] VARIANT * vValue,<br/>[in] [out] VARIANT * vTime):</a>   | 61 |
| <a href="#">iFlags</a>   | 61 |
| <a href="#">iHandle</a>  | 61 |
| <a href="#">iTask</a>  | 61 |
| <a href="#">iValue</a>   | 61 |
| <a href="#">iHistoryBase</a>   | 61 |
| <a href="#">iNumHistories, vNumHistories</a>   | 61 |
| <a href="#">vHandle, vValue, vTime</a>   | 62 |

## **Group: Connect**

This functional group provides functions to connect to winIDEA and disconnect from it.

## **HRESULT Connect()**

## **HRESULT ScriptConnect()**

Connects to winIDEA. Connects to the started instance of winIDEA. In case that there is no currently running instance of winIDEA it starts a new instance.

**HRESULT ConnectEx(  
[in] BSTR strAddress,  
[in] short iPort,  
[in] BSTR strLibPath,  
[in] long iLaunchFlags)**

**HRESULT ScriptConnectEx(  
[in] BSTR strAddress,  
[in] short iPort,  
[in] BSTR strLibPath,  
[in] long iLaunchFlags)**

Connects to winIDEA. It can optionally start a new instance of the application.

*strAddress*

The address of the machine where winIDEA is running. This address can be given in any format recognized by the system. Typically supported formats include dotted decimal and fully qualified name.

If strAddress is *NULL*, the local host is assumed.

*iPort*

The target TCP port number. This is the port configured in the winIDEA instance to which the connection is attempted.

*strLibPath*

The path to the iConnect library (DLL). If the strLibPath parameter is NULL, the library is searched in the directories specified in the system's and user's path environment variables.

*iLaunchFlags*

The *IfStartXXXX* flags determine under what conditions to start a new instance of winIDEA.

|                   |   |   |
|-------------------|---|---|
| IfStartNever      | Never starts a new winIDEA                      | 0 |
| IfStartIfRequired | Starts winIDEA if an instance isn't running yet | 1 |
| IfStartAlways     | Always launch a new instance                    | 2 |
| IfStartExisting   | Find an existing instance, do not launch new.   | 3 |

The *IfWaitXXXX* flags determine how long the iConnect should wait for running instances to respond.

|               |                      |     |
|---------------|----------------------|-----|
| IfWaitDefault | Currently one second | 0   |
| IfWait30ms    | 30 milliseconds      | 16  |
| IfWait100ms   | 100 milliseconds     | 32  |
| IfWait300ms   | 300 milliseconds     | 48  |
| IfWait1s      | 1 second             | 64  |
| IfWait3s      | 3 seconds            | 80  |
| IfWait10s     | 10 seconds           | 96  |
| IfWait30s     | 30 seconds           | 112 |

The *IfShowXXXX* flags determine how the newly launched instance will start

|                 |                              |     |
|-----------------|------------------------------|-----|
| lfShowDefault   | No special provision         | 0   |
| lfShowMinimized | winIDEA is started minimized | 256 |
| lfShowMaximized | winIDEA is started maximized | 512 |

**HRESULT ConnectEx2(  
[in] BSTR strAddress,  
[in] long iLaunchFlags,  
[in] BSTR strProjectPath,  
[in] BSTR strCmdLine)**

**HRESULT ScriptConnectEx2(  
[in] BSTR strAddress,  
[in] long iLaunchFlags,  
[in] BSTR strProjectPath,  
[in] BSTR strCmdLine)**

Connects to winIDEA. It can optionally start a new instance of the application.

*strAddress*

The address of the machine where winIDEA is running. This address can be given in any format recognized by the system. Typically supported formats include dotted decimal and fully qualified name.

If strAddress is *NULL*, the local host is assumed.

*iLaunchFlags*

The *IfStartXXXX* flags determine under what conditions to start a new instance of winIDEA.

|                   |   |   |
|-------------------|---|---|
| IfStartNever      | Never starts a new winIDEA                      | 0 |
| IfStartIfRequired | Starts winIDEA if an instance isn't running yet | 1 |
| IfStartAlways     | Always launch a new instance                    | 2 |
| IfStartExisting   | Find an existing instance, do not launch new.   | 3 |

The *IfWaitXXXX* flags determine how long the iConnect should wait for running instances to respond.

|               |                      |     |
|---------------|----------------------|-----|
| IfWaitDefault | Currently one second | 0   |
| IfWait30ms    | 30 milliseconds      | 16  |
| IfWait100ms   | 100 milliseconds     | 32  |
| IfWait300ms   | 300 milliseconds     | 48  |
| IfWait1s      | 1 second             | 64  |
| IfWait3s      | 3 seconds            | 80  |
| IfWait10s     | 10 seconds           | 96  |
| IfWait30s     | 30 seconds           | 112 |

The *IfShowXXXX* flags determine how the newly launched instance will start

|                 |                              |     |
|-----------------|------------------------------|-----|
| IfShowDefault   | No special provision         | 0   |
| IfShowMinimized | winIDEA is started minimized | 256 |
| IfShowMaximized | winIDEA is started maximized | 512 |

*strProjectPath*

The path to the project.

*strCmdLine*

The command line string .

**HRESULT ConnectEx3(  
[in] BSTR strAddress,  
[in] short iPort)**

**HRESULT ScriptConnectEx3(  
[in] BSTR strAddress,  
[in] short iPort)**

Connects to winIDEA.

*strAddress*

The address of the machine where winIDEA is running. This address can be given in any format recognized by the system. Typically supported formats include dotted decimal and fully qualified name.

If strAddress is *NULL*, the local host is assumed.

*iPort*

The target TCP port number. This is the port configured in the winIDEA instance to which the connection is attempted.

**HRESULT Disconnect()**

**HRESULT ScriptDisconnect()**

Disconnects from winIDEA.

**HRESULT Disconnect2(  
[in] DetachFlags iFlags)**

**HRESULT ScriptDisconnect2(  
[in] DetachFlags iFlags)**

Disconnects from winIDEA.

The *Detach* flags determine is winIDEA will be closed or remain open and how modified files will be handled. Use any combination of the following values, at most one from each table

Exit flags:

|                            |   |   |
|----------------------------|---|---|
| dfCloseServerIfLastClient  | winIDEA will exit if this is the last client attached | 1 |
| dfCloseServerUnconditional | winIDEA will exit unconditionally                     | 2 |

Save flags:

|                        |  |    |
|------------------------|--|----|
| dfCloseAutoSaveDefault | when closing default Prompts will be displayed               | 16 |
| dfCloseAutoSaveAll     | all documents and workspaces will be saved without prompting | 32 |
| dfCloseAutoSaveNone    | all changes to documents and workspace will be discarded     | 64 |

## **Group: IDE**

The IDE function group provides access to the winIDEA IDE.

**HRESULT IdeDocument (**  
**[in] FileFlags iFileFlags,**  
**[in] BSTR strFileName,**  
**[in] BSTR strParameter,**  
**[in] long iParameter)**

**HRESULT ScriptIdeDocument (**  
**[in] FileFlags iFileFlags,**  
**[in] BSTR strFileName,**  
**[in] BSTR strParameter,**  
**[in] long iParameter)**

Manages document files in winIDEA.

*iFileFlags*

The file flags should be composed of one value from each group. To join values add them together.

Operation:

|                |  |    |
|----------------|--|----|
| ffClose        | Close.                                 | 1  |
| ffCloseAll     | Close all.                             | 2  |
| ffSave         | Save under its original name.          | 3  |
| ffSaveAs       | Save under strFileName.                | 4  |
| ffSaveAsPrompt | Save using Save As... prompt.          | 5  |
| ffSaveAll      | Save all.                              | 6  |
| ffNew          | Create new named strFileName.          | 7  |
| ffNewPrompt    | Create new prompt for name and folder. | 8  |
| ffOpen         | Open strFileName.                      | 9  |
| ffOpenPrompt   | Prompt to open.                        | 10 |

Document type (used only with *ffNew*)

|                |                             |            |
|----------------|-----------------------------|------------|
| dtText         | ASCII Text file.            | 0x00000000 |
| dtAnalyzer     | winIDEA Analyzer type.      | 0x01000000 |
| dtCodeCoverage | winIDEA Code Coverage type. | 0x02000000 |
| dtDataCoverage | winIDEA Data Coverage type. | 0x03000000 |
| dtMask         | Mask for the document type. | 0xFF000000 |

Special windows:

|                 |   |            |
|-----------------|---|------------|
| dfSpecialWindow | This call refers to a non document window, defined by the dwXXXX value. The strFileName parameter is ignored. | 0x00080000 |
| dwTerminal      | Terminal window, supports <i>daStart</i> , <i>daStop</i> and <i>daStatus</i> actions                          | 0x01000000 |
| dwMask          | Mask for the special window type.   | 0x0F000000 |

Focus:

|         |   |            |
|---------|---|------------|
| dfFocus | Focus the file window, if dwParameter if non-zero, the file is scrolled | 0x00040000 |
|---------|---|------------|

|  |                        |  |
|--|------------------------|--|
|  | to the given position. |  |
|--|------------------------|--|

Marker:

|             |  |            |
|-------------|--|------------|
| dmMarkSet   | Puts a marker on the requested position (iParameter).    | 0x00010000 |
| dmMarkClear | Clears the global editor marker. strFileName is ignored. | 0x00020000 |
| dmMarkMask  | The mark mask.   | 0x00030000 |

Actions:

|           |  |            |
|-----------|--|------------|
| daStart   | Starts the document action (start trace, script, etc.)   | 0x10000000 |
| daStop    | Stop the document action.  | 0x20000000 |
| daStatus  | Probe the status of the document action. The function returns <b>S_OK</b> to indicate idle/finished, <b>S_FALSE</b> to indicate a busy state.  | 0x30000000 |
| daResume  | Resume the document action.  | 0x40000000 |
| daSelect  | Select a property of the document specified by <i>pszParameter</i> and <i>dwParameter</i> .  | 0x50000000 |
| daExport  | Export the document to a different format. <i>pszParameter</i> specifies the file to export to, <i>dwParameter</i> specifies the scope and export format to use. The export format is document specific, see the export format tables. | 0x60000000 |
| daExportV | Same as <i>daExport</i> , but the OS associated viewer is launched after export  | 0x70000000 |
| daMask    | Document action mask.  | 0xF0000000 |

Export format for coverage:

|               |                         |   |
|---------------|-------------------------|---|
| etcHTML       | HTML report.            | 0 |
| etcText       | Text file format.       | 1 |
| etcCSV        | Comma separated values. | 2 |
| etcXML        | XML file format.        | 3 |
| etcReviewHTML | HTML review report.     | 4 |
| etcReviewText | Text review report.     | 5 |

Export format for trace:

|           |                         |   |
|-----------|-------------------------|---|
| ettText   | Original text format.   | 1 |
| ettCSV    | Comma separated values. | 2 |
| ettBinary | Binary format.          | 3 |

Export format for profiler:

|         |                       |   |
|---------|-----------------------|---|
| etpText | Original text format. | 1 |
| etpHTML | HTML report.          | 2 |
| etpXML  | XML file format.      | 3 |

strFileName

File path. If this path is relative, winIDEA will consider the workspace file directory as the root.

*strParameter*

Parameter to the requested action, depending on the action (ffSaveAs, daSelect).

*iParameter*

Parameter for the requested action (dfFocus, dmMarkSet, daSelect, daExport). Use with matching flags.

**HRESULT IdeWorkspace (**  
**[in] FileFlags iFileFlags,**  
**[in] BSTR strWorkspaceName)**

**HRESULT ScriptIdeWorkspace (**  
**[in] FileFlags iFileFlags,**  
**[in] BSTR strWorkspaceName)**

Manage workspaces in winIDEA.

*iFileFlags*

|                |                                       |    |
|----------------|---------------------------------------|----|
| ffClose        | Close                                 | 1  |
| ffCloseAll     | Close all                             | 2  |
| ffSave         | Save under its original name          | 3  |
| ffSaveAs       | Save under strWorkspaceName           | 4  |
| ffSaveAsPrompt | Save using Save As... prompt          | 5  |
| ffSaveAll      | Save all                              | 6  |
| ffNew          | Create new named strWorkspaceName     | 7  |
| ffNewPrompt    | Create new prompt for name and folder | 8  |
| ffOpen         | Open strWorkspaceName                 | 9  |
| ffOpenPrompt   | Prompt to open                        | 10 |

*strWorkspaceName*

Workspace file path. If this path is relative, winIDEA will consider the workspace file directory as the root.

**HRESULT IdeApplication (**  
**[in] ApplicationFlags iApplicationFlags,**  
**[out] long \* iLeft,**  
**[out] long \* iTop,**  
**[out] long \* iRight,**  
**[out] long \* iBottom,**  
**[out] long \* iPID)**

**HRESULT ScriptIdeApplication(**  
**[in] ApplicationFlags iApplicationFlags,**  
**[out] VARIANT \* vLeft,**  
**[out] VARIANT \* vTop,**  
**[out] VARIANT \* vRight,**  
**[out] VARIANT \* vBottom,**  
**[out] VARIANT \* vPID)**

Manipulates the attached winIDEA window.

*iApplicationFlags*

|                  |  |   |
|------------------|--|---|
| afWindowActivate | Activate (bring to foreground) the application. Return the application window coordinates. | 1 |
| afWindowMinimize | Minimize. Return the application window coordinates.                                       | 2 |
| afWindowRestore  | Restore size. Return the application window coordinates.                                   | 3 |
| afWindowMaximize | Maximize. Return the application window coordinates.                                       | 4 |
| afWindowMove     | Move the application window to the passed coordinates.                                     | 5 |

*iLeft, vLeft*

The left coordinate of the application window.

*iTop, vTop*

The top coordinate of the application window.

*iRight, vRight*

The right coordinate of the application window.

*iBottom, vBottom*

The bottom coordinate of the application window.

## **Group: Debug**

The Debug interface provides access to debug functions exported by winIDEA.

**HRESULT GetInfo (**  
    **[out] short \* iCpu,**  
    **[out] short \* iCpuVariant)**

**HRESULT ScriptDebugGetInfo(**  
    **[out] VARIANT \* vCpu,**  
    **[out] VARIANT \* vCpuVariant)**

Returns information about the target's CPU.

*iCpu, vCpu*

Returns the CPU ID

*iCpuVariant, vCpuVariant*

The CPU variant ID

**HRESULT GetAddress (**  
**[in] AccessFlags iAccessFlags,**  
**[in] BSTR strExpression,**  
**[out] short \* iMemArea,**  
**[out] long \* iAddress,**  
**[out] long \* iSizeMAU,**  
**[out] short \* iType)**

**HRESULT ScriptDebugGetAddress(**  
**[in] AccessFlags iAccessFlags,**  
**[in] BSTR strExpression,**  
**[out] VARIANT \* vMemArea,**  
**[out] VARIANT \* vAddress,**  
**[out] VARIANT \* vSizeMAU,**  
**[out] VARIANT \* vType)**

Returns the location, size and type of an object.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |
| fCacheCode    | Get from downloaded code mirror     | 48 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*strExpression*

An expression that describes a location in memory (that can be evaluated to an IValue).

*iMemArea, vMemArea*

A variable that will receive the memory area of the object.

*iAddress, vAddress*

A variable that will receive the address of the object.

*iSizeMAU, vSizeMAU*

A variable that will receive the size of the object.

*iType, vType*

A variable that will receive the type ID of the object. Implemented types are: tSigned8, tSigned16, tSigned32, tSigned64, tUnsigned8, tUnsigned16, tUnsigned32, tUnsigned64, tFloat32, tFloat64, tAddress, tCompound, tUndefined, tUnsupported.

**HRESULT GetAddress2(**  
**[in] AccessFlags iAccessFlags,**  
**[in] BSTR strExpression,**  
**[out] short \* iMemArea,**  
**[out] long \* iAddress,**  
**[out] long \* iSizeMAU,**  
**[out] short \* iType,**  
**[out] long \* iData1,**  
**[out] long \* iData2)**

**HRESULT ScriptDebugGetAddress2(**  
**[in] AccessFlags iAccessFlags,**  
**[in] BSTR strExpression,**  
**[out] VARIANT \* vMemArea,**  
**[out] VARIANT \* vAddress,**  
**[out] VARIANT \* vSizeMAU,**  
**[out] VARIANT \* vType,**  
**[out] VARIANT \* vData1,**  
**[out] VARIANT \* vData2)**

Returns the location, size and type of an object. The parameters Data1 and Data2 provide additional information.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |
| fCacheCode    | Get from downloaded code mirror     | 48 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*strExpression*

An expression that describes a location in memory (that can be evaluated to an IValue).

*iMemArea, vMemArea*

A variable that will receive the memory area of the object.

*iAddress, vAddress*

A variable that will receive the address of the object.

*iSizeMAU, vSizeMAU*

A variable that will receive the size of the object.

*iType, vType*

A variable that will receive the type ID of the object. Implemented types are: tSigned8, tSigned16, tSigned32, tSigned64, tUnsigned8, tUnsigned16, tUnsigned32, tUnsigned64, tFloat32, tFloat64, tAddress, tCompound, tUndefined, tUnsupported.

*iData1, vData1*

When the expression evaluates to a bit field, this variable holds the size in bits.

*iData2, vData2*

When the expression evaluates to a bit field, this variable holds the offset in bits inside the containing type.

**HRESULT DebugGetSymbol(  
[in] SymbolFlags iSymbolFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[out] BSTR \* strName)**

**HRESULT ScriptDebugGetSymbol(  
[in] SymbolFlags iSymbolFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[out] VARIANT \* vName)**

Returns the name of the symbol at specified address.

*iSymbolFlags*

Specifies symbol classes to consider.

|            |                                 |    |
|------------|---------------------------------|----|
| sVariables | Global variables (data objects) | 1  |
| sLabels    | Global code labels              | 2  |
| sFunctions | High level functions            | 4  |
| sLines     | Source lines                    | 8  |
| sConstants | Global constants                | 16 |

*iMemArea*

The memory area where the object is located.

*iAddress*

The address of the object.

*strName, vName*

A variable that will receive the name of the symbol.

**HRESULT DebugGetSourceAddress(  
    [in] SourceFlags iSourceFlags,  
    [in] BSTR strFileName,  
    [in] long iLine,  
    [in] [out] VARIANT \* vAddresses)**

**HRESULT ScriptDebugGetSourceAddress(  
    [in] SourceFlags iSourceFlags,  
    [in] BSTR strFileName,  
    [in] long iLine,  
    [in] [out] VARIANT \* vAddresses)**

Returns the address(es) of a source line.

*iSourceFlags*

Reserved, must be zero.

*strFileName*

File name.

*iLine*

Line number.

*vAddress*

An array of variables that will receive the addresses to which the source line translates.

**HRESULT DebugGetAddressSource(  
    [in] SourceFlags iSourceFlags,  
    [in] long iAddress,  
    [out] BSTR \* strFileName,  
    [out] long \* iLine)**

**HRESULT ScriptDebugGetAddressSource(  
    [in] SourceFlags iSourceFlags,  
    [in] long iAddress,  
    [out] VARIANT \* vFileName,  
    [out] VARIANT \* vLine)**

Returns the source code location matching an address.

*iSourceFlags*

Reserved, must be zero.

*iAddress*

The address to be enquired.

*strFileName, vFileName*

A variable that will receive the file name.

*iLine, vLine*

A variable that will receive the line number.

**HRESULT DebugEvaluateString(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [out] BSTR \* strResult)**

**HRESULT ScriptDebugEvaluateString(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [out] VARIANT \* vResult)**

**HRESULT DebugEvaluateVariant(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [out] VARIANT \* vResult)**

**HRESULT ScriptDebugEvaluateVariant(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [out] VARIANT \* vResult)**

**HRESULT DebugEvaluateAddress(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [out] short \* iMemArea,  
    [out] long \* iAddress)**

**HRESULT ScriptDebugEvaluateAddress(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [out] VARIANT \* vMemArea,  
    [out] VARIANT \* vAddress)**

Evaluates an expression and returns its value in string, variant or address format.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |
| fCacheCode    | Get from downloaded code mirror     | 48 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*strExpression*

Any valid C syntax expression.

*strResult, vResult*

A variable that will receive the result of the evaluation.

*iMemArea, vMemArea, iAddress, vAddress*

A pair of variables that will receive the result of the evaluation in address format.

**HRESULT DebugModifyString(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [in] BSTR strValue)**

**HRESULT ScriptDebugModifyString(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [in] BSTR strValue)**

**HRESULT DebugModifyVariant(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [in] VARIANT vValue)**

**HRESULT ScriptDebugModifyVariant(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [in] VARIANT vValue)**

**HRESULT DebugModifyAddress(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [in] short iMemArea,  
    [in] long iAddress)**

**HRESULT ScriptDebugModifyAddress(  
    [in] AccessFlags iAccessFlags,  
    [in] BSTR strExpression,  
    [in] short iMemArea,  
    [in] long iAddress)**

Modifies an expression. The expression must evaluate to a lvalue.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |
| fCacheCode    | Get from downloaded code mirror     | 48 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*strExpression*

Any C syntax expression that evaluates to a lvalue.

*strValue*

A variable that hold the value to wich the expression will be set.

*iMemArea, vMemArea, iAddress, vAddress*

A pair of variables that hold the value of the address to wich the expression will be set.

**HRESULT DebugReadMemory(**  
**[in] AccessFlags iAccessFlags,**  
**[in] short iMemArea,**  
**[in] long iAddress,**  
**[in] long iNumMau,**  
**[in] short iBytesPerMau,**  
**[in] [out] VARIANT \* vDataBuffer,**  
**[in] [out] VARIANT \* vAccessBuffer)**

**HRESULT ScriptDebugReadMemory(**  
**[in] AccessFlags iAccessFlags,**  
**[in] short iMemArea,**  
**[in] long iAddress,**  
**[in] long iNumMau,**  
**[in] short iBytesPerMau,**  
**[in] [out] VARIANT \* vDataBuffer,**  
**[in] [out] VARIANT \* vAccessBuffer)**

Reads memory from the target system.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |
| fCacheCode    | Get from downloaded code mirror     | 48 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*iMemArea*

The memory area that is to be accessed.

*iAddress*

Address of the first element to be accessed.

*iNumMau*

Number of MAUs (memory addressable units) to read.

*iBytesPerMau*

Number of bytes required to fit one MAU (memory addressable unit). The number of bytes required for every MAU is:

| MAU Size | Bytes Required per MAU |
|----------|------------------------|
| 1 - 8    | 1                      |
| 9 - 16   | 2                      |
| 17 - 32  | 4                      |

winIDEA will check this parameter and return `ICONNECT_E_SIZE` in case of mismatch.

### *vDataBuffer*

The buffer to receive the read out data. Note that for memory areas with MAU size other than 8 bits (PIC Program, ARM CP, etc), the `vDataBuffer` must be large enough to accommodate all data. Data is placed in consecutive locations in the buffer, i.e. the MAU from *aAddress* is placed first, MAU from *aAddress+1* is placed next etc.

The function `DebugReadMemory` expects the `vDataBuffer` to be a variant, holding an array of bytes. The function `ScriptDebugReadMemory` expect the `vDataBuffer` to be a variant, holding an array of variants. Both functions will check the input array to be of the correct type and big enough to hold the requested data.

Within a MAU the byte ordering is MSB. For example the 14 bit `maProgramPIC16XX` area a read from address 10 would yield the following format:

| ADDR = 10                          |                                 | ADDR = 11                          |                                 | ADDR = 12                          |                                 |
|------------------------------------|---------------------------------|------------------------------------|---------------------------------|------------------------------------|---------------------------------|
| xx(13-8)<br><code>pbyBuf[0]</code> | (7-0)<br><code>pbyBuf[1]</code> | xx(13-8)<br><code>pbyBuf[2]</code> | (7-0)<br><code>pbyBuf[3]</code> | xx(13-8)<br><code>pbyBuf[4]</code> | (7-0)<br><code>pbyBuf[5]</code> |

### *vAccessBuffer*

The buffer to receive information about access to every individual location. The buffer must contain at least one array element per MAU. If this parameter is `NULL`, no access information will be returned.

The function `DebugReadMemory` expects the `vDataBuffer` to be a variant, holding an array of bytes. The function `ScriptDebugReadMemory` expect the `vDataBuffer` to be a variant, holding an array of variants. Both functions will check the input array to be of the correct type and big enough to hold the requested data.

**HRESULT DebugWriteMemory(  
[in] AccessFlags iAccessFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[in] long iNumMau,  
[in] short iBytesPerMau,  
[in] VARIANT \* vDataBuffer,  
[in] [out] VARIANT \* vAccessBuffer)**

**HRESULT ScriptDebugWriteMemory(  
[in] AccessFlags iAccessFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[in] long iNumMau,  
[in] short iBytesPerMau,  
[in] VARIANT \* vDataBuffer,  
[in] [out] VARIANT \* vAccessBuffer)**

Writes memory to the target system.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*iMemArea*

The memory area that is to be written.

*iAddress*

Address of the first element to be written.

*iNumMau*

Number of MAUs (memory addressable units) to write.

*iBytesPerMau*

Number of bytes required to fit one MAU (memory addressable unit). The number of bytes required for every MAU is:

| MAU Size | Bytes Required per MAU |
|----------|------------------------|
| 1 - 8    | 1                      |
| 9 - 16   | 2                      |
| 17 - 32  | 4                      |

winIDEA will check this parameter and return ICONNECT\_E\_SIZE in case of mismatch.

### *vDataBuffer*

The buffer that holds the data to be written. Data is expected in consecutive locations in the buffer: the MAU from *aAddress* is placed first, MAU from *aAddress+1* is placed next etc.

The function `DebugWriteMemory` expects the `vDataBuffer` to be a variant, containing an array of bytes. The function `ScriptDebugWriteMemory` expect the `vDataBuffer` to be a variant, containing an array of variants. Both functions will check the input array to be of the correct type and big enough to match the passed `iNumMau` and `iBytesPerMau`.

Within a MAU the byte ordering is MSB. For example the 14 bit `maProgramPIC16XX` area a read from address 10 would yield the following format:

| ADDR = 10                          |                                 | ADDR = 11                          |                                 | ADDR = 12                          |                                 |
|------------------------------------|---------------------------------|------------------------------------|---------------------------------|------------------------------------|---------------------------------|
| xx(13-8)<br><code>pbyBuf[0]</code> | (7-0)<br><code>pbyBuf[1]</code> | xx(13-8)<br><code>pbyBuf[2]</code> | (7-0)<br><code>pbyBuf[3]</code> | xx(13-8)<br><code>pbyBuf[4]</code> | (7-0)<br><code>pbyBuf[5]</code> |

### *vAccessBuffer*

The buffer to receive information about access to every individual location. The buffer must contain at least one array element per MAU. If this parameter is NULL, no access information will be returned.

The function `DebugWriteMemory` expects the `vDataBuffer` to be a variant, containing an array of bytes. The function `ScriptDebugWriteMemory` expect the `vDataBuffer` to be a variant, containing an array of variants. Both functions will check the input array to be of the correct type and big enough to hold the requested data.

**HRESULT DebugReadValue(  
[in] AccessFlags iAccessFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[in] [out] VARIANT \* vValue)**

**HRESULT ScriptDebugReadValue(  
[in] AccessFlags iAccessFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[in] [out] VARIANT \* vValue)**

Reads a value from the target system. This function resembles *ReadMemory* but additionally it formats the value according to the type specified (including endian conversions).

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |
| fCacheCode    | Get from downloaded code mirror     | 48 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*iMemArea*

The memory area that is to be written.

*iAddress*

Address of the first element to be written.

*vValue*

Variant to receive the value. Note that the value will be formatted according to the subtype of this parameter on input. For example if before calling the function you set the vValue to have a long value, the output parameter will be a long. The default output parameter type (in case the vValue variable doesn't have a supported type on input) is long.

**HRESULT DebugWriteValue(  
[in] AccessFlags iAccessFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[in] VARIANT vValue)**

**HRESULT ScriptDebugWriteValue(  
[in] AccessFlags iAccessFlags,  
[in] short iMemArea,  
[in] long iAddress,  
[in] VARIANT vValue)**

Writes a value to the target system. This function resembles WriteMemory but additionally it formats the value according to the type specified (including endian conversions).

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*iMemArea*

The memory area that is to be written.

*iAddress*

Address of the first element to be written.

*vValue*

A variant that carries the value to be written to the specified memory location.

**HRESULT DebugReadRegister(  
[in] AccessFlags iAccessFlags,  
[in] BSTR strRegisterName,  
[out] VARIANT \* vValue)**

**HRESULT ScriptDebugReadRegister(  
[in] AccessFlags iAccessFlags,  
[in] BSTR strRegisterName,  
[out] VARIANT \* vValue)**

Reads the specified register.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|       |                           |   |
|-------|---------------------------|---|
| fCore | CPU core register         | 0 |
| fSFR  | Special function register | 4 |

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|               |                                     |    |
|---------------|-------------------------------------|----|
| fCacheNever   | Do not cache the access             | 0  |
| fCacheDefault | Cache according to winIDEA settings | 16 |
| fCacheStop    | Cache while CPU is stopped          | 32 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*strRegisterName*

Name of the register to read.

*vValue*

Variable that will receive the value.

**HRESULT DebugWriteRegister(  
[in] AccessFlags iAccessFlags,  
[in] BSTR strRegisterName,  
[in] VARIANT vValue)**

**HRESULT ScriptDebugWriteRegister(  
[in] AccessFlags iAccessFlags,  
[in] BSTR strRegisterName,  
[in] VARIANT vValue)**

Writes the specified register.

*iAccessFlags*

The type of memory access to be used. You can specify at most one flag from each group. The desired access flag value is composed by adding the individual flag values.

|       |                           |   |
|-------|---------------------------|---|
| fCore | CPU core register         | 0 |
| fSFR  | Special function register | 4 |

|           |                           |   |
|-----------|---------------------------|---|
| fMonitor  | Use regular memory access | 0 |
| fRealTime | Use real time access      | 1 |

|            |  |    |
|------------|--|----|
| fNoRefresh | Do not refresh winIDEA GUI after write operation | 64 |
|------------|--|----|

*strRegisterName*

Name of the register to read.

*vValue*

Variable that carries the value that will be written to the register.

**HRESULT DebugGetStatus(  
[in] StatusFlags iStatusFlags,  
[out] short \* iStatus,  
[out] short \* iExecutionArea,  
[out] long \* iAddress)**

**HRESULT ScriptDebugGetStatus(  
[in] StatusFlags iStatusFlags,  
[out] VARIANT \* vStatus,  
[out] VARIANT \* vExecutionArea,  
[out] VARIANT \* vAddress)**

Returns information about of the CPU status.

*dwStatusFlags*

Specifies which status is obtained.

|          |                                 |    |
|----------|---------------------------------|----|
| tCPU     | Status of the CPU               | 0  |
| rCurrent | Last status obtained by winIDEA | 0  |
| rRefresh | Status refresh is forced        | 16 |

*iStatus, vStatus*

A variable that will receive the status of the CPU. The possible CPU status values are:

|            |  |   |
|------------|--|---|
| stMustInit | the debug system must initialize                                     | 0 |
| stStopped  | CPU is stopped   | 1 |
| stRunning  | CPU is running   | 2 |
| stReset    | CPU is held in reset   | 3 |
| stHalted   | CPU is halted by target  | 4 |
| stWaiting  | CPU is halted by emulator  | 5 |
| stAttach   | debugger is initialized and waiting for hot attachment on the target | 6 |
| stIdle     | CPU idle   | 7 |

*iExecutionArea, vExecutionArea*

A variable that will receive the memory area of the currently executing instruction.

*iAddress, vAddress*

A variable that will receive the address of the currently executing instruction.

**HRESULT DebugSetBreakpoint(**  
**[in] BreakpointFlags iBreakpointFlags,**  
**[in] short iMemArea,**  
**[in] long iAddress,**  
**[in] BSTR strAddress,**  
**[in] long iLine)**

**HRESULT ScriptDebugSetBreakpoint(**  
**[in] BreakpointFlags iBreakpointFlags,**  
**[in] short iMemArea,**  
**[in] long iAddress,**  
**[in] BSTR strAddress,**  
**[in] long iLine)**

Sets, clears, enables and disables breakpoints.

*iBreakpointFlags*

Specifies the breakpoint operation to be performed. In order to compose the final flag value select one and only one flag from each group and add their values.

|          |   |   |
|----------|---|---|
| bAddress | Use the aAddress as breakpoint address              | 0 |
| bSymbol  | Use the pszAddress as breakpoint address            | 1 |
| bSource  | Use the pszAddress and dwLine as breakpoint address | 2 |

|          |  |    |
|----------|--|----|
| bSet     | Set a breakpoint   | 0  |
| bClear   | Clear a breakpoint   | 16 |
| bEnable  | Enable a breakpoint  | 32 |
| bDisable | Disable a breakpoint   | 48 |
| bAll     | Use in combination with bClear, bDisable and bEnable. When used, it applies to all configured breakpoints. | 64 |

*iMemArea*

The memory area of the breakpoint. This value is used only if **bAddress** flag is used.

*iAddress*

The address of the breakpoint. This value is used only if **bAddress** flag is used.

*strAddress*

The address of the breakpoint. If **bSymbol** flag is used this string is interpreted as a symbol (function name, code label). If **bSource** flag is used, this string is interpreted as file name.

*iLine*

If **bSource** flag is used, **dwLine** specifies the line number.

**HRESULT DebugRunControl(  
[in] RunControlFlags iRunControlFlags,  
[in] short iMemArea,  
[in] long iAddress)**

**HRESULT ScriptDebugRunControl(  
[in] RunControlFlags iRunControlFlags,  
[in] short iMemArea,  
[in] long iAddress)**

Performs a run control operation.

*iRunControlFlags*

Specify which register domain to access:

|                 |   |    |
|-----------------|---|----|
| rNothing        | Do nothing  | 0  |
| rReset          | Reset the CPU   | 1  |
| rResetAndRun    | Reset and runs the CPU  | 2  |
| rDownload       | Download the executable   | 3  |
| rStop           | Stop the CPU  | 4  |
| rRun            | Run the CPU   | 5  |
| rStep           | Execute a single instruction step or high level step according to context | 6  |
| rStepOver       | Same as rStep, do not enter subroutine calls                              | 7  |
| rStepInst       | Execute a single instruction step   | 8  |
| rStepOverInst   | Same as rStepInst, do not enter subroutine calls                          | 9  |
| rStepHigh       | Execute a high level step   | 10 |
| rStepOverHigh   | Same as rStepHigh, do not enter function calls                            | 11 |
| rRunUntil       | Run until aAddress  | 12 |
| rRunUntilReturn | Run until current function returns  | 13 |
| rGoto           | Preset execution point to aAddress  | 14 |

*iMemArea*

The address space of the breakpoint.

*iAddress*

Address used by **rRunUntil** and **rGoto** commands.

*Remarks*

Note that rRun, rRunUntil and rRunUntilReturn functions are ‘non-blocking’ – when they return the CPU might still be running.

**HRESULT DebugDownload(**  
**[in] DownloadFlags iDownloadFlags,**  
**[in] [out] DownloadFileType \* iDownloadFileType,**  
**[in] [out] DownloadFileFlags \* iDownloadFileFlags,**  
**[in] [out] short iListIndex,**  
**[in] [out] BSTR \* strFileName,**  
**[in] [out] BSTR \* strOptions,**  
**[in] [out] short \* iMemArea,**  
**[in] [out] long \* iCodeOffset,**  
**[in] [out] long \* iSymbolsOffset);**

**HRESULT ScriptDebugDownload(**  
**[in] DownloadFlags iDownloadFlags,**  
**[in] [out] VARIANT \* vDownloadFileType,**  
**[in] [out] VARIANT \* vDownloadFileFlags,**  
**[in] [out] VARIANT \* vListIndex,**  
**[in] [out] VARIANT \* vFileName,**  
**[in] [out] VARIANT \* vOptions,**  
**[in] [out] short \* iMemArea,**  
**[in] [out] long \* iCodeOffset,**  
**[in] [out] long \* iSymbolsOffset);**

Obtains download file information, adds a download file or downloads a file.

*iDownloadFlags*

|               |   |            |
|---------------|---|------------|
| daDownload    | Download this file now  | 0x00000100 |
| daDLFromList  | Download an existing file from the list. The file is identified by pszFileName.   | 0x00000200 |
| daAddToList   | Add a new DL file to the download file list   | 0x00000300 |
| daGet         | Obtain info from the list.  | 0x00000400 |
| dListMain     | Use primary download list.  | 0x00000000 |
| dListTarget   | Use target download list.   | 0x00001000 |
| dRealTime     | Use real-time memory access to download.  | 0x00010000 |
| dAbsolutePath | daGet: Return the absolute file path.<br>daAddToList: the specified path is not converted to a workspace relative path. | 0x00020000 |

*iDownloadFileType, vDownloadFileType*

Current file type

|                     |  |   |
|---------------------|--|---|
| ftBinary            |  | 0 |
| ftIntelHex          |  | 1 |
| ftMotorolaS         |  | 2 |
| ftTektronix         |  | 3 |
| ftExtendedTektronix |  | 4 |
| ftUBROF             |  | 5 |

|                    |  |    |
|--------------------|--|----|
| ftOMF166           |  | 6  |
| ftICFOFF           |  | 7  |
| ftOMF51            |  | 8  |
| ftSLOTtext         |  | 9  |
| ftMicrotek         |  | 10 |
| ftIEEE695          |  | 11 |
| ftAD2500           |  | 12 |
| ftAXE              |  | 13 |
| ftByteCraftCOD     |  | 14 |
| ftMotorolaSymbolic |  | 15 |
| ftHiTech           |  | 16 |
| ftOMF86            |  | 17 |
| ftOMF96            |  | 18 |
| ftOMF251           |  | 19 |
| ftProspero         |  | 20 |
| ftLogitechModula2  |  | 21 |
| ftELF              |  | 22 |
| ftMARC4            |  | 23 |
| ftCR16COFF         |  | 24 |
| ftTMSCOFF          |  | 25 |
| ftStabs            |  | 26 |
| ftZardoz           |  | 27 |

*iDownloadFileFlags, vDownloadFileFlags*

Flags for the current file.

|                  |  |            |
|------------------|--|------------|
| fLoad            | When used in the list, this flag indicates if the file is loaded in the full download. | 0x00000100 |
| fIsProjectOutput | This is the project output file.   | 0x00000200 |
| fUseMemoryArea   | Use m_byMemoryArea as destination memory space.  | 0x00000400 |
| fInProjectTarget | The file is taken from current project target directory.                               | 0x00000800 |
| fLoadCode        | Load code.   | 0x00001000 |
| fLoadSymbols     | Load symbols.  | 0x00002000 |
| fLoadLines       | Load line symbols.   | 0x00004000 |
| fOptTypeByName   | If two types have the same name, assume they're equal.                                 | 0x00008000 |

*strFileName, vFileName*

daGet: The name is ignored on input for operation. On output it will contain the name of the queried file.

daDLFromList: The name can be empty – in that case the iListIndex/vListIndex variable is used as an index into the download file list.

*strOptions, vOptions*

File format specific options encoded as a whitespace separated string. If the string is empty, default options are assumed.

*iMemArea, vMemArea*

The memory space to load the code from this download file.

*iCodeOffset, vCodeOffset*

Offset by which the code is to be loaded.

*iSymbolsOffset, vSymbolsOffset*

Offset by which the symbols are to be loaded.

## **Group: Profiler**

The Profiler interface provides access to profiling functions exported by winIDEA.

**HRESULT ProfilerGetInfo(  
[in] [out] ProfilerInfoCapabilitiesFlags \* iFlags,  
[out] long \* iNumExecAreas,  
[out] long \* iNumDataAreas,  
[out] short \* iSingleDataWidth)**

**HRESULT ScriptProfilerGetInfo(  
[in] [out] VARIANT \* vFlags,  
[out] VARIANT \* vNumExecAreas,  
[out] VARIANT \* vNumDataAreas,  
[out] VARIANT \* vSingleDataWidth)**

Obtains information about the profiler capabilities.

*iFlags*

Profiler capabilities. In order to parse each flag value use binary arithmetic.

|                       |   |      |     |
|-----------------------|---|------|-----|
| flAvail               | The profiler is available.                            | 0x01 | 1   |
| flCanUseStartingPoint |   | 0x02 | 2   |
| flCanProfileExec      |   | 0x04 | 4   |
| flCanProfileFuncLines |   | 0x08 | 8   |
| flCanProfileData      | Data can be profiled.                                 | 0x10 | 16  |
| flSingleData          | Single data can be profiled.                          | 0x20 | 32  |
| flSingleDataAddress   | Single data profiling requires address specification. | 0x40 | 64  |
| flSingleDataSize      | Single data profiling requires size specification.    | 0x80 | 128 |

*iNumExecAreas, vNumExecAreas*

Number of execution areas that can be profiled.

*iNumDataAreas, vNumDataAreas*

Number of data areas that can be profiled.

*iSingleDataWidth, vSingleDataWidth*

Width of single data profiling.

**HRESULT ProfilerStartConfig(**  
     [in] StartConfigFlags iFlags,  
     [in] ProfilerStartingPointEvent iEvent,  
     [in] short iMemArea,  
     [in] short iDataSize,  
     [in] long iAddress,  
     [in] long iAddressRM,  
     [in] long iDataAux,  
     [in] long iDataAuxRm)

**HRESULT ScriptProfilerStartConfig(**  
     [in] StartConfigFlags iFlags,  
     [in] ProfilerStartingPointEvent iEvent,  
     [in] short iMemArea,  
     [in] short iDataSize,  
     [in] long iAddress,  
     [in] long iAddressRM,  
     [in] long iDataAux,  
     [in] long iDataAuxRm)

Begins configuration for a new session of the profiler. Once a profiler is configured, it can be activated any number of times.

*dwStartConfigFlags*

Profiler flags.

|                 |   |      |
|-----------------|---|------|
| cfTimeStampTime | Default time resolution                                     | 0x01 |
| cfTimeStampM0   | Timestamp resolution in magnitude order of 1 second         | 0x10 |
| cfTimeStampM1   | Timestamp resolution in magnitude order of 100 milliseconds | 0x11 |
| cfTimeStampM2   | Timestamp resolution in magnitude order of 10 milliseconds  | 0x12 |
| cfTimeStampM3   | Timestamp resolution in magnitude order of 1 millisecond    | 0x13 |
| cfTimeStampM4   | Timestamp resolution in magnitude order of 100 microseconds | 0x14 |
| cfTimeStampM5   | Timestamp resolution in magnitude order of 10 microseconds  | 0x15 |
| cfTimeStampM6   | Timestamp resolution in magnitude order of 1 microsecond    | 0x16 |
| cfTimeStampM7   | Timestamp resolution in magnitude order of 100 nanoseconds  | 0x17 |
| cfTimeStampM8   | Timestamp resolution in magnitude order of 10 nanoseconds   | 0x18 |
| cfTimeStampM9   | Timestamp resolution in magnitude order of 1 nanosecond     | 0x19 |

*iEvent*

In order to compose the final flag value select one and only one flag from each group and add their values.

Start trigger:

|            |                                    |      |
|------------|------------------------------------|------|
| eAnything  | Start immediately                  | 0x00 |
| eExecution | Execution from a specified address | 0x01 |
| eRead      | Read access                        | 0x02 |

|            |   |      |
|------------|---|------|
| eWrite     | Write access  | 0x03 |
| eRW        | Read or write access  | 0x04 |
| eAUX_State | State on a certain AUX inputs. iDataAUX, iDataAUXRM determine value/mask  | 0x05 |
| eAUX_Edge  | Transition on certain AUX input. iDataRM configures a single bit only, iData value 0 determines falling, 1 rising | 0x06 |

Address starting point:

|               |   |      |
|---------------|---|------|
| eAddressRange | Starting point is execution or access within a range of addresses [iAddress - iAddressRM] | 0x00 |
| eAddressMask  | Starting point is execution or access on specific mask: (iAddress & iAddressRM)           | 0x10 |

Data starting point:

|            |  |      |
|------------|--|------|
| eDataRange | starting point is read/write of data within the range [iData - iDataRM]      | 0x00 |
| eDataMask  | starting point is read/write of data with a specific mask: (iData & iDataRM) | 0x20 |

### *iMemArea*

Memory area of iAddress

### *iDataSize*

Number of MAUs to consider. 0 = full

### *iAddress and iAddressRM*

Address used for eRead, eWrite and eRW

### *iDataAux and iDataAuxRm*

iDataAuxRm masks bits to be considered and iDataAux specifies the value. If data is of no interest, iDataAuxRm should be zero.

**HRESULT ProfilerAddArea(**  
**[in] ProfilerAreaFlags iFlags,**  
**[out] long \* iHandle,**  
**[in] BSTR strName,**  
**[in] short iMemArea,**  
**[in] long iAddress,**  
**[in] long iSize,**  
**[int] [out] VARIANT \* vExits);**

**HRESULT ScriptProfilerAddArea(**  
**[in] ProfilerAreaFlags iFlags,**  
**[out] VARIANT \* vHandle,**  
**[in] BSTR strName,**  
**[in] short iMemArea,**  
**[in] long iAddress,**  
**[in] long iSize,**  
**[in] [out] VARIANT \* vExits);**

Adds an area to the current profiler configuration.

*iFlags*

|                        |  |        |
|------------------------|--|--------|
| afValueTypeMask        | lower 4 bits used as by EProfilerDataValueType   | 0x000F |
| afTypeFunction         | The (execution) area is a high level function  | 0x0010 |
| afTypeRoutine          | The (execution) area is a low level routine  | 0x0020 |
| afTypeVariable         | The (data) area is variable specified by its name  | 0x0030 |
| afTypeRegion           | The (data) area is a region of memory specified by address and size  | 0x0040 |
| afFunctionIncludeLines | valid with afTypeFunction - includes function lines of the specified function  | 0x0100 |
| afDataTaskID           | valid with afTypeVariable, afTypeRegion, identifies that the specified area is the task ID object  | 0x0100 |
| afDataSingleData       | valid with afTypeVariable, afTypeRegion - area is the 'single data' (OTM style) object. if CPUs single data resource is configurable, byMemArea/aAddress and aSize should be specified | 0x0200 |

*iHandle*

Specifies the location of a **DWORD** type variable that will accept the handle of the new area. This handle is subsequently used in calls to **GetStatistics** and **GetHistory**.

*strName*

Specifies the name of the area. For **afTypeFunction** and **afTypeVariable**, this name is used to obtain the address. For **afTypeRoutine** and **afTypeRegion** the name will be displayed in the profiler window.

*iMemArea, iAddress*

Specify the entry point for **afTypeRoutine** and starting address for **afTypeRegion**.

*iSize*

Specifies the MAU size for **afTypeRegion**.

*vExits*

Points to an array of addresses, each specifying an exit point from *afTypeRoutine* object.

**HRESULT ProfilerActivate(  
[in] ProfilerActivateFlags iFlags);**

**HRESULT ScriptProfilerActivate(  
[in] ProfilerActivateFlags iFlags);**

Activates or deactivates the profiler. When activating the profiler, the configured areas are evaluated. Activation can fail if an area symbol does not evaluate or number of resulting monitoring points exceeds hardware capabilities.

*iFlags*

|        |   |   |
|--------|---|---|
| aStart | Activates the profiler  | 1 |
| aStop  | Deactivates the profiler – note the profiler deactivates automatically when the trace buffer is filled. | 2 |

**HRESULT ProfilerGetStatus(  
    [in] ProfilerGetStatusFlags iFlags,  
    [out] ProfilerStatus \* iStatus,  
    [out] long \* iNumAvailable);**

**HRESULT ScriptProfilerGetStatus(  
    [in] ProfilerGetStatusFlags iFlags,  
    [out] VARIANT \* vStatus,  
    [out] VARIANT \* vNumAvailable);**

Gets the profiler status.

*iFlags*

|           |                                     |   |
|-----------|-------------------------------------|---|
| sfDefault | Get the default status information. | 0 |
|-----------|-------------------------------------|---|

*iStatus, vStatus*

The status of the profiler. It can be one of the following values

|            |                                    |   |
|------------|------------------------------------|---|
| stMustInit | Not configured.                    | 0 |
| stIdle     | Configured, but not active.        | 1 |
| stWaiting  | Active, waiting for starting point | 2 |
| stActive   | Active and recording data          | 3 |

*iNumAvailable, vNumAvailable*

The number of recorded entries available in the profiler buffer.

**HRESULT ProfilerGetStatistics(**  
 [in] ProfilerStatisticsResultFlags iFlags,  
 [in] long iHandle,  
 [in] long iTask,  
 [in] long iValue,  
 [in] [out] long \* iNumStatistics,  
 [in] [out] VARIANT \* vHandle,  
 [in] [out] VARIANT \* vValue,  
 [in] [out] VARIANT \* vTotalTime,  
 [in] [out] VARIANT \* vMaxTime,  
 [in] [out] VARIANT \* vMinTime,  
 [in] [out] VARIANT \* vNumHits);

**HRESULT ScriptProfilerGetStatistics(**  
 [in] ProfilerStatisticsResultFlags iFlags,  
 [in] long iHandle,  
 [in] long iTask,  
 [in] long iValue,  
 [in] [out] VARIANT vNumStatistics,  
 [in] [out] VARIANT \* vHandle,  
 [in] [out] VARIANT \* vValue,  
 [in] [out] VARIANT \* vTotalTime,  
 [in] [out] VARIANT \* vMaxTime,  
 [in] [out] VARIANT \* vMinTime,  
 [in] [out] VARIANT \* vNumHits);

Retrieves statistic results.

*iFlags*

|               |   |      |
|---------------|---|------|
| rfByHandle    | Retrieve data for the area specified by iHandle only.                               | 0x01 |
| rfAllAreas    | Retrieve data for all areas.  | 0x02 |
| rfFilterTask  | Include only data within specified task. Use with rfByHandle or rfAllAreas.         | 0x10 |
| rfFilterValue | Include data results for a specific value of a data area only. Use with rfByHandle. | 0x20 |

*iHandle*

Handle of the area for which the statistics should be retrieved. Used with *rfByHandle*.

*iTask*

Task ID for which the statistics should be retrieved. Used with *rfFilterTask*.

*iValue*

Data value for which the statistics should be retrieved. Used with *rfFilterValue*.

*iNumStatistics, vNumStatistics*

Receives the number of statistics collected.

*vHandle, vValue, vTotalTime, vMaxTime, vMinTime, vNumHits*

Arrays of statistics data. The array elements are index correlated (index *i* on each array corresponds to the same statistic).

*vHandle*: area handle

*vValue*: statistic value

*vTotalTime*: total time spent

*vMaxTime*: maximum time spend in a single hit

*vMinTime*: minimum time spend in a single hit

*vNumHits*: number of hits

**HRESULT ProfilerGetHistory**(  
 [in] ProfilerResultFlags iFlags,  
 [in] long iHandle,  
 [in] long iTask,  
 [in] long iValue,  
 [in] long iHistoryBase,  
 [in] [out] long \* iNumHistories,  
 [in] [out] VARIANT \* vHandle,  
 [in] [out] VARIANT \* vValue,  
 [in] [out] VARIANT \* vTime);

**HRESULT ScriptProfilerGetHistory**(  
 [in] ProfilerResultFlags iFlags,  
 [in] long iHandle,  
 [in] long iTask,  
 [in] long iValue,  
 [in] long iHistoryBase,  
 [in] [out] VARIANT \* vNumHistories,  
 [in] [out] VARIANT \* vHandle,  
 [in] [out] VARIANT \* vValue,  
 [in] [out] VARIANT \* vTime);

Retrieves profiling history data for the specified area or areas.

*iFlags*

|               |   |      |
|---------------|---|------|
| rfByHandle    | Retrieve data for the area specified by iHandle only.                               | 0x01 |
| rfAllAreas    | Retrieve data for all areas.  | 0x02 |
| rfFilterTask  | Include only data within specified task. Use with rfByHandle or rfAllAreas.         | 0x10 |
| rfFilterValue | Include data results for a specific value of a data area only. Use with rfByHandle. | 0x20 |

*iHandle*

Handle of the area for which the statistics should be retrieved. Used with **rfByHandle**.

*iTask*

Task ID for which the statistics should be retrieved. Used with **rfFilterTask**.

*iValue*

Data value for which the statistics should be retrieved. Used with **rfFilterValue**.

*iHistoryBase*

Specify the zero based offset of the history. To get all history entries in multiple calls use this value to specify the offset inside history buffer.

*iNumHistories, vNumHistories*

On input it holds the size of the Handle, Value and Time buffers. On output it receives the number of history entries collected.

*vHandle, vValue, vTime*

Arrays of history data. The array elements are index correlated (index i on each array corresponds to the same history entry).

vHandle: area handle

vValue: statistic value

vTime: hit time in nanoseconds